D_{++}: Structural Credit Assignment in Tightly Coupled Multiagent Domains

Aida Rahmattalabi, Jen Jen Chung, Kagan Tumer

***Abstract*— Autonomous multi-robot teams can be used in complex coordinated exploration tasks to improve exploration performance in terms of both speed and effectiveness. However, use of multi-robot systems presents additional challenges. Specifically, in domains where the robots' actions are tightly coupled, coordinating multiple robots to achieve cooperative behavior at the group level is difficult. In this paper, we demonstrate that reward shaping can greatly benefit learning in multi-robot explorations tasks. We propose a novel reward framework based on the idea of *counterfactuals* to tackle the coordination problem in tightly coupled domains. We show that the proposed algorithm provides superior performance (166% performance improvement and a quadruple convergence speed up) compared to policies learned using either the global reward or the difference reward [1].**

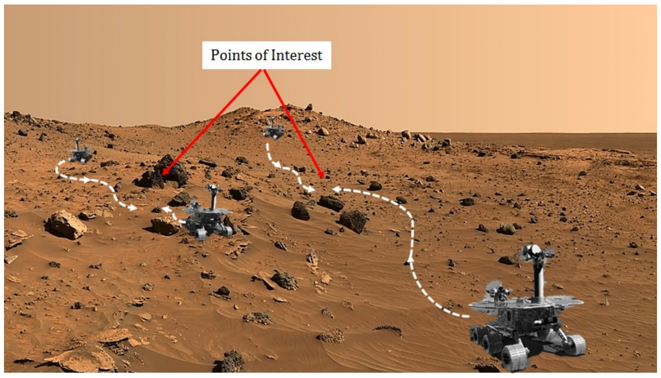
I. Introduction

Autonomous multi-robot teams can be used in complex exploration tasks for improved information gathering performance over single robot systems, in terms of both speed and effectiveness. However, multi-robot coordination is a complex control problem, especially in tightly-coupled tasks which involve a mutual dependence of the robots on each other's performance [2]. In such cases, the performance of a multi-robot system is highly sensitive to the level of coordination among the individual robots' policies.

In many multi-robot problems, such as space exploration, environmental monitoring, and search and rescue tasks, only a high level description of the mission is at hand, and agents only have access to local sensor information to decide their actions. Communication, which is a tool to improve the performance of cooperative agents is also often expensive or limited. Effective team performance becomes even more challenging when robots' actions are tightly coupled requiring agents to extensively coordinate their actions in order fulfill their mission. In such cases, coordinated policies can be difficult to define a priori and distributed policy learning is often employed to optimize team strategies.

Distributed policy learning has been demonstrated to produce effective team performance for loosely coupled tasks [3]–[6]. In particular, reward-shaping techniques [7]–[9] have been used to address the structural credit assignment problem for implicit coordination solutions where inter-robot communication is unavailable. The difference evaluation function [1], [10] is a shaped reward that makes use of *counterfactuals* to query the direct effect of an individual's contribution to

Aida Rahmattalabi, Jen Jen Chung, Kagan Tumer are with the Autonomous Agents and Distributed Intelligence Lab, School of Mechanical, Industrial and Manufacturing Engineering, Oregon State University, OR, 97330, USA {rahmatta, jenjen.chung, kagan.tumer}@oregonstate.edu



http://www.teciber.com/wp-content/uploads/2015/03/marte.jpg

Fig. 1: Rover domain representation with sample robot paths. Multiple simultaneous observations (in this case two) must be made of a particular POI to have any value to the system. Starting from different locations, well-coordinated robots navigate to the POIs and make observations.

the team performance. This reward signal can be computed from locally available information [11], it is both *aligned* to the system evaluation and *sensitive* to the individual robot's action, and provides an effective training signal when the system reward function is smooth.

Missions that require tightly coupled actions introduce reward functions that are inherently non-smooth and often involve step changes that represent the specific multi-robot coupling requirements of the tasks. Thus, existing reward-shaping techniques do provide an adequate evaluation signal for learning joint policies that are necessary for achieving such tightly coupled tasks.

In order to improve policy learning for these problems, we extend the idea of the difference reward and explore the manipulation of counterfactuals to provide robots with a stronger feedback signal on potential joint actions. The counterfactual that we propose computes the effect of introducing multiple identical robots to the system. Executing this targeted sweep over the number of available robots provides a rapid estimate of the joint action required to achieve a task. The proposed reward framework, D_{++}, is used to compute the evaluation signal in a cooperative coevolutionary algorithm for training the neural network control policies of each robot.

We demonstrate our proposed reward framework on an environmental monitoring scenario that requires robots to perform multiple simultaneous observations of points of interest (POIs) to successfully gather information. We evaluate our method in two different monitoring tasks, one which is agnostic to the types of robots that form an observation team (*homogeneous* case), and one which requires a specific set of *heterogeneous* robot types to successfully perform the task.

We show that with D_{++}, the team of robots is able to greatly improve policy learning rates over existing shaped rewards in terms of the overall team information gathering performance. Furthermore, in cases where learning via the the global system evaluation or difference reward completely fail to achieve an effective policy, the team policies learned using D_{++} are able to successfully achieve the mission objectives.

II. Problem Formulation

The problem that we investigate in this paper is multi-robot coordination in a tightly coupled environmental monitoring task. It consists of a set of homogeneous rovers on a two dimensional plane that must observe a set of POIs within a given time window [1], [5]. It is of note that the problem domain as described in this section, only consists of homogeneous agents. In Section VII, the problem domain will be redefined for the heterogeneous case. Prior knowledge of POIs, such as their locations, the number of POIs and the utility of observing any particular POI is not known and robots must learn to coordinate such that the team's utility function is maximized.

The key difficulty of this domain is that multiple simultaneous observations of a POI are required otherwise no reward is received by the team for those observations. Figure 1 illustrates the basic problem setup. A POI is considered observed only if $m > 0$ robots observe that POI from within a specified but unknown sensing distance. This problem formulation ensures that team coordination is essential to the completion of the task.

To formalize the problem, we first focus on the simple case where $m = 2$ observations are required in order to observe a POI. In this case, if more than two robots observe a POI, only the observations of the closest two robots are considered and their observation distances are averaged in the computation of the global system evaluation G, which is formulated as:

$$G(\mathbf{z}) = \sum_i \sum_j \sum_k \frac{V_i N^1_{i,j} N^2_{i,k}}{\frac{1}{2}(\delta_{i,j} + \delta_{i,k})}, \tag{1}$$

where $\mathbf{z}$ is the joint state-action of the team, V_i is the value of observing the i-th POI, and $\delta_{i,j}$ is the distance between the j-th robot, and the i-th POI. The variables $N^1_{i,j}$ and $N^2_{i,k}$ indicate whether robots j, k were within the observation distance δ_0 and were the closest two robots to the i-th POI. That is,

$$N^1_{i,j} = \begin{cases} 1, & \text{if } \delta_{i,j} < \delta_0 \text{ and } \delta_{i,j} < \delta_{i,l},\ \forall l \neq j, \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

$$N^2_{i,k} = \begin{cases} 1, & \text{if } \delta_{i,k} < \delta_0 \text{ and } \delta_{i,k} < \delta_{i,l},\ \forall l \neq j,k, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

The overall team objective is to maximize the global system evaluation (1).

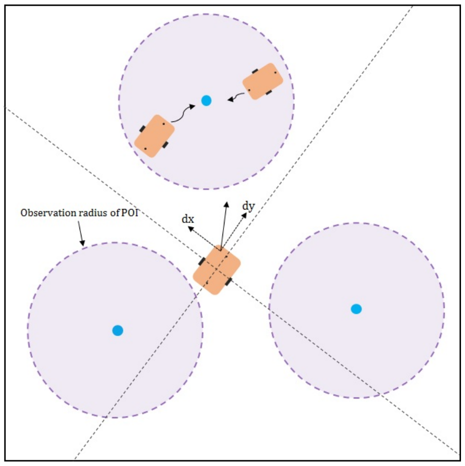


Fig. 2: Diagram of the rover domain. The world is broken up into four quadrants relative to the robots position and orientation. POIs and fellow robots that are observed in each quadrant are summed resulting in 8 state input variables. At each timestep, the robot's neural network controller yields two continuous outputs $[d_x, d_y]$, which determine the robot's motion in the next timestep. Each POI has an observation radius such that only robots within that radius are able to observe that POI.

Sensor detections of POIs and other robots are discretized into four quadrants, (*north-east, north-west, south-west, south-east*), with respect to the robot body frame (see Fig. 2). The detections within each quadrant are used to compute the state input vector of the learned neural network controller. More specifically, the state variable representing robot detections in quadrant q for robot j is defined as:

$$s_{ROB_{j,q}} = \sum_{j' \in J_q} \frac{1}{L_{j'}}, \tag{4}$$

where J_q is the set of the observed and estimated robots in quadrant q, and $L_{j'}$ is the relative distance from robot j to robot $j^{'}$. The state variable representing POI detections in quadrant q of robot j is defined as:

$$s_{POI_{j,q}} = \sum_{i \in I_q} \frac{V_i}{L_i}, \tag{5}$$

where I_q is the set of observed POIs in quadrant q, and L_i is the relative distance between robot j and POI i. These state variables give an approximate representation of the world, reducing the location and number of robots and POIs in each quadrant to a representative value.

Each member of the team of robots executes a policy described by a neural network. Known as universal approximators [12], neural networks have the ability to model continuous state-action control policies given only a coarse representation of the system state [10]. To train a neural network, the weights will be adjusted in such a way that given the current state as an input, the network returns an action that maximizes a particular utility function. The performance of the learned policy is strongly dependent on the utility function used to evaluate policies.

In this work, we train neural network controllers with cooperative coevolutionary algorithms (CCEA) while we use the D_{++} evaluation function as the fitness function and we compare the results against those trained on the difference (D) and global (G) evaluation functions.

III. Background

A. Fitness Function Shaping

In many multi-robot coordination domains there is a difference between maximizing the system-level fitness function and maximizing a single agent's fitness value. This can be referred to as reward shaping and is aimed at enhancing learning by providing agents with a reward signal that is more sensitive to their own actions while still reflecting the system's global objective. In this context, Hoen and De Jong shaped the utilities of the agents such that an agent maximizing its individual utility would act to also increase the system evaluation function [13]. This work is similar to that of Agogino and Tumer, and Knudson and Tumer, who utilized difference evaluations as fitness functions to improve coordination in multiagent systems. [1], [14]. Difference evaluation function is a shaped reward that uses *counterfactual* agents to provide agent-specific rewards. Section below will elaborate on the definition and properties of difference evaluation function which provides the necessary background for the present proposed reward framework, D_{++}.

B. Difference Evaluation Function

The difference evaluation function [1] is a shaped reward signal that provides agent-specific evaluation by removing a large amount of the noise created by the actions of other agents in the system [15]. It is defined as:

$$D_{\mathbf{i}} = G\left(\mathbf{z}\right) - G\left(\mathbf{z}_{-i} \cup \mathbf{c}_i\right), \tag{6}$$

where $\mathbf{z}$ is the joint state, $G(\mathbf{z})$ is the global system performance, $\mathbf{z}_{-i}$ are all the state-actions on which agent i has no effect and $\mathbf{c}_i$ is the *counterfactual* term, which is a fixed vector used to replace the effects of agent i. Thus, $G(\mathbf{z}_{-i} \cup \mathbf{c}_i)$ is the evaluation of a theoretical system without the contribution of agent i. Any action taken by agent i to increase D_i simultaneously increases G. This property is termed alignment, and is a key property of any shaped reward. Further, the second term in (6) removes portions of $G(\mathbf{z})$ that are not related to agent i, resulting in an improved signal-to-noise ratio. In other terms, agent i's impact on D_i is much higher than its relative impact on $G\left(\mathbf{z}\right)$ [16]. This property is termed sensitivity. Both [10], [17] have previously shown that the alignment and sensitivity properties of the difference evaluation function results in agent-specific feedback, which leads to superior learning performance.

IV. Approach

In this section, we elaborate on the coordination algorithm. Since the major contribution of this paper is on shaping reward in tightly coupled multiagent domains, we start by introducing our proposed reward function, D_{++}.

A. D_{++}: An Extension to the Difference Reward in Tightly Coupled Multiagent Domains

One of the challenges in distributed learning is that with no prior knowledge of the task and limited communication capabilities, agents must randomly search the state-action space until they accomplish the system objective and receive a reward signal to evaluate their policy. In highly coupled domains, however, the task cannot be accomplished unless tight coordination among the agents is established and maintained. This poses a great challenge to learning agents since the probability of agents simultaneously executing the right action is very low. Thus, agents receive no feedback on their policies in large portions of the joint space. To illustrate, in the case of the exploration problem discussed in this paper, consider a scenario where three robots are required to simultaneously observe a POI. The robots start from random locations and should synchronize their policies to reach the POIs simultaneously. However, it is unlikely that randomly moving robots are able to learn this task within a reasonable amount of time since only after all three robots successfully observe a POI do they receive a reward. In the absence of system feedback they are not capable of evaluating any potential improvement in their policies.

It is clear that the key problem is to be able to distinguish between a case where two robots are within the observable region of a POI and they only need one more robot to complete their subteam, and a case where all three of them are wandering without purpose in the environment.

Any reward function should enable agents to evaluate their actions in the absence of their teammates. This can lead to a huge reduction in the search space and an increase in the probability of success. Building upon the above argument, we extend the idea of *counterfactuals* in the difference reward and we propose a reward function, referred to as D_{++}, to address this problem. The *counterfactual* that we propose computes the effect of introducing multiple identical agents to the system and compares it to the current state of the system. The mathematical formulation of D_{++} is,

$$D_{++}^{n}\left(i\right) = \frac{G\left(\mathbf{z}_{+(\cup_{\mathbf{i=1,\ldots,n}})\mathbf{i}}\right) - G\left(\mathbf{z}\right)}{n}, \tag{7}$$

In the above equation, $\mathbf{z}_{+(\cup_{\mathbf{i=1,\ldots,n}})\mathbf{i}}$ indicates all the states on which agent i has added n identical agents. Furthermore, the division by n is for the purpose of normalization. Since we assume no prior knowledge on the number of agents required to fulfill a task, the agents should iteratively increase the value of n.

Notably, in the above formulation of D_{++}, if the value of n is set to -1, D_{++} will yield the definition of D as described in (6) since the *counterfactual* term $\mathbf{c}_i$ is an arbitrary vector which can indeed be set to $\vec{0}$.

As in the difference reward, agents optimize the global system objective by calculating each agents' contribution to the system performance. This is calculated based on the difference an agent makes by executing a certain action. This idea is further extended in D_{++}, which evaluates how the introduction of multiple identical agents benefit the system. It is noteworthy that in tightly coupled tasks, D does not provide any feedback unless tight coordination among the agents is established and maintained. On the other hand, D_{++} fails to provide this gradient information where sufficient number of agents have coordinated. So it is reasonable to leverage both D and D_{++} to the benefit of the system. This leads us to the final definition of our reward framework which is based on the idea of both removing and adding agents, described by D and D_{++}, respectively.

At this stage, the reward shaping will be combined with a search problem on different numbers of *counterfactual* agent additions. In order to guide the search and improve the computation, we use the basic assumption that in any state of the robot, the entire multiagent system must be able to accomplish the task. In other words, in the present problem domain, if a robot is within the observable region of a POI, by adding the entire team of robots as *counterfactual* agents, the system can make an observation of that POI. This simple assumption allows us to evaluate whether there exists a solution before we begin incrementing *counterfactual* agents in D_{++} algorithm. Algorithm 1 summarizes our approach.

Algorithm 1 D_{++}

```
1:  calculate $D_{++}^{-1}$ using (7)
2:  calculate $D_{++}^{totalAgents-1}$ using (7)
3:  **if** $D_{++}^{totalAgents-1} == 0$ **then**
4:      Return $D_{++}^{-1}$
5:  **else**
6:      $n \leftarrow 0$
7:      **repeat**
8:          $n = n + 1$
9:          Calculate $D_{++}^{n}$ using (7)
10:         **if** $D_{++}^{n} > D_{++}^{n-1}$ **then**
11:             Return $D_{++}^{n}$
12:     **until** $(n \leq totalAgents - 1)$
13: Return $D_{++}^{-1}$
```

Algorithm 1 begins with the calculation of $D = D_{++}^{-1}$ and $D_{++}^{totalAgents-1}$. It is noteworthy that $totalAgents - 1$ plus the agent itself make up the entire team of agents. If $D_{++}^{totalAgents-1}$ is equal 0, it means that there is no possible combination of existing robots that can make an observation and receive a reward from the system. This corresponds to a case where an agent is outside the observable region of all the POIs and thus it is not possible to observe any of the POIs. It can easily be seen that by using this knowledge, we can greatly reduce the computation, eliminating the need to loop over all the values of n before realizing all this computation is in vain.

Given the above definition of the D_{++}, we will dedicate the following two sections to describe the learning framework adopted in this work and and how the D_{++} is incorporated in the learning procedure.

B. Evolutionary Algorithms

Evolutionary Algorithms (EAs) are a class of stochastic search algorithms that often outperform classical optimization algorithms [18]. Typically, EAs contain three basic mechanisms: solution generation, mutation, and selection. Starting with an initial set of candidate solutions (the population), these mechanisms are used to generate new solutions and retain existing solutions that show improvement based on a system utility. Simple EAs can be applied to a variety of single-agent learning tasks, however, when dealing with large cooperative multi-robot problems, modifications are required for an effective performance. One such modification is coevolution, where multiple populations evolve simultaneously in order to develop policies for interacting agents.

C. Cooperative Coevolutionary Algorithms

The standard Cooperative Coevolutionary Algorithm (CCEA) [19] has been used in this study as a base algorithm for learning control policies [15]. CCEAs are an extension of EAs for multiagent systems and have been shown to perform well in cooperative multiagent domains [20]. The standard CCEA is detailed in Algorithm 2. In CCEA, N coevolving populations of neural networks (neuro-controllers) are utilized to form teams comprised of N agents. One member of each population is extracted for each agent to form a team which then operates in the problem domain. At each generation, k mutated networks are generated in each population by mutating the parent networks. Then, $2k$ teams of agents are formed and simulated. The performance of each simulated team is subsequently evaluated using a fitness function, $F(z)$ and assigned to every agent in the team. In this paper, we propose D_{++} as the fitness function, and we compare it to using G and D. Finally, k networks from each population are selected based on greedy selection to proceed to the next generation. This process is repeated for a set number of generations.

Algorithm 2 Standard CCEA

```
1:  Initialize $N$ populations of $k$ neural networks
2:  **for** Generation **do**
3:      **for** Population **do**
4:          produce $k$ successor solutions
5:          mutate successor solutions
6:      **for** $i = 1 \rightarrow 2k$ **do**
7:          randomly select one agent from each population
            without replacing it into population pool
8:          add agents to team $T_i$
9:          simulate $T_i$ in domain
10:         assign fitness to each agent in $T_i$ using $F(z)$
11:     **for** Population **do**
12:         select k solutions using e-greedy selection
```

D. Computational Complexity

Computing the difference reward requires two calculations of the global evaluation function. Robots learning using the global reward only call $G(\cdot)$ once for each policy evaluation. For D_{++} however, this number is increased since agents must continue adding *counterfactual* agents until they reach a non-zero value, and each iteration requires additional calls to $G(\cdot)$. This adds to the computational time of the learning algorithm. Table I compares the computational complexity of each evaluation as the number of calls to $G(\cdot)$ for evaluating a single policy.

TABLE I: COMPARISON OF NUMBER OF CALLS TO G FOR THREE DIFFERENT REWARD FUNCTIONS

Evaluation function	G	D	D_{++}
Calls to G	1	2	$c_i + 3$

In Table I, c_i is the the minimum number of *counterfactual* agents needed to be added by agent i in order to receive a nonzero value. This term is bounded between $[0, \textit{total number of agents} - 1]$ since we assume the entire team of robots should be able to accomplish the task. In cases where sufficient number of agents are present to complete a coupled task, c_i is equal to 0 (i.e. D provides a sufficient learning signal). In cases where adding any number of agents cannot increase the system performance, c_i is also 0. This is because, when calculating D_{++} reward, first ($\textit{total number of agents} - 1$) agents are added and if the returning reward is zero, it can be understood that iterating over fewer $counterfactual$ agents will similarly not yield a nonzero reward. An example of this situation in the present exploration domain is when a robot is outside the observable range of a POI, where adding any number of agents cannot fulfill the observation task. Lastly, if not enough robots are within the observable radius of a POI, c_i takes a value between $[1, \textit{required number of observations} - 1]$. We will revisit this problem in the results section where we show that the average value of c_i decreases for the team of robots throughout learning, making it increasingly more efficient to use D_{++} as the learning signal.

V. EXPERIMENT SETUP

To investigate the performance of the proposed shaped reward, different sets of experiments will be presented. We compare the performance of D_{++} learners against those using the difference evaluation function signal and global evaluation signal. The experiments are performed in simulation and the robot policies are evaluated and learned offline.Generally, at every timestep, each robot uses the policy encoded by the neural network controller to determine two controls in the (x, y) directions. Each controller is an 8-input, 9-hidden unit, 2-output fully connected feedforward neural network. These neural network controllers are randomly initialized.The weights are adjusted via the (CCEA) algorithm, explained in algorithm 2. The eight inputs are the state variables as defined in (4) and (5). The neural network output provides two values that lie in $[0, 1]$ and each of these are mapped to one of the two controls. Each agent has a population of 15 policies, initialized using a normal distribution $\mathcal{N}(0, 1)$.

In the following experiments, we first focus on the homogeneous multi-robot system for which the survey region is 30×30 units in size, containing a fixed number of randomly distributed POIs. Each POI requires a fixed number of simultaneous observations within its observation range, which is defined as r_{POI} = 4.0. The maximum allowable robot movement is defined as $d_{max} = 1$ unit/timestep. In this problem, we assume full observability for the robot sensors. In future work, we will address partial observability and its effect on the performance of the proposed learning algorithm.

VI. RESULTS: HOMOGENEOUS TEAMS

Two sets of experiments are conducted. In both experiments, a team of 12 robots explore the region to observe 10 POIs. In the first experiment, each POI requires 3 simultaneous observations. In the second experiment, in order to analyze how the learning algorithm performs with tighter agent coupling, the number of required observations for each POI was doubled.

A. 12 Robots, 10 POIs, 3 Required Simultaneous Observations

Averaged learning results associated with all three reward functions, G, D, D_{++} with error bars reporting the standard error in the mean are given in Fig. 3. These results show that agents learning with the D_{++} evaluation signal outperform agents using global evaluations or the difference evaluation function alone. In fact, D_{++} produces policies that perform up to 100% better that those of D and at a quadruple speedup in the learning rate.

Figure 4 shows the paths of the team of robots after being trained with the D_{++} reward function. The dots in magenta are the POIs, and they vary in size based on their value to the system. The POIs are spread across the exploration region. Robots are initially located in the center and should spread and explore the area of interest. As seen in Fig. 4, robots have successfully formed teams of three robots and have coordinated to observe multiple POIs in their proximity. The significance of these results is that team formation has been performed in an implicit manner, requiring no communication among agents or any prior information about other agents' intended behavior. This approach is particularly useful in cases where communication is limited or expensive since it can provide more reliable and robust results.

B. 12 Robots, 10 POIs, 6 Required Simultaneous Observations

To investigate the ability of the proposed algorithm in dealing with tighter agent couplings, we doubled the number of required simultaneous observations. Averaged results over 20 trials are shown in Fig. 5. These results indicate that as the coupling of the system increases, both G and D lose applicability since learning algorithms using either of these reward functions relies heavily on multiple agents simultaneously selecting the right actions, which is highly unlikely

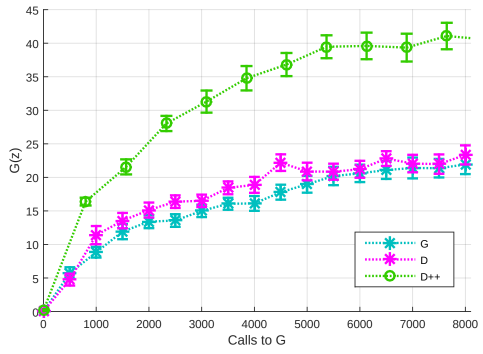


Fig. 3: Observation performance of policies trained on G, D, D_{++} for 12 robots, 10 POIs each requiring 3 simultaneous observations.

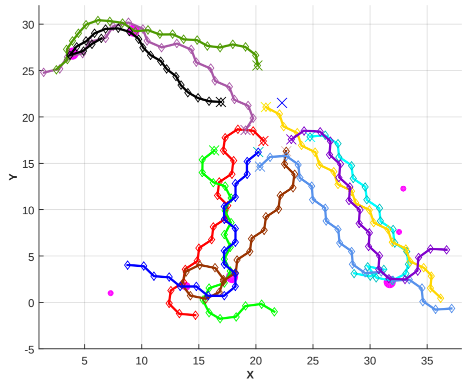


Fig. 4: Robots paths executed by policies learned using the D_{++} reward function. POIs are represented as pink circles; larger circles indicate that the POI has a higher observation value.

in tightly coupled domains. However, D_{++} overcomes this issue by using *counterfactual* agents. In particular in the rover domain, D_{++} rewards a policy that leads an agent to the observable region of a POI. Even if insufficient number of robots are available to accomplish the observation task; it promotes coordination by rewarding the *stepping stones* that ultimately lead to achieving the system objective.

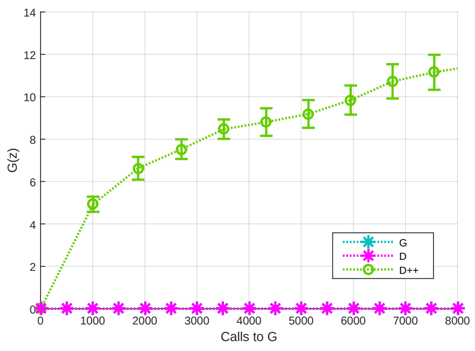


Fig. 5: Observation performance of policies trained on G, D, D_{++} for 12 robots, 10 POIs each requiring 6 simultaneous observations.

C. Computation Time Analysis

Recall from Section IV-D that while calculating D_{++} one must continue introducing *counterfactual* agents until the resulting reward provides a gradient that can be used as the

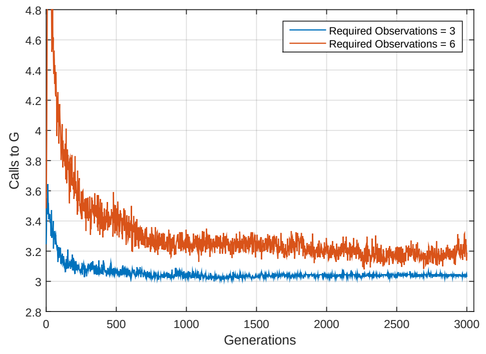


Fig. 6: Average calls to G across progressive learning generations in the D_{++} calculation for two cases where POIs require 3 and 6 simultaneous observations, respectively. Averages over 20 trial runs are plotted along with 95% confidence intervals.

reward. However, as previous results on the performance of policies trained using D_{++} show, agents learn to coordinate relatively quickly compared to the results from using either G or D. This results in a decrease in the number of required *counterfactual* agents and also a decrease in the computation effort needed. Figure 6 supports the above argument by indicating the average required number of simulated *counterfactual* agents in D_{++} calculations during training. As shown in Fig. 6, there is a sharp decrease in the calculations required in D_{++} within the first 500 generations.

VII. Heterogeneous Team Formation

So far, we have only considered robots that are homogeneous. However, some tasks may require multiple agents of differing construction and capabilities to cooperate and perform a task. Considering heterogeneous robots in a multiagent domain adds a great deal of potential and power at the price of added complexity. In such domains, team formation is indeed more challenging since each agent must both identify the subset of agents that it needs to collaborate with and to synchronize its action with its teammates in order to fulfill the task.

A. D_{++} for Heterogeneous Agents

In this section, we extend the D_{++} algorithm to handle coordination of heterogeneous agents. We revisit the rover exploration domain described in Section II, and this time we assume there are t types of robots. We also assume that each robot is able to determine that there are t types of robots in the environment and also the number of robots of each type. Finally, in order for an observation to be counted as successful, n_i agents of each type i should simultaneously observe the POI within its observation radius, that is, a total of $\sum_{i=1}^{t} n_i$ agents must be present in the observable region surrounding each POI. Comparing to the homogeneous case described before, it is clear that coordination is more challenging since the probability that the sufficient number of agents of the exact required types select the right actions to make a successful observation is even further reduced. Thus we need a coordination mechanism that is able to provide the useful incentive for each agent and to guide them toward the more promising policies even in the absence of their teammates. In order for D_{++} to account for such heterogeneity, we need to slightly modify the search procedure. This time, we start by assigning $\vec{n} = \vec{0}$, where $\vec{n}$ is a vector of size t, equal to the number of types of robots. $\vec{n}$ holds the number of *counterfactual* agents needed to be added for each type in order to receive a non-zero reward. In order to find $\vec{n}$, we add one agent of each type at each step, and we continue incrementing the number of agents of all types until we reach a non-zero value for the reward. Algorithm 3 summarizes the approach.

Algorithm 3 D_{++} (Heterogeneous Agents)

```
1:  calculate $D_{++}^{-1}$ using (7)
2:  calculate $D_{++}^{totalAgents-1}$ using (7)
3:  $\vec{n} \leftarrow \vec{0}$
4:  if $D_{++}^{totalAgents-1} == 0$ then
5:      Return $D_{++}^{-1}$
6:  else
7:      repeat
8:          for $t = 1 : totalTypes$ do
9:              $n_t = n_t + 1$
10:             Calculate $D_{++}^{n}$ using (7)
11:             if $D_{++}^{n} > D_{++}^{n-1}$ then
12:                 Return $D_{++}^{n}$
13:     until $(n \leq totalAgents - 1)$
14: Return $D_{++}^{-1}$
```

VIII. Results: Heterogeneous Teams

In this section, two sets of results on the heterogeneous exploration domain are presented. In both experiments, $t = 3$ types of robots are considered. In the first experiment, each POI requires at least one robot of each type for a successful observation. In the second experiment, we increase the level of heterogeneity by requiring one robot of the first two types, $t = 1, 2$ and 3 robots of type three, $t = 3$, to simultaneously observe a POI. This will indicate the power of our algorithm in dealing with different levels of system heterogeneity. Similar to the homogeneous case, the survey region is 30×30 units in size and each POI has a fixed observation radius of r_{POI} = 4.0. Also, the maximum allowable robot movement is $d_{max} = 1$ unit/timestep as defined before. The following results are generated from 20 statistical trials.

A. 9 Robots, 15 POIs, Simultaneous Observations of 3 Agents of 3 Different Types Required

In this experiment, we assume that there are 3 different types of robots, each having a particular capability required for the successful observation of the POIs. A POI is counted as observed only if it is simultaneously observed by at least one robot of each type. Figure 7 indicates the learning curves comparing performance of policies learned by three different functions, G, D and D_{++}. The learning curves are plotted against number of calls to G to account for the required computation of each reward function. As seen, with the same

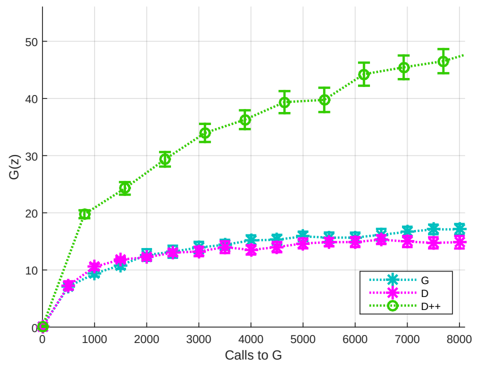


Fig. 7: Observation performance of policies trained on G, D, D_{++} for 9 robots, 15 POIs each requiring simultaneous observations of the robots, one of each type t.

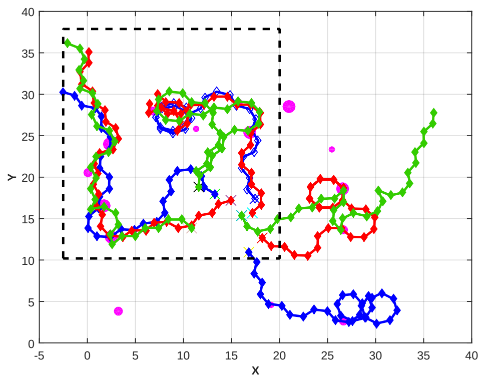


Fig. 8: Robots paths executed by policies learned using the D_{++} reward function. Different colors of *blue*, *green* and *red* are used to represent different types of robots in the system. As seen, two groups of robots, bounded in the dashed box, consist of all the three required types and have successfully formed teams to explore the POIs in that region. However, the remaining robots are still optimizing their policies trying to observe the remaining POIs.

number of calls to G, policies learned using G and D perform similarly while D_{++} yields policies that perform about 1.5 times better and up to four times faster compared to the results of G and D.

Figure 8 is a snapshot of the exploration domain which depicts the actual robot paths executed by policies learned using the D_{++} reward function. The path are colored in *blue*, *green* and *red*, each representing a different type of robot in the system. The POIs are marked with magenta dots with varying sizes, indicating their importance to the system. As seen, the robots, starting from the center of the domain, have formed two distinctive teams consisting of one robot of each type which have successfully coordinated to make observations of the POIs along their paths. Meanwhile, three robots, shown outside of the dashed box, have not fully coordinated.

B. 9 Robots, 15 POIs, Simultaneous Observations of $[1, 1, 3]$ Agents of 3 Different Types Required

This experiment measures the performance of the D_{++} algorithm when the level of the system heterogeneity increases. For this purpose, we assume that each POI must be

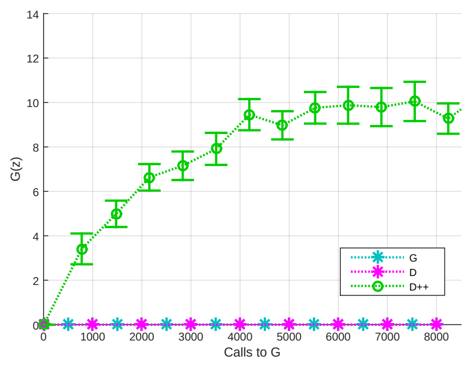


Fig. 9: Observation performance of policies trained on G, D, D_{++} for 9 robots, 15 POIs each requiring simultaneous observations of a total of 5 robots, including one of each types of $t = 1$ and $t = 2$, and three robots of type $t = 3$.

simultaneously observed by at least five distinctive robots, one from each of the first two types, and three of the third type. An equivalent problem in a real world scenario can be where we have different robot types each equipped with different sensors and tools to explore an unknown area. In such a case, three robots can assist in robots localization, and the second and third type may provide the tools to perform excavation and data collection. Figure 9, indicates the performance of the learned policies in a stimulated scenario. It is clear from these results that training using either G or D fails to yield a high-reward policy. This can be explained by the fact that both of these reward functions rely on multiple agents simultaneously selecting the right actions, which is highly unlikely in such a tightly coupled problem. The problem is even more severe compared to the homogeneous domain since the robot types also matter and any coordination algorithm must account for that as well. In contrast to G and D, D_{++} manages to reward policies that are *stepping stones* to the ultimate system objective even in the case of agent heterogeneity.

IX. Conclusion

In this paper, we developed control policies for a team of robots in a tightly coupled environmental monitoring task. We explored this problem along two dimensions, the degree of coupling and the heterogeneity of the multiagent system. We first proposed D_{++}, a novel reward framework which builds on the idea of *counterfactuals*. D_{++} was successfully tested in the homogeneous domain. Additionally, to account for system heterogeneity, the D_{++} reward function was extended to a domain with different types of robots, all required for the completion of the task.

Unlike the global evaluation function G and the difference evaluation function D, D_{++} promotes coordination by rewarding policies that are the *stepping stones* to accomplishing the system objective. Notably, as the coupling of the system increases, both G and D lose applicability since they rely heavily on multiple agents simultaneously selecting the appropriate actions, which is highly unlikely in tightly coupled domains. We also investigated the impact of system heterogeneity on the learned policies and We showed that policies trained on the D_{++} resulted in superior performance compared to those trained on global evaluations and difference rewards in terms of both convergence speed and performance.

References

[1] A. Agogino and K. Tumer, “Efficient evaluation functions for multi-rover systems,” in *Genetic and Evolutionary Computation–GECCO 2004*. Springer, 2004, pp. 1–11.

[2] B. P. Gerkey and M. J. Matarić, “Pusher-watcher: An approach to fault-tolerant tightly-coupled robot coordination,” in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 1. IEEE, 2002, pp. 464–469.

[3] P. Stone and M. Veloso, “Multiagent systems: A survey from a machine learning perspective,” *Autonomous Robots*, vol. 8, no. 3, pp. 345–383, 2000.

[4] M. Bowling and M. Veloso, “Simultaneous adversarial multi-robot learning,” in *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, vol. 3, 2003, pp. 699–704.

[5] L. Panait and S. Luke, “Cooperative multi-agent learning: The state of the art,” *Autonomous Agents and Multi-Agent Systems*, vol. 11, no. 3, pp. 387–434, 2005.

[6] K. Tuyls, P. J. T. Hoen, and B. Vanschoenwinkel, “An evolutionary dynamical analysis of multi-agent learning in iterated games,” *Autonomous Agents and Multi-Agent Systems*, vol. 12, no. 1, pp. 115–153, 2006.

[7] A. Y. Ng, D. Harada, and S. Russell, “Policy invariance under reward transformations: Theory and application to reward shaping,” in *International Conference on Machine Learning*, 1999, pp. 278–287.

[8] 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*. International Foundation for Autonomous Agents and Multiagent Systems, 2011, pp. 225–232.

[9] ——, “Dynamic potential-based reward shaping,” in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2012, pp. 433–440.

[10] A. Agogino and K. Tumer, “Efficient evaluation functions for evolving coordination,” *Evolutionary Computation*, vol. 16, no. 2, pp. 257–288, 2008.

[11] M. Colby, J. J. Chung, and K. Tumer, “Implicit adaptive multi-robot coordination in dynamic environments,” in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 5168–5173.

[12] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[13] J. Pieter and E. D. de Jong, “Evolutionary multi-agent systems,” in *Parallel Problem Solving from Nature-PPSN VIII*. Springer, 2004, pp. 872–881.

[14] M. Knudson and K. Tumer, “Coevolution of heterogeneous multi-robot teams,” in *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. ACM, 2010, pp. 127–134.

[15] M. Colby and K. Tumer, “Shaping fitness functions for coevolving cooperative multiagent systems,” in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 2012, pp. 425–432.

[16] D. H. Wolpert and K. Tumer, “Optimal payoff functions for members of collectives,” *Advances in Complex Systems*, vol. 4, no. 02n03, pp. 265–279, 2001.

[17] M. Colby and K. Tumer, “Fitness function shaping in multiagent cooperative coevolutionary algorithms,” *Autonomous Agents and Multi-Agent Systems*, pp. 1–28, 2015.

[18] D. B. Fogel, “An introduction to simulated evolutionary optimization,” *Neural Networks, IEEE Transactions on*, vol. 5, no. 1, pp. 3–14, 1994.

[19] M. A. Potter and K. A. De Jong, “A cooperative coevolutionary approach to function optimization,” in *Parallel problem solving from naturePPSN III*. Springer, 1994, pp. 249–257.

[20] S. G. Ficici, O. Melnik, and J. B. Pollack, “A game-theoretic and dynamical-systems analysis of selection methods in coevolution,” *Evolutionary Computation, IEEE Transactions on*, vol. 9, no. 6, pp. 580–602, 2005.