Fitness Shaping For Multiple Teams

Joshua Cook
Oregon State University
Corvallis, Oregon
cookjos@oregonstate.edu

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

ABSTRACT

Coevolutionary algorithms have effectively trained multiagent teams to collectively solve complex problems. However, in many real-world applications, changes to the environment or agent functionality require agents to function well with multiple *different* teams. In this paper, we provide a counterfactual-state-based shaped fitness evaluation that provides an agent-specific signal that promotes effective cooperation across a variety of teams. The key insight leading to this result is that the shaped fitnesses across multiple teams can be aggregated because those performances are independent of each other. As a result, this approach leads to a single signal that captures an agent's performance across multiple teams. We show that this method provides significant improvement over standard multiagent fitness-shaped methods in learning robust cooperative behavior.

CCS CONCEPTS

• **Computing methodologies** → **Multi-agent systems**.

KEYWORDS

Multiagent Learning, Fitness Shaping, General Teaming

ACM Reference Format:
Joshua Cook and Kagan Tumer. 2022. Fitness Shaping For Multiple Teams. In *Genetic and Evolutionary Computation Conference (GECCO '22), July 9–13, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3512290.3528829

1 INTRODUCTION

Evolutionary algorithms have proven successful in training decentralized multiagent systems such as robot soccer [1, 10, 24], UAV traffic control [2, 35, 37, 40], and multi-robot exploration [15, 21, 33]. In each of these examples, each agent in the system is trained to cooperate with only one team. However, the ability to cooperate with multiple teams is a desirable and even necessary trait in many real-world domains.

Consider autonomous exploration, where a group of robots exploring a site may eventually disband and aid other groups exploring other sites which may require different skills. The robots must learn to not only cooperate with their initial group but learn more general cooperation strategies to successfully aid multiple teams to explore multiple sites, as exemplified by Figure 1. Multiagent learning is effective in training *multiple* agents to cooperate as a *single* team [38]. Ad hoc teaming solves an orthogonal problem of training a *single* agent to interact with *multiple* teams of preexisting agents [34]. We address the problem of training *multiple* agents to cooperate with *multiple* teams to achieve a given task.

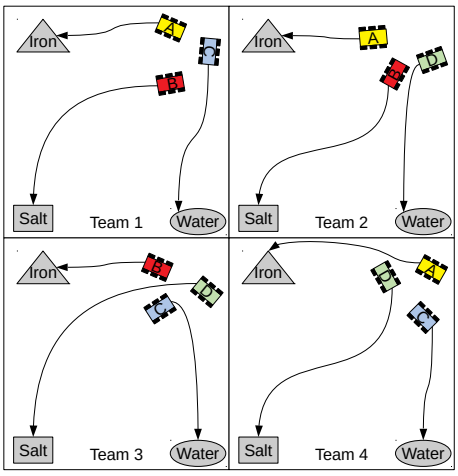

Figure 1: Agents A, B, C, D must collect iron, salt, and water on another planet in different teams. Each agent must learn its role in any of the four teams. For example, Agents A and C learn to go to the same resource in all teams, while agents B and D learn supportive roles, moving to the remaining ignored resources.

In this work, we provide an agent-specific, shaped fitness to evolve agents to perform in multiple teams. We first generate multiple teams to enable the agent to assess its potential contributions. Then, we evaluate that agent's performance through a new fitness evaluation that captures its aggregated performance across all teams. To ensure that fitness measures the agent's true potential, we derive a new shaped fitness that uses counterfactuals to evaluate an agent's contribution to multiple teams. This approach ensures that each agent receives a learning signal that promotes robust teamwork skills, but is significantly easier to optimize than the full system evaluation function.

The key insight of this work is that the counterfactual states an agent uses across multiple teams are independent of each other. Therefore, an agent optimizing the collective performance also optimizes its performance across arbitrary teams. Providing each agent with this agent-specific feedback trains multiple agents to cooperate with multiple teams.

The contributions of this work include:

GECCO '22, July 9–13, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9237-2/22/07…$15.00
https://doi.org/10.1145/3512290.3528829

(1) generating the right variety of teams to train an agent; and
(2) deriving an informative counterfactual-based shaped fitness that promotes cooperation across arbitrary teams.

Experimental results in a simulated multi-robot exploration domain show that the shaped fitness can effectively train agents to cooperate with all training teams. Further results show that optimizing this shaped fitness remains effective, regardless of the number of training teams.

2 BACKGROUND

2.1 Cooperative Co-evolutionary Algorithms for Multiagent Learning

Learning in multiagent domains introduces a unique set of challenges over single-agent domains. For example, multiple local agents must individually learn from global feedback [39]. Cooperative Co-Evolutionary Algorithms (CCEA) offer one approach to learning in these distributed domains. CCEA solves complex problems by dividing them into subproblems, then evolves solutions to each subproblem independently and in parallel [19, 25]. First, each subproblem is given a population of solutions. During evaluation, solutions from each population are used to generate a complete solution to the overarching problem. The evaluation of this complete solution is assigned to each of the partial solutions as their fitness [26]. In the application of CCEA to multiagent systems, the control policies of each agent are commonly used to represent the subproblems in training a team to complete an objective [27, 36].

Each control policy is frequently represented as a parameterized function approximator that maps its local state to an action. For every time step in an episode, each agent receives a local state view of the world, determines an action to take using its policy, then executes this action in the environment. At the end of the episode, the global performance describes how effective these collective joint actions are in this joint state. The agent parameters are coevolved to optimize the global objective function [18].

2.2 Fitness Shaping

The design of a fitness function can have a drastic impact on the converged solution; two fitness evaluations can promote the same behavior, with one being significantly easier to optimize than the other. The purpose of fitness shaping is to provide an alternative fitness function that is easier to learn from, but also optimizes the original function [22].

The global evaluation function used in cooperative multiagent learning is one example of a challenging evaluation to learn from. This introduces the credit assignment problem, where each agent in a multiagent system must learn from the global reward, which provides no information about each agent's contribution to the shared score [7]. With larger numbers of agents, this signal is difficult to learn from as the contribution of a single agent is trivialized by the impact of the other agents in the system.

Local evaluations partially ameliorate the credit assignment problem by providing agent-specific feedback, thus reducing the effect of other agents' actions on an individual's evaluation. However, without proper design of the local evaluations, an agent can learn competitive and selfish strategies, maximizing their objective instead of the team objective [3, 23]. The question arises, how does one design a local objective such that maximizing it also maximizes the global objective?

Fitness shaping introduces the concept of alignment to help answer the previous question. Two evaluations are considered aligned with each other if a change in the solution leads to an equal change in both evaluations. Optimizing a local fitness aligned with the global fitness is guaranteed to also optimize the global fitness [13].

Sensitivity is another relevant concept in fitness shaping for multiagent systems. This idea describes how sensitive an evaluation is with respect to a single agent's change in actions taken. More sensitive evaluations are easier to optimize as they more accurately describe an agent's individual performance [12]. The global evaluation represents a less sensitive function because it describes the performance of an entire team, in which a single agent may play a small part. Conversely, local evaluations are highly sensitive as they largely depend on the actions of the evaluated agent. These local rewards may be easier to learn, but lack alignment with the global objective. Ideally, the local rewards should be more sensitive than and aligned with the global reward.

Difference evaluations are shaped evaluations that are agent-specific and sensitive while maintaining alignment with the global objective [4, 29]. Equation 1 defines the difference evaluation for agent i as the difference of the global evaluation G, of the joint state-action of the team z, with and without agent i. The second term substitutes a counterfactual action c_i in place of agent i's action. Conceptually this provides an evaluation of agent i's contribution to the team:

$$D_i(z) = G(z) - G(z_{-i} \cup c_i) \tag{1}$$

One can easily show that this difference evaluation is aligned with the global evaluation by taking the partial derivative of the difference evaluation with respect to z_i, the state-action pair of agent i, and taking the same partial derivative of the global evaluation [4]. Both partial derivatives are equal to the partial derivative of the global evaluation, leading to the conclusion that both functions are aligned.

The high sensitivity of the difference evaluation is attributed to the inclusion of the counterfactual term. This term subtracts the team performance without agent i, largely removing the contributions of the other agents in the system from the original global evaluation. The overall evaluation describes agent i's contribution to the team, which is highly sensitive to the actions of agent i.

2.3 Ad Hoc Teaming

Learning to adapt to an arbitrary team is a useful skill for many multiagent domains. The goal of ad hoc teaming is to create an agent which can cooperate with a group of other agents with little or no prior coordination [34]. This is a challenging task as the other agents in an arbitrary team inject a high degree of uncertainty into the system, similar to the credit assignment problem. To reduce this uncertainty, multiple methods include agent modeling to determine what types of agents are present in the team [6, 8, 16]. With these agent models, an ad hoc teaming agent can better assess the impact of the other agents on the system. This provides an ad hoc teammate with the necessary information to identify its role within the team

and act on it. The actions are then determined by either a planning algorithm [9] or more commonly a learning agent.

Early learning-based approaches involved learning a model of the team, then selecting the corresponding policy for that team [5]. Other approaches focus on more specific ad hoc teaming scenarios such as dynamic team sizes [28] or dynamic teammates [30]. Each of these learning-based approaches involves training a *single* agent to interact with a team of other predefined agents. Conceptually, the single agent is trained to be able to join and contribute to pre-existing teams. This problem is fundamentally different from the multi-team, multiagent learning problem. Ad hoc teaming trains a single agent to operate with other already effective team members. In many applications, other high-quality agents may not be available. As a result, these other agents may also need to learn robust cooperative skills. This presents a multiagent learning problem where each agent must learn to cooperate with multiple teams.

3 FITNESS SHAPING FOR MULTI-TEAM LEARNING

We provide a fitness shaping approach to training *multiple* agents to cooperate with *multiple teams*. This can be viewed as an extension of multiagent learning, which trains *multiple* agents to cooperate with a *single* team. In standard multiagent learning, training a single team of agents does not leave much room to learn robust behaviors across multiple teams. To evolve cooperative policies across arbitrary teams, our fitness shaping method needs team variety.

We introduce a method of converting a multiagent task into a multi-team multiagent task. The first step is to create a pool of learning agents that is larger than the team size. Then, teams are formed by choosing different combinations of the agents. To maximize the number of learning teams, we generate all combinations of agents from the pool into the given team size. This provides the general team variation for our fitness evaluations.

The objective is to maximize performance in each of the generated teams. Optimizing multiple teams' evaluations would appear to be a multi-objective optimization problem, yet this is not the case. Each team evaluation uses the joint state of the respective team. The joint states are influenced by the composition of the team and are independent of one another. This leads to independence in team evaluations, meaning each team can increase their respective performance without a trade-off in other teams' performances. Thus, optimizing the aggregate team score optimizes scores of the individual teams [14, 20].

We define a new fitness function that promotes performance across these varied teams. The global multi-team fitness function is introduced in Equation 2. Here, z_j represents the joint state-action of the $j-th$ team out of n teams. Equation 2 is a sum of the global performances of all teams, promoting quality team performance across all teams.

$$G^{\Sigma}(z_1, z_2, ..., z_n) = \sum_{j=1}^{n} G(z_j) \tag{2}$$

This multiple team evaluation is an extension of the original single-team evaluation. In the new task, a larger number of agents must still complete the task defined by G, but in teams consisting of a variety of combinations of the agents. As a result, G^{Σ} provides a global performance of all agents, identical to any other multiagent global evaluation. This enables any multiagent learning algorithm, such as CCEA, to optimize multi-team performance without the need for modification. As an example CCEA, each agent would be represented as a population of solutions. To evaluate the agents, one solution from each population would be drawn. These solutions would be paired in the various teams and evaluated using G. Then each solution would be assigned a fitness as a sum of these G evaluations. This would be repeated for each solution in a population until all solutions have fitnesses assigned. Selection and mutation would occur and the process would repeat until some convergence criterion is met.

Similar to the global performance function in single-team systems, Equation 2 suffers from the credit assignment problem. Each agent struggles to discern their level of contribution to this global objective which can stifle learning. We seek an alternative shaped fitness that is aligned with this global objective but is more sensitive.

We present Equation 3 as our shaped fitness solution. This function can be seen as the difference evaluation equivalent of Equation 2. For each team's joint state-action global evaluation, we subtract the global evaluation of that team, j without agent i's contribution, $z_{j,-i}$ with the replacement counterfactual action, c_i. In this work, we use a null counterfactual, which represents an agent that never takes an action. Other counterfactuals, such as random actions or actions taken by other agents, could be used but are not studied in this work.

$$D_i^{\Sigma}(z_1, z_2, ..., z_n) = \sum_{j=1}^{n} G(z_j) - G(z_{j,-i} \cup c_i) \tag{3}$$

To justify using D^{Σ}, it needs to be aligned with G^{Σ} so that maximizing D^{Σ} will also maximize G^{Σ}. This can be shown by taking the partial derivative of both functions with respect to the joint state-action of a single agent. This shows how changes in an agent's decision, affect the performance evaluation. If both functions are aligned, a useful behavior defined by one function will be similarly labeled as a useful behavior by the other function.

We prove D^{Σ} and G^{Σ} are aligned by taking the partial derivative of D^{Σ} with respect to $z_{k,i}$, where $z_{k,i}$ represents the joint state-action pair of agent i from team k for all $k \in [1, 2, ..., n]$. The only joint state-action that depends on team k and contains the state-action from agent i is z_k. As a result, the partial derivative of the rest of the global evaluations is equal to zero.

$$\begin{aligned}
\frac{\partial}{\partial z_{k,i}} D_i^{\Sigma}(z_1, z_2, ..., z_n) &= \frac{\partial}{\partial z_{k,i}} \sum_{j=1}^{n} G(z_j) - G(z_{j,-i} \cup c_i) \\
&= \sum_{j=1}^{n} \frac{\partial}{\partial z_{k,i}} G(z_j) - 0 \\
&= \frac{\partial}{\partial z_{k,i}} G(z_k)
\end{aligned}$$

Next, we take the partial derivative of G^{Σ} with respect to $z_{k,i}$. Again, only one term depends on team k and contains the state-action pair

of agent i.

$$\frac{\partial}{\partial z_{k,i}} G^{\Sigma}(z_1, z_2, ..., z_n) = \frac{\partial}{\partial z_{k,i}} \sum_{j=1}^{n} G(z_j)$$
$$= \frac{\partial}{\partial z_{k,i}} G(z_k)$$

Thus, the partial derivatives of D_i^{Σ} and G^{Σ} with respect to $z_{k,i}$ are equal for each team, indicating full alignment.

$$\frac{\partial}{\partial z_{k,i}} G^{\Sigma}(z_1, z_2, ..., z_n) = \frac{\partial}{\partial z_{k,i}} D_i^{\Sigma}(z_1, z_2, ..., z_n)$$

Agents learn robust teaming behavior using either evaluation method due to their alignment. As an agent-specific evaluation, D^{Σ} provides a much more sensitive signal to optimize. Similar to D, D^{Σ} largely removes the impact of the other agents through the evaluation of counterfactual states, allowing an agent to learn its contribution to multiple teams. Each agent learning this behavior results in a group of agents that can successfully cooperate in a variety of teams.

4 ROVER DOMAIN

We evaluate our method on the Rover Domain, a multiagent exploration task [4]. In the basic formulation of this problem, a group of rovers must explore a two-dimensional location and successfully observe various Points Of Interest (POI). These POIs represent complex objects requiring multiple simultaneous observations from the rovers. The number of rovers required to successfully view a POI is referred to as the coupling requirement. Each POI is also assigned a value proportional to its relative importance. The rovers have a limited range in which they can provide high-quality observation of the POIs. To complete this task, the rovers must form groups of size equal to the coupling requirement near the highest valued POIs at the end of the episode.

Each rover is equipped with two types of sensors to view its surroundings: rover sensors and POI sensors. Each sensor can view a 90-degree cone area from the robot. The rovers are equipped with four of each type of sensor to fully view their surroundings. The rover sensor is modeled by equation 4. Here S represents the sensor value for rover i. $\delta(i, i')$ represents the Euclidean distance between the rover and any other rover, i', within the viewing angle. This sensor is designed to give a stronger signal as a rover moves closer to other rovers.

$$S_{robot} = \sum_{i'} \frac{1}{\delta(i, i')} \tag{4}$$

The sensors to detect POIs are similarly modelled using equation 5. Here the distance is calculated between the rover and a POI, j. The value, V_j of the POI is also included in the sensor to allow the rover to better differentiate higher-valued POIs.

$$S_{PoI} = \sum_{j} \frac{V_j}{\delta(i, j)} \tag{5}$$

At each time step, each rover uses this local sensor information to determine an action to take. Each action consists of a continuous linear and angular displacement. In a given action a robot can move linearly between -1 and +1 units in the direction of the rover's heading and change their heading by between -90 and +90 degrees. The rovers do not need to take a separate action to observe a POI. Instead, they need to be within the observation radius of the POI.

Evaluation of the rovers occurs at the end of the episode, after a fixed number of time steps. This global evaluation function is defined by:

$$G = \sum_{j} \frac{V_j I(j)}{max(1, \delta(i'', j))} \tag{6}$$

In this equation the indicator function I returns a value of 1.0 if enough rovers are within the observation radius to meet or exceed the coupling requirement of the POI j, otherwise a 0.0 is returned. The value of this observed POI is scaled by the distance of the closest rover i'' to promote close observation of the POI. Overall, this function provides an evaluation that represents the combined values of the successfully observed POIs. To maximize this value, the rovers should group around the highest-valued POIS.

5 EXPERIMENTS

To show the effectiveness of D^{Σ} and G^{Σ} in training agents with robust teamwork skills, we conduct three sets of experiments in the Rover Domain:

(1) **Training Few Teams:** The first experiment trains agents to cooperate in a small number of teams. We train five agents across teams of four and seven agents across teams of six to demonstrate the effectiveness of training agents across a low number of teams. This experiment shows the preliminary quality of D^{Σ} and G^{Σ} as learning signals to promote cooperation in multiple teams. (Results in Section 6.1.)
(2) **Training Many Teams:** The second experiment increases the number of training teams. We train seven agents across teams of four and nine agents across teams of six. Maintaining a constant team size, but increasing the number of agents results in a higher number of teams to train with. The purpose of this second experiment is to show how well these shaped rewards scale to many teams, a more difficult task to learn. (Results in Section 6.2.)
(3) **Training On a Subset of Teams:** The third set of experiments investigates the impact of the number of teams the agents use to learn their policies by varying the subset of teams selected. We again train nine agents with teams of six, but we vary the number of total training teams. Each set of agents is evaluated using all possible teams. This experiment shows how well the agents perform in teams they are not trained with and whether they need to train with every team. (Results in Section 6.3.)

For each experiment, we compare our results to single-team learning signals applied to a multi-team task. We train these baseline agents by randomly selecting a team each episode and provide the agents with either G or D to learn from. This provides a direct comparison between single-team and our proposed multi-team learning signals. The agents are trained using the same evolutionary algorithms as D^{Σ} and G^{Σ}. The only difference between learning agents is the fitness shaping function.

Note that we do not compare to any ad hoc teaming methods as they solve a fundamentally different problem. In this work, agents must simultaneously learn to work together to achieve a task. In ad

hoc teaming, a single agent must learn to cooperate with various pre-existing teams. The ad hoc teaming agent must learn its role within a team, whereas in this work, each agent must learn to cooperatively complete a task in multiple teams.

5.1 Experimental Parameters

The agent policies were represented as neural networks which mapped local state information to an action. Each network contained one hidden layer of size 20 with tanh used for each activation function. The weights of these networks were initialized using Xavier initialization [17].

All agents were cooperatively evolved using CCEA. Each agent was given a population size of 32 and selected using binary tournament selection. New agents were created by copying one of each of the selected agents and then mutating the copies. The mutation operator consisted of uniformly selecting weights from the network with a probability of 10%. These selected weights were then multiplied by a value sampled from $\mathcal{N}(1.0, 0.01)$. Evolution occurred over the course of 4000 generations.

In the Rover Domain simulation, the rovers began each episode in the middle of a 30-by-30 unit area surrounded by six POIs. Each POI had a coupling requirement of two, with values and locations described in table 1. These POIs also had an observation radius of 5.0 units.

Table 1: POI Settings

POI	1	2	3	4	5	6
Value	0.6	0.2	0.8	0.1	0.3	1.0
X	0	0	0	30	30	30
Y	30	15	0	0	15	30

6 RESULTS

All results reported in this section are averaged over $n = 16$ statistical trials. Instead of giving a specific confidence interval, we provide the standard error $(\sigma/\sqrt{n})$ with shaded regions. Performance in each test is evaluated using G^Σ because it describes the cumulative performances of the agents across all teams. While this metric is one of the training signals, it also provides the best measurement of global performance across multiple teams.

6.1 Training Few Teams

In this first set of experiments, we train agents to work with a low number of teams using G^Σ and D^Σ. These tests assess the viability of these shaped fitnesses in training multiple agents to complete a task in multiple teams. In the first experiment, we train a pool of five agents in teams of size four, giving $\binom{5}{4} = 5$ teams to learn with. The results of this test can be viewed in Figure 2. In this figure, it is clear that G is not an informative signal to learn from. Poor learning is expected due to the low sensitivity and general lack of information provided by the signal. To view any of the POIs, two agents need to be within the observation radius of the same POI at the end of the episode, a fairly rare event. The global nature of this feedback further impedes learning as each agent is given the same fitness, regardless of contribution. Unsurprisingly, D can train a better set of agents. This improvement can largely be attributed to the higher sensitivity in the feedback provided.

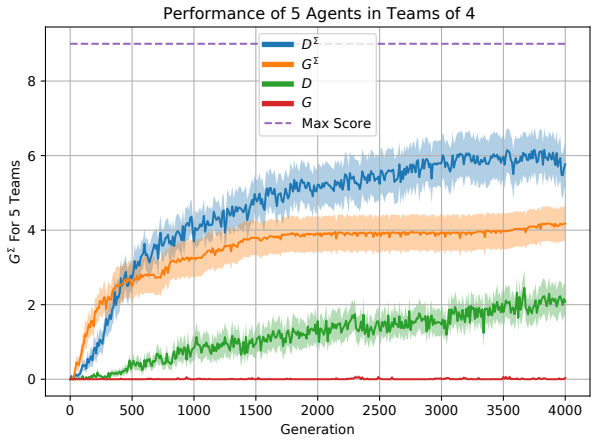

Figure 2: The performance of D^Σ, G^Σ, D, and G for 5 agents across teams of 4, resulting in 5 teams.

Moderate performance is shown by the agents trained from G^Σ. Evaluating across multiple teams provides agents with a much more dense signal in comparison to G and D, which is easier to learn from. However, the lack of agent-specific feedback inhibits the learning of higher quality teaming policies.

Lastly, the highest-scoring fitness was D^Σ. This follows expectation as it provides a more informative signal, similar to G^Σ, but is agent-specific allowing for increased sensitivity. Using D^Σ agents were able to learn optimal teamwork behaviors in some trials, reaching the maximum score in all teams. Figure 3 shows the learned behaviors from D^Σ for all five teams. Here, there are enough agents to form two teams of size two to observe two of the POIs. The optimal behavior is to send two agents to each of the two highest valued POIs in each team, which was learned by D^Σ. Agents were able to learn their roles within arbitrary teams to effectively cooperate with all teams. Conversely, agents learning from G^Σ were never able to learn this optimal behavior. These agents learned to observe two of the lower-valued POIs, which is an artifact of the credit assignment problem. We train agents with other team sizes to verify that these performance trends hold for team sizes other than four agents. These tests include a pool of seven agents and a team size of six with results presented in Figure 4. The figure shows that for few teams ($\binom{7}{6} = 7$ teams) but different team sizes, the overall trends hold. Again, D^Σ trained agents with optimal team behaviors during some of the trials, while G^Σ does not. For small team sizes, D^Σ seems to provide the best signal to optimize, with G^Σ being a close second. Single-team learning methods do not provide nearly the same level of performance.

6.2 Training Many Teams

The second set of experiments keeps the sizes of the teams the same but increases the number of agents, which leads to a larger number of teams to cooperate with. The goal of this set of tests is to determine how well G^Σ and D^Σ trains agents to cooperate with

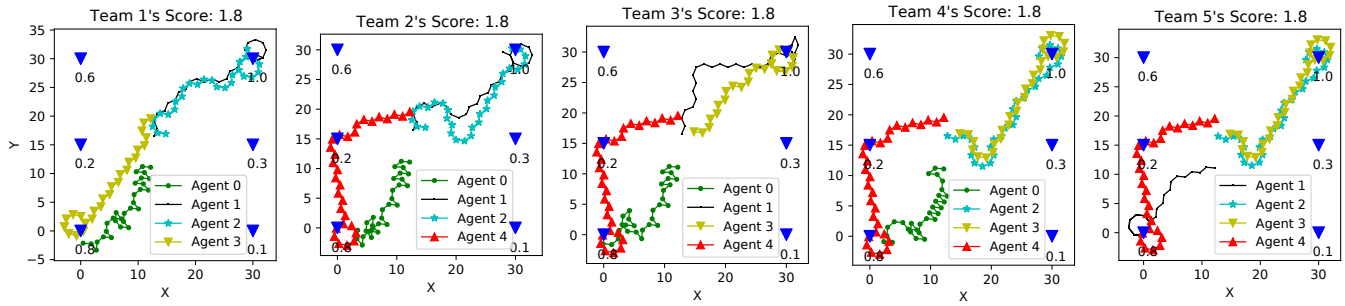

Figure 3: The paths of the agents are shown moving toward and observing the POIs, denoted by blue triangles. Each agent learns its role within multiple teams to achieve the maximum score across all teams. For example, Agent 3 generally learns to observe the top right POI. If two agents are already on the way to this POI, it learns to go to the next best POI in the bottom left.

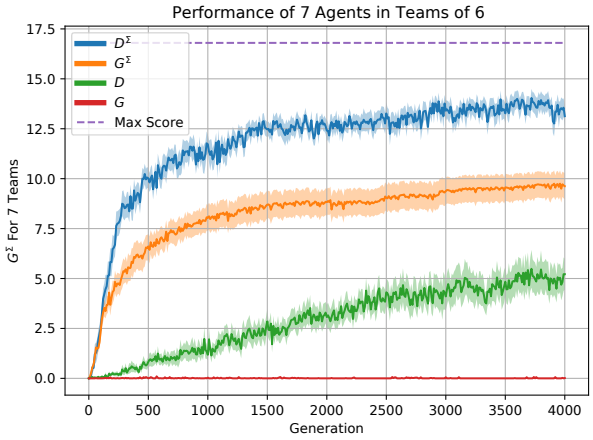

Figure 4: The performance of D^Σ, G^Σ, D, and G for 7 agents across teams of 6, resulting in 7 teams. D^Σ and G^Σ show better performance when training on a small number of teams.

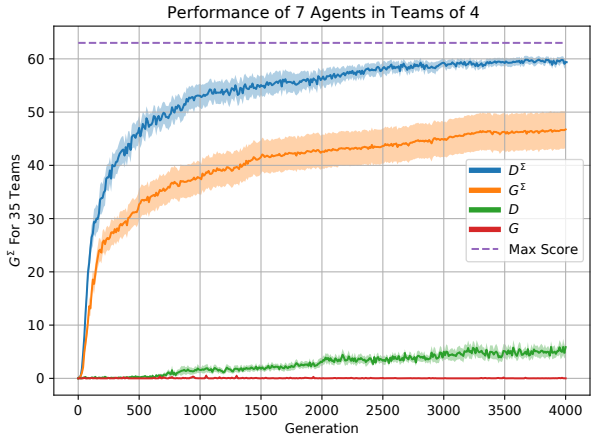

Figure 5: The performance of D^Σ, G^Σ, D, and G for 7 agents across teams of 4, resulting in 35 teams.

many teams. One would expect that learning in a larger number of teams would increase the difficulty of the task as there are more interactions to learn. In the first part of the test, we train with seven agents in teams of four for a total of $\binom{7}{4} = 35$ teams. Results of this test are shown in Figure 5. Again, we compare G^Σ and D^Σ to standard single team evaluation methods G and D. The difference between our shaped fitnesses and the multiagent fitnesses appears to grow with larger numbers of teams. Again, G fails to learn anything, due to the sparsity of the fitness received. D shows reduced relative performance in comparison to the smaller team methods. This can be best explained by the noise introduced by the higher number of teams. Each team provides a potentially different difference reward which can be difficult to optimize when the team structure is constantly changing.

Both G^Σ and D^Σ perform better relative to the fewer team case. Increasing the number of teams increases the density of the evaluation for both cases, thus providing a higher quality learning signal. Another contributing factor could be the overlap in an agent's contribution from one team to another. If a single agent provides the same contribution to multiple teams, increasing the number of teams will not change this contribution. Overall, it seems that D^Σ is still a sufficient signal to learn from in the presence of an increased number of teams.

Similar to previous experiments, D^Σ produces multiple examples of optimal agent policies where each team reaches the maximum performance. Again, G^Σ fails to produce similar quality agents.

We also test with a larger team size to verify these trends for team sizes other than four agents. This involves training nine agents across teams of six to produce $\binom{9}{6} = 84$ total teams. The results of this experiment are shown in Figure 6. The widening gap between the shaped fitnesses and the multiagent fitnesses is still apparent in Figure 6. Again, G and D fail to learn meaningful agent policies. D^Σ can learn high-quality policies in the presence of an even higher team count, while G^Σ falls behind. This would indicate that G^Σ and D^Σ do scale well with the number of teams.

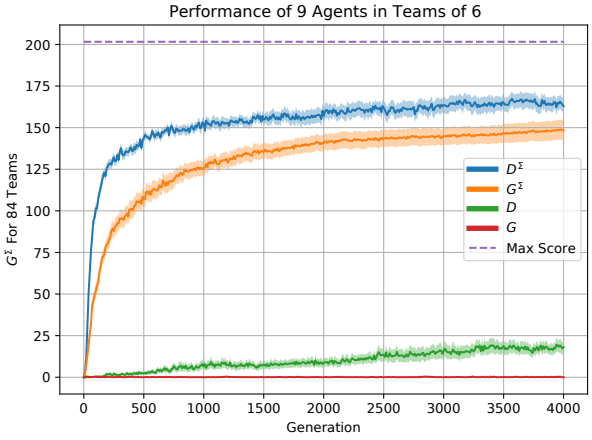

Figure 6: Performance of D^Σ, G^Σ, D, and G for 9 agents across teams of 6, resulting in 84 teams. D^Σ and G^Σ show significantly better performance when training on a large number of teams.

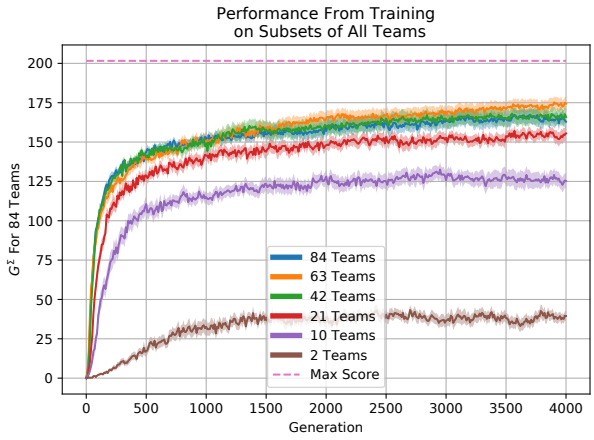

Figure 7: We train using a subset of the full set of teams and show the performance on the full set of teams. We vary the size of the subset from 2 to the full set of 84. This test used 9 agents and a team size of 6. Note that training on 21 teams achieves 90% the performance of training with the full set but requiring 25% the number of team evaluations.

From the two sets of tests, a few trends are clear. D^Σ consistently provides the best signal to learn from, with G^Σ falling a bit behind. D and G are unable to evolve these agents to effectively cooperate with multiple teams, especially in settings with many teams.

6.3 Training On a Subset of Teams

In the presence of training agents across multiple teams, overlap between an agent's contribution in one team and another likely exists. As a result, evaluating these agents in all combinations of teams may be excessive. The purpose of the last experiment is to determine how many teams need to be evaluated to achieve most or full performance in comparison to training with all teams.

In the final experiment, we vary the number of teams the agents trained with. Each part of the experiment uses the same number of agents in the same team sizes, but we vary the number of training teams to determine their effect on learned performance. We train nine agents across a team size of six for a maximum number of 84 teams. For each part of the experiment, we randomly sample a set of these teams without replacement. Agents evolve based on their evaluation of D^Σ across this subset of teams each generation. The policies are periodically evaluated using the G^Σ of all 84 teams to measure general teaming performance across all teams. We record results for the cases with 2, 10, 21, 42, 63, and 84 teams. The training curves are included in Figure 7. When compared to training with the full set of 84 teams, training on 63 and 42 teams shows nearly identical performance. This result is surprising as one would expect these agents to perform poorly in teams they were not trained with, yet this is not the case. Training on half of the teams produces agents that perform as well as the agents that were trained on all teams. This is most likely due to the overlap in an agent's contribution across multiple teams. A single agent is unlikely to have a different role in each team, so it does not need to interact with every team. Each agent only needs to learn with teams requiring unique behaviors.

A surprising result is the high performance of the agents trained on the sub-sample of 21 teams. These teams represent 25% of all teams, yet the agents learn 90% of the performance of the agents learning on the full set. In this test, an agent needs to move to one of the three highest valued POIs along with another agent for successful observation of the POI. As a result, each agent may need roughly three unique behaviors to fulfill the needs of a team. Each agent does not need to interact with 84 teams to learn such a low number of behaviors. In the 21 teams randomly chosen, there is nearly enough variation to train most of these unique behaviors for each agent, leading to high performance.

A large decrease in performance is first seen with 10 teams and continues with 2 teams. The agents do not receive experiences with enough teams to learn quality teamwork behavior. These agents ultimately struggle to cooperate with varying teams.

The main conclusion of this test is that training agents in all teams may be excessive. Training on a sub-sample of these teams can be sufficient due to the overlap in an agent's contribution across multiple teams. The optimal number of teams to sample is likely to be domain-dependent. Domains requiring agents to learn many unique skills from one team to another are likely to require many training teams, while teams of agents that complete similar tasks in parallel are likely to need fewer teams.

6.4 The Computational Cost of G^Σ and D^Σ

It is important to acknowledge increased computational cost in calculating D^Σ over G^Σ. Calculating G requires evaluating a single team, or $O(1)$ team evaluations while evaluating G^Σ requires $O(N)$ team evaluations, where N is the number of teams. D^Σ imposes an even higher cost, with $O(NM)$ evaluations, where M is the size of a single team. However, it should be noted that in some applications,

a closed-form difference evaluation for each agent exists, reducing the complexity of D^Σ to be equal to G^Σ.

Previous experiments directly compared D^Σ to G^Σ across generations, which may not be entirely fair due to each D^Σ evaluation receiving more information and computation. It could be the case that G^Σ and D^Σ perform equivalently if given the same amount of computation. To test this hypothesis, we recycle the results from a previous experiment and show performance as a function of the number of team evaluations, G. This allows us to directly compare the two evaluations considering that D^Σ requires significantly more team evaluations. The scaled results are shown in Figure 8. In Figure 8, G^Σ shows a higher initial increase, but converges to a lower value than D^Σ. The initial increase of G^Σ is due to the density of the signal itself while being less computationally expensive than D^Σ. However, D^Σ converges to a higher value due to the increased sensitivity of the local fitness providing a more informative signal to optimize.

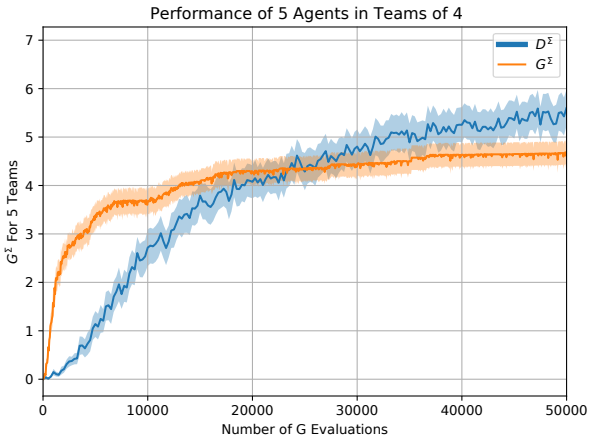

Figure 8: We show the performance of G^Σ and D^Σ scaled for equal number of team evaluations, G. D^Σ shows better long-term performance due to its agent-specific nature. This was taken from the 5 agent, team size of 4 experiment.

This trend of D^Σ providing better long-term performance continues in the experiments with a larger number of training teams. Figure 9 shows the performance of G^Σ and D^Σ as a function of the number of team evaluations, G, for the test with seven agents in teams of four. Again, G^Σ shows initial promise, but plateaus due to the low sensitivity of the signal. D^Σ shows a slower increase in performance but still shows the highest solution quality. These results generally show that despite the increased computational requirement of D^Σ, it most likely provides the best performance for the same computation in the long term. Even though evaluating D^Σ requires more calculations than G^Σ, the efficiency of D^Σ is likely much better. Overall, D^Σ produces the highest quality learning signal unless the computational cost of G is very high. If this is the case, then G^Σ provides a more computationally efficient signal in the short term.

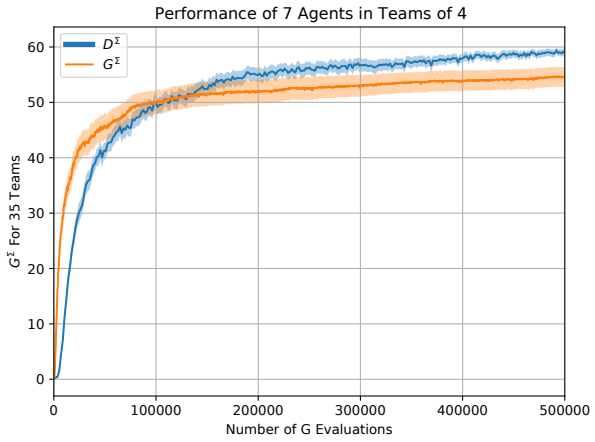

Figure 9: We show the performance of G^Σ and D^Σ scaled for equal number of team evaluations, G. This was taken from the 7 agent, team size of 6 experiment.

7 CONCLUSION

In this work, we provide a shaped fitness function that leverages counterfactual states to successfully coevolve multiple agents to cooperate in multiple teams. We showed the effectiveness of this method in the rover domain across varying numbers of agents, teams, and team sizes. These agents were even able to learn optimal control strategies, showing maximum team performance across all teams.

This shaped fitness performed well for a variety of reasons, with the first reason being the sensitivity of the function. As a local fitness evaluation, agents were able to learn individual contributions to a variety of teams. Another contributing factor to the success of D^Σ is the density of the evaluation. Evaluating multiple teams at once allows agents to learn from their collective experiences across the teams, improving overall learning. Lastly, we show that this function is aligned with the global multi-team score, guaranteeing agents to learn quality teamwork across multiple teams.

Future work in this area could focus on improving the sample efficiency of D^Σ. Fitness modeling could train a computationally efficient model of this function through interaction with the environment [32]. Specific fitness modeling techniques such as fitness critics and difference evaluation approximation have similarly modeled complex evaluation functions in multiagent domains to provide agent-specific feedback [11, 31]. These methods could be extended to learn D^Σ from individual team performances, reducing the overall number of team evaluations needed to learn quality teamwork across multiple teams.

ACKNOWLEDGMENTS

This work was partially supported by the National Science Foundation under grant No. IIS-1815886 and by the Office of Naval Research under grant No. UWSC12017-BPO49408

REFERENCES

[1] Arvin Agah and Kazuo Tanie. 1997. Robots Playing to Win: Evolutionary Soccer Strategies. In *Proceedings of International Conference on Robotics and Automation*, Vol. 1. IEEE, 632–637.

[2] A. Agogino, C. Holmes Parker, and K. Tumer. 2012. Evolving Large Scale UAV Communication Systems. In *Proceedings of the Genetic and Evolutionary Computation Conference*. Philadelphia, PA.

[3] Adrian Agogino and Kagan Tumer. 2005. Reinforcement Learning in Large Multi-Agent Systems. In *Proc. of AAMAS-05 Workshop on Coordination of Large Scale Multiagent Systems*. Citeseer.

[4] Adrian K Agogino and Kagan Tumer. 2008. Analyzing and Visualizing Multiagent Rewards in Dynamic and Stochastic Domains. *Autonomous Agents and Multi-Agent Systems* 17, 2 (2008), 320–338.

[5] Samuel Barrett and Peter Stone. 2015. Cooperating with Unknown Teammates in Complex Domains: A Robot Soccer Case Study of Ad Hoc Teamwork.. In *AAAI*, Vol. 15. Citeseer, 2010–2016.

[6] Samuel Barrett, Peter Stone, Sarit Kraus, and Avi Rosenfeld. 2012. Learning Teammate Models For Ad Hoc Teamwork. In *AAMAS Adaptive Learning Agents (ALA) Workshop*. 57–63.

[7] Yu-Han Chang, Tracey Ho, and Leslie P Kaelbling. 2004. All Learning is Local: Multi-agent Learning in Global Reward Games. In *Advances in neural information processing systems*. 807–814.

[8] Shuo Chen, Ewa Andrejczuk, Zhiguang Cao, and Jie Zhang. 2020. AATEAM: Achieving the Ad Hoc Teamwork by Employing the Attention Mechanism. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 7095–7102.

[9] Shuo Chen, Ewa Andrejczuk, Athirai Aravazhi Irissappane, and Jie Zhang. 2019. ATSIS: Achieving the Ad Hoc Teamwork by Sub-task Inference and Selection. (2019).

[10] André LV Coelho and Daniel Weingaertner. 2001. Evolving Coordination Strategies in Simulated Robot Soccer. In *Proceedings of the fifth international conference on Autonomous agents*. 147–148.

[11] Mitchell Colby, Theodore Duchow-Pressley, Jen Jen Chung, and Kagan Tumer. 2016. Local approximation of difference evaluation functions. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. 521–529.

[12] Mitchell K Colby, Sepideh Kharaghani, Chris HolmesParker, and Kagan Tumer. 2015. Counterfactual Exploration for Improving Multiagent Learning.. In *AAMAS*. 171–179.

[13] Mitchell K Colby and Kagan Tumer. 2012. Shaping Fitness Functions for Coevolving Cooperative Multiagent Systems.. In *AAMAS*, Vol. 1. 425–432.

[14] Carlos M Fonseca and Peter J Fleming. 1995. An Overview of Evolutionary Algorithms in Multiobjective optimization. *Evolutionary computation* 3, 1 (1995), 1–16.

[15] Ping-an Gao, Zi-xing Cai, and Ling-li Yu. 2009. Evolutionary Computation Approach to Decentralized Multi-Robot Task Allocation. In *2009 Fifth International Conference on Natural Computation*, Vol. 5. IEEE, 415–419.

[16] Katie Long Genter, Noa Agmon, and Peter Stone. 2011. Role-Based Ad Hoc Teamwork.. In *Plan, Activity, and Intent Recognition*. Citeseer, 1782–1783.

[17] Xavier Glorot and Yoshua Bengio. 2010. Understanding The Difficulty of Training Deep Feedforward Neural Networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 9)*, Yee Whye Teh and Mike Titterington (Eds.). PMLR, Chia Laguna Resort, Sardinia, Italy, 249–256. https://proceedings.mlr.press/v9/glorot10a.html

[18] Jorge Gomes, Miguel Duarte, Pedro Mariano, and Anders Lyhne Christensen. 2016. Cooperative Coevolution of Control for a Real Multirobot System. In *International Conference on Parallel Problem Solving from Nature*. Springer, 591–601.

[19] Atil Iscen, Ken Caluwaerts, Jonathan Bruce, Adrian Agogino, Vytas SunSpiral, and Kagan Tumer. 2015. Learning Tensegrity Locomotion Using Open-loop Control Signals and Coevolutionary Algorithms. *Artificial Life* (2015). https://doi.org/10.1162/artl_a_00163

[20] Edwin D de Jong. 2003. Representation Development from Pareto-Coevolution. In *Genetic and Evolutionary Computation Conference*. Springer, 262–273.

[21] Xin Ma, Qin Zhang, and Yibin Li. 2007. Genetic Algorithm-Based Multi-Robot Cooperative Exploration. In *2007 IEEE International Conference on Control and Automation*. IEEE, 1018–1023.

[22] Andrew Y Ng, Daishi Harada, and Stuart Russell. 1999. Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In *Icml*, Vol. 99. 278–287.

[23] MohammadJavad NoroozOliaee, Bechir Hamdaoui, and Kagan Tumer. 2013. Efficient Objective Functions for Coordinated Learning in Large-Scale Distributed OSA Systems. *IEEE Transactions on Mobile Computing* (2013). https://doi.org/10.1109/tmc.2012.67

[24] Esben H Østergaard, Henrik H Lund, and Reality Gap. 2002. Co-Evolving Robot Soccer Behavior. In *From Animals to Animats 7: Proceedings of the Seventh International Conference on Simulation of Adaptive Behavior*, Vol. 7. MIT Press, 351.

[25] Liviu Panait, R Paul Wiegand, and Sean Luke. 2004. A Sensitivity Analysis of a Cooperative Coevolutionary Algorithm Biased for Optimization. In *Genetic and Evolutionary Computation Conference*. Springer, 573–584.

[26] Mitchell A Potter and Kenneth A De Jong. 1994. A Cooperative Coevolutionary Approach to Function Optimization. In *International Conference on Parallel Problem Solving from Nature*. Springer, 249–257.

[27] Mitchell A Potter and Kenneth A De Jong. 2000. Cooperative Coevolution: An Architecture For Evolving Coadapted Subcomponents. *Evolutionary computation* 8, 1 (2000), 1–29.

[28] Muhammad A Rahman, Niklas Hopner, Filippos Christianos, and Stefano V Albrecht. 2021. Towards Open Ad Hoc Teamwork Using Graph-based Policy Learning. In *International Conference on Machine Learning*. PMLR, 8776–8786.

[29] 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, 4424–4429.

[30] Manish Chandra Reddy Ravula. 2019. *Ad-hoc Teamwork with Behavior-switching Agents*. Ph. D. Dissertation.

[31] Golden Rockefeller, Patrick Mannion, and Kagan Tumer. 2019. Fitness Critics for Multiagent Learning. In *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 222–224.

[32] Michael Schmidt and Hod Lipson. 2005. Coevolution of Fitness Maximizers and Fitness Predictors. *GECCO Late Breaking Paper* (2005).

[33] Reid Simmons, David Apfelbaum, Wolfram Burgard, Dieter Fox, Mark Moors, Sebastian Thrun, and Håkan Younes. 2000. Coordination for Multi-Robot Exploration and Mapping. In *Aaai/Iaai*. 852–858.

[34] Peter Stone, Gal A Kaminka, Sarit Kraus, Jeffrey S Rosenschein, et al. 2010. Ad Hoc Autonomous Agent Teams: Collaboration without Pre-Coordination.

[35] Qingyu Tan, Zhenkun Wang, Yew-Soon Ong, and Kin Huat Low. 2019. Evolutionary Optimization-Based Mission Planning for UAS Traffic Management (UTM). In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 952–958.

[36] K. Tumer and A. K. Agogino. 2007. Coordinating Multi-Rover Systems: Evaluation Functions for Dynamic and Noisy Environments. In *Evolutionary Computation in Dynamic and Uncertain Environments*, S. Yang (Ed.). Springer, 371–388.

[37] Kagan Tumer, Zachary T. Welch, and Adrian Agogino. 2008. Aligning Social Welfare and Agent Preferences to Alleviate Traffic Congestion. *AAMAS* (2008). https://doi.org/10.1145/1402298.1402315

[38] Karl Tuyls and Gerhard Weiss. 2012. Multiagent Learning: Basics, Challenges, and Prospects. *Ai Magazine* 33, 3 (2012), 41–41.

[39] Eiji Uchibe, Masateru Nakamura, and Minoru Asada. 1999. Cooperative and Competitive Behavior Acquisition for Mobile Robots Through Co-evolution. In *Proc. of the Genetic and Evolutionary Computation Conference*. Citeseer, 1406–1413.

[40] Csaba Virágh, Máté Nagy, Carlos Gershenson, and Gábor Vásárhelyi. 2016. Self-Organized UAV Traffic in Realistic Environments. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 1645–1652.