# Structural Credit Assignment for Multiagent Leader-Based Learning

Everardo Gonzalez
Oregon State University
Corvallis, United States
gonzaeve@oregonstate.edu

Andrew Festa
Oregon State University
Corvallis, United States
festaa@oregonstate.edu

Nathan Hewitt
Oregon State University
Corvallis, United States
hewittna@oregonstate.edu

Kagan Tumer
Oregon State University
Corvallis, United States
ktumer@oregonstate.edu

## ABSTRACT

Coevolutionary agents offer new solutions to domains such as air traffic control, autonomous vehicle management, and deep ocean exploration where multiple decision makers need to coordinate to satisfy a system objective. Agents fail to coevolve useful policies as the number of agents in the system increases due to the challenge imposed by each agent learning from a system wide feedback signal. Structural credit assignment techniques provide some relief, as they distill system feedback into cleaner, individualized learning signals for each agent. However, these techniques struggle to learn tightly coupled system objectives, where several agents must discover a sequence of joint-actions together in order to receive a positive feedback signal. In this paper, we introduce structural credit assignment for team "leaders", learning agents which are accompanied by "follower" agents with pre-programmed simple behaviors. These multiagent leader-based teams more easily learn joint-policies that solve tightly coupled system objectives. We demonstrate that this combination of followers and modified credit assignment results in much faster and reliable learning than comparable credit assignment techniques without followers for tightly coupled system objectives.

## KEYWORDS

Multiagent Learning, Reward Shaping, Credit Assignment

**ACM Reference Format:**
Everardo Gonzalez, Andrew Festa, Nathan Hewitt, and Kagan Tumer. 2023. Structural Credit Assignment for Multiagent Leader-Based Learning. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 7 pages.

## 1 INTRODUCTION

Multiagent systems offer the potential for designing for interactions among various entities to solve complex problems, such as collaborative excavation of remote dig sites. The ability for autonomous agents to perform these tasks would allow for operation in dangerous and hostile environments, and solving larger problems should only require using more agents to complete a given task.

However, the difficulty lies in the ability for agents to scale on cooperative tasks. As the number of learning agents increases in the system, the learning signal directing each individual agent becomes weaker, and the likelihood of the agents discovering an effective joint policy quickly diminishes.

At the core, the issue is one of providing enough information to agents to correctly learn how to effectively cooperate. Carefully designing reward structures can help with this by making sure that agents receive adequate information, such as how difference rewards approaches the issue. By focusing designing the reward to tell an agent how they contributed to the system, rather than how the system performed, an agent can more clearly understand how they might better act.

These techniques require computing such a reward for every agent in the system leading to a dramatic increase in computation requirement for coordinating joint actions between a large number of agents. This assumes that the agents are even able to learn from the shaped reward, which is itself not always valid. Difference rewards, for example, fails on tightly coupled tasks, where multiple agents must act in tandem to achieve a system reward.

Multifitness learning addresses some of these issues, namely with respect to an exponential joint-action space, by learning not individual policies, but by instead selecting from a set of already known behaviors at any given point in time. This partially sidesteps the problem by reducing the action space of all individual agents, thereby making the learning problem easier for the agents. The problem with such an approach is in requiring these behaviors to be known beforehand. And then, even if they are able to be known, each agent is still learning individually, meaning that multifitness learning only alleviates the issue of an individual agent learning in a complex environment rather than directly addressing the issue of a large number of simultaneous learners.

In this paper, we blend ideas from both of these approaches to introduce structural credit assignment for team leaders where a portion of the agents are "followers" with preset policies and the "leaders" must learn to complete system tasks as well as influence followers to help complete system tasks. Assigning the followers with a definite and simple policy allows for injecting knowledge about the structure of the problem. This also shifts the learning problem from many agents learning to cooperate effectively in a highly non-stationary environment to a less non-stationary problem of leaders guiding followers to accomplish the requisite task.

*Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.* 

The contributions of this work are to:

- Provide a mechanism for reducing the complexity of interactions between agents
- Derive a method for calculating team contribution in a leader-follower paradigm

Reframing the task in such a way, coupled with the shaped knowledge regarding the reward and problem, allows the agents to learn a more optimal solution faster than agents trained only with shaped rewards or those learning to select between policies. Additionally, this approach is able to continue to solve problems requiring coordination between many more agents beyond those capable of the other methods described.

In the following sections, we provide background knowledge required to understand the approach in section 2, followed by a more in-depth explanation of the approach and experimental setup in section 4. Section 5 gives the results and analysis of the experiments, and section 6 provides concluding remarks along with steps for future work.

## 2 BACKGROUND

### 2.1 Swarm Shepherding

The swarm shepherding problem is one where one class of agents referred to as "sheepdogs" must learn to guide another class of agents referred to as "sheep" [9]. Existing approaches generally formulate this problem as a single-agent learning problem, where the sheepdogs are separate entities, but each sheepdog is learning the same policy [5, 8, 16].

Hussein et al and Gee et al both take a curriculum based approach to learning policies for the sheepdogs, indicating that the shepherding problem oftentimes suffers from uniformative rewards that must be addressed by clever learning approaches [5, 8]. Nguyen et al takes a multiagent learning approach to learning policies for sheepdogs, where several policies are learned according to where a sheepdog is relative to the herd [11]. While this approach begins to address the challenge of learning individual policies for individual sheepdogs, it does not fully commit to experimenting with every sheepdog learning its own policy.

Tuzel et al frames the swarm shepherding problem as a problem of learning based leadership [16], which is similar to the problem we address in this paper. Tuzel successfully used an evolutionary approach to learn a policy for a class of "leader" agents that guided a class of "follower" agents to a goal. We scale up this approach to include individualized policies for leaders, and to solve more complex objectives with less informative feedback.

### 2.2 Multiagent Learning

Multiagent learning problems typically involve multiple agents, each with their own observation and action spaces, learning their own individual policies in a shared environment as part of a multiagent team [15]. One of the primary challenges in multiagent learning is distilling a system wide feedback signal into useful feedback for individual agents. A team may perform well overall, but this does not mean that each agent performed well. Reward shaping techniques such as structural credit assignment begin to address this problem [1, 3, 10, 14]. Difference evaluations in particular compare the system's performance against the performance without a particular agent to determine the impact that agent had on the system [1].

While these techniques help to distill system feedback into individualized agent feedback, they generally do not scale well with tightly coupled team objectives, where several agents must stumble upon the correct sequence of joint-actions in order to satisfy that objective. This problem is further exacerbated when feedback is sparse and uninformative. Rahmattalabi et al extend difference evaluations through D++ to offer "stepping stone" feedback for an agent if it is partially satisfying a tight coupling requirement [12]. Unfortunately, calculating these "stepping stones" can be quite computationally expensive compared to calculating the standard difference evaluation, especially as we increase the coupling requirement.

### 2.3 Cooperative Co-evolution

Evolutionary algorithms are based on the ideas of natural selection and that good performing individuals will tend to continue on and produce better solutions over generations [17]. In the single agent case, a population of agents is evaluated on a task based on some fitness function. Their relative fitnesses are used to select individuals to continue on to a future iteration. At some point in this process, the individuals are mutated in some fashion so as to produce a slightly different solution. The process then continues for a future generation, and the expectation is that the overall fitness of the population of solutions will improve from generation to generation.

When applied to a multiagent setting, specifically for cooperative tasks, the fitnesses are no longer based solely on the performance of the individual, but on the team of individuals [2, 6, 18]. Evaluating an individual is no longer based solely on its own fitness, but on the fitness of the team it is a part of during evaluation. Care must be taken so as to not penalize an agent for its teammates while also capturing how it performed as part of the team. Further, the evolutionary algorithm itself must be modified to account for learning with multiple agents, rather than just one. Instead of tracking a single population of policies for an agent, a cooperate co-evolutionary algorithm tracks one population for every agent on the multiagent team.

### 2.4 Neuroevolution

Neuroevolution is one such manner for altering neural networks through an evolutionary algorithm. By viewing the network weights as characterizing the policy performance, the weights can be mutated in order to generate a slightly different solution from an existing solution [4, 7, 13].

This method of mutation allows the new policy to be viewed as a new individual to be added to the population of solutions for a given agent. The mutated policy is added alongside the previous policy, and thus it does not destroy partial solutions between generations.

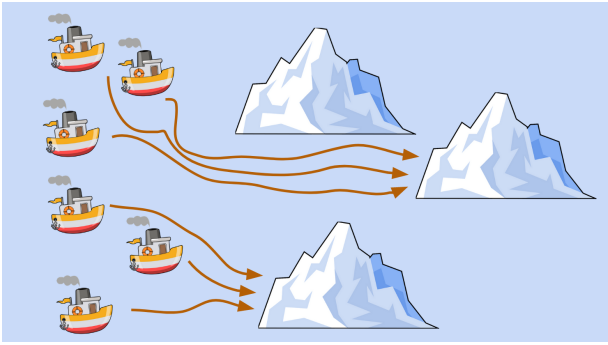


**Figure 1: Arctic Monitoring Environment: A homogenous team of research vessels must coordinate to observe several iceberg POIs with a tight coupling requirement. In this example, 3 vessels are required to simultaneously be within a POI's observation radius for that POI to be observed, and this team successfully observes all POIs.**

# 3 PROBLEM FORMULATION

## 3.1 Arctic Monitoring Environment

We frame our contribution using the Arctic Exploration Environment, shown in Figure 1. A team of homogeneous research vessels must coordinate on a tightly coupled arctic exploration task to observe as many Points of Interest (POIs) as possible. The research vessels start with no prior knowledge of where POIs are, or how many POIs there are to observe. There is no direct communication between research vessels beyond limited sensing of other research vessels' positions. The team must learn to coordinate to maximize a team objective function based on observing as many POIs as possible.

The primary challenge of this environment is that several research vessels must observe a single POI simultaneously for that POI to count as "observed" by the team. The number of research vessels required to observe a POI is referred to as the coupling requirement. As this coupling requirement increases, the probability of the team randomly stumbling upon a sequence of joint-actions to observe even a single POI becomes vanishingly small.

The research vessels and POIs are situated on a continuous 2D plane. Each vessel has a position, velocity, and heading which a vessel can influence through its actions. Research vessels move by accelerating forward or backward, and turning left or right. We impose kinematic constraints on the vessels such as a maximum linear velocity, linear acceleration, and angular velocity. Each POI has a fixed position in the plane. The initial state of research vessels and POIs depends on the particular experiment. Research vessels positions, velocities, headings, and POI positions along with the number of research vessels and POIs constitute the initial state.

The observation space for a research vessel is split into 4 quadrants. The quadrants are relative to the research vessel's heading. A research vessel has 2 sensors in each quadrant: 1 for sensing research vessels density in that quadrant, and 1 for sensing POI density. Density is the average inverse distance to all entities of a class within a quadrant. For example, POI density in a research vessel's front right quadrant is the average inverse distance to all POIs in that quadrant. This makes it so the POI density sensor increases if there are many POIs in a quadrant, or if POIs are nearby. The same logic applies to sensing other research vessels. With 2 sensors in each quadrant, this makes the observation space 8 dimensional.

The action space of a research vessel is a desired velocity and desired heading relative to the research vessel's current heading. This makes the action space 2 dimensional. The velocity and heading commands run through a proportional controller which accounts for the kinematics of the research vessel to command a linear acceleration and angular velocity.

## 3.2 Team Objective Function

$G_{discrete}$ is a sparse feedback function with discrete "steps" for evaluating how well the team performed. A POI counts as observed if $\geq C$ agents are within the POI observation radius at any time during the episode, where $C$ is the coupling requirement. This gives the team very little information to learn from, making it so the team has to stumble upon an entire sequence of correct joint actions to get a nonzero, positive feedback signal. We formalize this in Equations 1 and 2 for a case where $C = 2$ with additional context supplied by Equations 3 and 4.

$$G_{discrete}(z) = \frac{P_{observed}}{P_{total}} \tag{1}$$

where $z$ is the joint-trajectory of the team, $P_{observed}$ is the number of POIs observed by the team, and $P_{total}$ is the total number of POIs in the environment.

$$P_{observed} = \sum_{i} \max_{t} \left( \sum_{j} \sum_{k} O^1_{i,j,t} O^2_{i,k,t} \right) \tag{2}$$

where $O^1_{i,j,t}$ and $O^2_{i,k,t}$ are boolean values indicating whether the $i$-th and $k$-th agents were within the observation radius of the $i$-th POI at step $t$. That is,

$$O^1_{i,j,t} = \begin{cases} 1, & \text{if } \sigma_{i,j,t} \leq \sigma_0 \text{ and } \sigma_{i,j,t} < \sigma_{i,l}, \forall l \neq j, \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

$$O^2_{i,k,t} = \begin{cases} 1, & \text{if } \sigma_{i,k,t} \leq \sigma_0 \text{ and } \sigma_{i.k,t} < \sigma_{i,l}, \forall l \neq j,k, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

where $\sigma_{i,j,t}$ is the distance from the $i$-th POI to the $j$-th agent at step $t$, $\sigma_{i,k,t}$ is the distance from the $i$-th POi to the $k$-th agent at step $t$, and $\sigma_0$ is the observation radius of the POIs.

# 4 METHOD

## 4.1 Leaders and Followers

We tackle a system-wide multiagent learning problem by first splitting the team into "leaders" and "followers", as shown in Figure 2. Rather than each agent on the team learning a policy individually, the leaders are the only agents which learn on our team. The followers on the team use policies that are preset based on some expert knowledge of a behavior that would generally be useful in tackling the overall system objective.

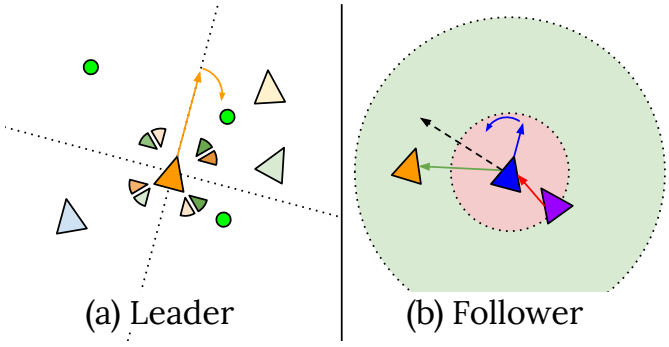



**Figure 2: (a) Leaders sense the density of POIs and teammates in each quadrant. A learned policy commands a new desired velocity and heading. (b) Followers calculate a vector sum of attraction and repulsion forces from teammates in their attraction or repulsion radii. The follower policy commands a desired velocity and heading to align the follower's trajectory with this vector.**

In our Arctic Monitoring Environment, we have the expert knowledge that individual tasks are tightly coupled, and we know the coupling requirement. Thus, we know that a generally useful behavior for an agent would be to stay close to other agents. This makes it easier for the team to satisfy the coupling requirement for a POI when they get close enough to observe the POI.

We formalize this behavior in how a follower calculates its desired velocity and heading. A follower is only influenced by teammates within its observation radius, and the follower does not have the same observation space as a leader. However, a follower does have the same action space of desired velocity and heading. To calculate a desired velocity and heading, a follower decomposes a sum of forces acting on that follower from nearby teammates. Two forces act on a follower: repulsion and attraction. A follower has an attraction radius the same size as its observation radius, and a smaller repulsion radius. If a teammate is within a follower's repulsion radius, then the follower is pushed away from that teammate. If a teammate is instead outside the repulsion radius and within the attraction radius, then the follower is pulled towards that teammate. This makes it so that followers' trajectories are malleable and can be influenced by various teammates at once.

## 4.2 Difference Objective Functions

We experiment with various difference objective functions as learning signals for our leader agents. These difference objectives are designed to better capture a particular agent's contribution to the team's performance better than the team objective function. A difference objective is computed for an agent by calculating the team objective function and subtracting the team objective function with a particular agent removed from the system. We formalize this in Equation 5.

$$D_i = G(z) - G(z_{-i}) \tag{5}$$

where $D_i$ is the difference objective function for agent $i$, $G(z)$ is the team objective function for joint-trajectory $z$, and $G(z_{-i})$ is the team objective function with agent $i$'s trajectory removed.

The interesting part about this difference objective that we experiment with is how to properly compute $z_{-i}$. In our leader-follower team, only the leaders are learning, but they strongly influence the trajectory of the followers. This makes it so that removing just a leader's trajectory when computing $G(z_{-i})$ may not capture what the team objective function would have been without this leader. We detail two different methods for calculating $G(z_{-i})$.

**Leader Trajectory Removal:** This is the baseline that we compare our other methods against. With our baseline, we simply remove the trajectory of the leader, and do not try to account for the impact the leader had on any followers.

**Leader-Follower Trajectory Assignment:** We assign followers to leaders such that there is a one to one mapping of followers to leaders. One follower cannot be assigned to more than one leader. To assign a follower to leaders, we track at every timestep what leaders are within the observation radius of that follower. A follower has a counter for every leader, and that counter is incremented by 1 for every timestep the corresponding leader is observed by that follower. At the end of the episode, we aggregate all of these counters, and whichever leader has the highest counter is the one that we assign that follower to.

## 4.3 CCEA with n-Elites Binary Tournament

We use a Cooperative Co-Evolutionary Algorithm for learning, formalized in algorithm 1. We initialize $N$ populations of $k$ neural networks, where we have $N$ learning agents, or "leaders", on a team, and we have $k$ policies we are tracking for each agent. We start the co-evolutionary process by evaluating all of the policies. To evaluate policies, we randomly form $k$ teams of policies, with each policy represented once across all teams. Each team is evaluated in an episode of the arctic monitoring environment, and assigned a fitness score according to the team objective function. At this point, each policy on a team is assigned either the team fitness score or a difference evaluation. This depends on the particular trial.

**Algorithm 1** CCEA with n-Elites Binary Tournament

```
Initialize N populations of k neural networks as policies
for Generation do
    /* Team Evaluation */
    for i = 1 → k do
        Randomly select policy from each population
        Add policies to agents on team T_i
        Simulate T_i in environment
        Assign score to each agent in T_i
    end for
end for
/* n-Elites Binary Tournament */
for Population do
    Initialize new, empty population
    Add top n policies
    for i = n → k do
        Compare 2 policies at random
        Add highest scoring policy to new population
        Mutate policy in new population
    end for
end for
```

Once each policy has a score assigned to it, we downselect each population of policies. For our downselection process, we run an n-elites binary tournament, where n=1. We create an empty new population, and add the highest scoring policy from the original population. To fill out the rest of the population, we iteratively select two policies from the original population at random, compare their scores, add the highest scoring policy to the new population, and mutate that policy in the new population. This process helps to preserve the diversity of policies in the co-evolutionary process. We repeat the evaluation and downselection processes for a specified number of generations to learn with our CCEA.

## 4.4 Experiments

We test the performance of the leader-follower approach with three agent and POI configurations to evaluate the ability of the agents to learn to successfully learn to observe the POIs. These configurations are additionally tested using the different reward structures described in section 4. Each of the layouts were designed in order tests how the agents learns to interact together to accomplish a tightly coupled task. These are shown in figure 3. All the positions shown are drawn to scale with respect to the implementation configurations.

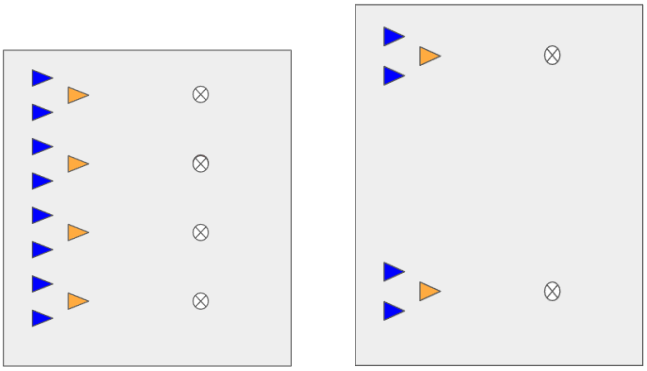
**Generally Forward:** This configuration is meant as a simple problem where the agents only need to learn how to move generally forward to the POIs. Each POI has a coupling requirement of 3, and each leader is positioned to be able to initially influence two other followers, when using the leader-follower paradigm.

**Strictly Forward:** The next configuration makes the previous configuration harder by removing the center two sets of agents and POIs. This makes the problem harder by requiring the agents to move in a much more directed direction that in the first configuration. More explicitly, in the first configuration, the agents only needed to move in roughly a forward direction to receive some form of a reward. However, in this configuration, the agent must move straight forward to observe the POI along its path.

**Chaotic Interactions:** The final configuration adds two more agents to nearby each leader along with increasing the coupling requirement to 5. Note that this increase is commensurate to the additional agents added to the system, so each leader would still be expected to guide a set of unique agents to observe a POI. However, each follower is additionally influenced by the followers on either side of it. In all, this presents a much more challenging problem to the leader-follower paradigm specifically due to all of the additional interactions between followers, other followers, and leaders that impact the ability of the system to receive any system reward.

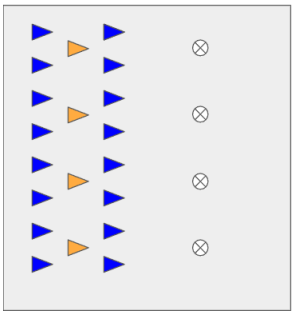
# 5 RESULTS

In this section, we analyze the ability of the various agent and POI configurations to learn to solve the task using several reward functions. In general, the agents using the leader-follower paradigm were able to learn a more optimal joint policy faster and more reliably than those where all the agents in the system were learning agents.



**(a) The simplest configuration used to evaluate the agent and reward structures. Each POI has a coupling requirement as 3.**

**(b) The second configuration removed the center two sets of agents and POIs while keeping the coupling requirement of each POI as 3.**



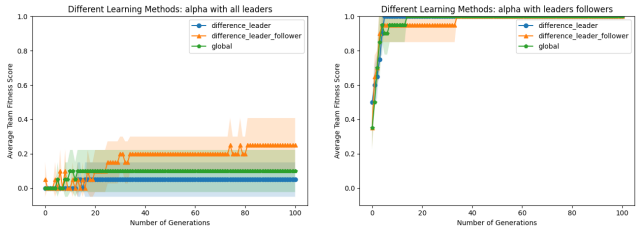
**(c) The final configuration adds two more agents nearby to each leader position and increases the coupling requirement of each POI to 5.**

**Figure 3: Each POI is marked as a grey 'X' in a circle. The blue triangles represent where the followers are placed in the leader-follower paradigm. And the orange triangles denote where the leaders are placed. When all agents in the system are leaders, both the blue and orange triangles are evolving their policies.**

## 5.1 Generally Forward

The simplest configuration requires the agents to learn to move in a general forward direction. For the system with all learners, the problem is simpler due to the likelihood of three of twelve agents simultaneously stumbling upon a POI. As for the leader-follower paradigm, The mechanics of the follower updates would lead to the coupling requirement being met if the leader modulates its speed so as to not leave behind, or fall behind, the followers.

Even though the policy the leaders must follow is rather straightforward, they struggle to learn to observe all 4 POIs using the global or difference reward signals, as shown in figure 4a. After 100 generations, they average observing less than a single POI. Contrasted with this, the agents learning using the leader-follower paradigm quickly learn to observe all 4 POIs using all of the reward functions.




(a) (b)

**Figure 4: When using all leaders, the agents are unable to learn to effectively observe more than a single POI on average. Whereas when using the leader-follower paradigm, the agents are quickly able to learn to observe all of the POIs.**

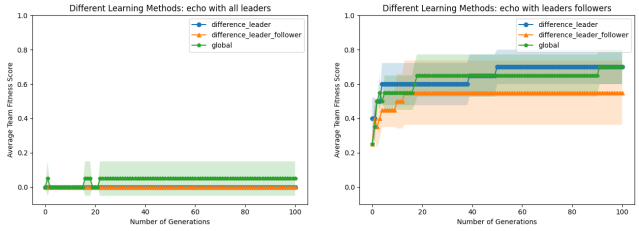
In fewer than 20 generations, they all observe at least 3, on average, and they all consistently observe all 4 POIs within 40 generations.

This suggests that the learning problem required of the leader-followers is simpler than the equivalent problem when all the agents are trying to learn a joint policy from scratch.

## 5.2 Strictly Forward

In the second configuration, the center sets of agents and POIs were removed. For both the all learners and the leader-follower paradigms, this restricts the policies the leaders must learn. Same as before, they must learn to move in a general forward direction. However, in order to successfully receive any reward, they are not able to deviate much from this forward direction to the reward.

When looking at the performance of the system using all leaders, shown in figure 5a, the agents were almost never able to learn to obtain any reward using any of the reward functions. This is largely due to just how sparse any system feedback may be, as in order to obtain any reward, three agents must coordinate simultaneously over a period of time to reach the POI. Even if a single agent acts in a correct manner, they may not receive any positive feedback as their performance is intrinsically linked to the performance of the other agents in the system.




(a) (b)

**Figure 5: The ability of the all learners approach to fail to solve the problem is more drastic than in the previous configuration. Additionally, the leaders-followers paradigm continues to be able to perform well rather quickly, learning to observe 3 of the 4 POIs on average withing 30 generations.**

Switching to the leader-follower paradigm, shown in figure 5b, the agents are able to use any of the reward functions to successfully solve the task. However, the impact of the various reward structures is not as readily apparent based solely on the average reward. Instead, when using the leader-follower trajectory assignment, the variance of the performance of the system was significantly higher, suggesting that this method may introduce noise in the system under particular configurations. Looking further into why this may be for this configuration, due to the environmental mechanics and state representation, an agent is likely going to continue along their initial policy after the first few time steps. During these steps, the followers will adjust themselves to the directions of the leaders, and so, for most time steps beyond this initial period, the observation of the agent is not likely to change. They cannot see the POIs or the other agents, and the followers will remain in a similar relative location to the leader. Thus, when the leader-follower trajectory assignment tries to account for the effect of the leaders and its associated followers, it adds too much noise to the leader's reward as it assumes the leader is actively guiding the followers during that entire trajectory. Instead, it is making slight nudges in the beginning then allowing the followers to mostly follow their current trajectory to reach the POI.

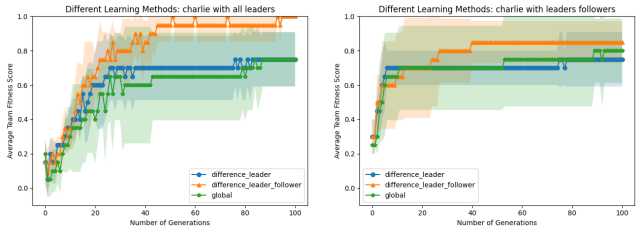
Compare this effect to the global reward and the leader trajectory removal. The global reward is, by definition, aligned, but the misalignment of the leader-follower trajectory assignment comes from the agents having to focus more on their ability to shepherd the followers in the correct direction. Relative to the leader trajectory removal, removing both the leaders and the followers is less sensitive to the actions of the leader itself due to a similar cause, but instead of having to learn a slightly different functional form, the reward structure introduces noise to the learning agents through trying to model the effect of its actions on its followers. In this configuration, the behavior the learning agents must exhibit to accomplish the task is much simpler than potential tasks that may require more explicit modeling of its impact on its assigned followers.

## 5.3 Chaotic Interactions

The final configuration adds more agents to the system and increases the coupling requirement of the POIs. With all leaders, the difficulty in this configuration comes from the coupling requirements of the POIs, while for the leader-follower paradigm, the difficulty arises due to how learners must interact with the followers. Specifically, while the followers are closer to the POI than the leaders, the update rules of the followers will direct them back towards the leader and away from the POI. But the followers further away from the POI than the leader will move in the correct direction needed to correctly help to observe the POI.

When all the agents are learners, due to the number of agents in the system relative to the POIs, and that there is no restriction to learners only observing a POI, there is a high likelihood of random actions successfully causing the agents to observe the POI. This can cause the agents to be able to learn a successful policy, but with a rather large variance in the average policy performance.

Using the leader-follower paradigm, the agents learn about a similar policy. The difference comes in that they learn this policy




(a) (b)

**Figure 6: The all learners approach is able to solve the problem some of the time due to the number of agents in the system and the likelihood that random actions might cause enough agents to simultaneously be within the observation radius of a POI. However, using the leader-follower paradigm, the agents are able to learn at least a comparable policy, but do so much faster and more consistently than when using all learners.**

much faster and more consistently. Effectively, the leader-follower paradigm is able to simplify some of the chaos in the problem and clean up the ability of the agents to discern how to act to learn to solve the task. However, while these is a slight benefit of using the leader-follower trajectory assignment, the global reward is able to provide enough of a reward that the agents do not receive much of a benefit in being able to better discern the effect of their actions specifically within the context of its teammates.

The ability of the global reward to provide enough of a learning signal to direct agent to solve the task suggest that the leader-follower paradigm is powerful in simplifying the learning task required of the agents to solve an equivalent problem, given they are provided information about the *structure of the task* in the form of the simple follower update rules.

## 6 CONCLUSION

The ability of leaders directing followers using simple pre-programmed behaviors allows for simplifying the learning task and allows agents to better make use of the information provided to them by the system reward. Additionally, this manner of directing the followers based the actions of leaders allows a designer to inject some knowledge about the structure of the task to the agent. Thus, as the number of agents required to interact in a system increases, the agents are able to solve more complex problems faster than when using all learners to accomplish the same task.

A trouble with this approach comes in as the number of followers increases relative to the learners, as the learners must learn not only how to accomplish the goal, but also how to shepherd the followers along to aid the leaders in the required task. In this way, it puts more onus on the leader to learn an effective policy. Still, even as this ratio of followers to leaders is increased, the leader-follower approach is able to find a similarly performant policy faster than when using all learners.

Continuing on this work could further explore the effect of greatly altering this ration of leaders to followers, or by requiring setups where agents must learn to assert influence over followers at different times in order to observe a set of POIs. This would likely require a more informative difference reward in order to account for this switching behavior. Another avenue might be to address how this type of framework might be used with heterogeneous agents. For instance, where the leaders are unable to directly observe the POIs and must rely on how their exert their influence to cause the followers to observe the POIs. In these types of scenarios, the reward provided to the learning agent is further removed from its direct actions and makes the actual problem the leaders must learn different than the task required by the system to receive a reward.

## REFERENCES


[1] Mitchell Colby and Kagan Tumer. 2012. Shaping Fitness Functions for Coevolving Cooperative Multiagent Systems. (2012), 8.

[2] Joshua Cook and Kagan Tumer. [n. d.]. Ad hoc teaming through evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (Lille France, 2021-07-07). ACM, 89–90. https://doi.org/10.1145/3449726.3459560

[3] Lei Feng, Yuxuan Xie, Bing Liu, and Shuyan Wang. [n. d.]. Multi-Level Credit Assignment for Cooperative Multi-Agent Reinforcement Learning. 12, 14 ([n. d.]), 6938. https://doi.org/10.3390/app12146938

[4] Dario Floreano, Peter Dürr, and Claudio Mattiussi. [n. d.]. Neuroevolution: from architectures to learning. 1, 1 ([n. d.]), 47–62. https://doi.org/10.1007/s12065-007-0002-4

[5] Alexander Gee and Hussein Abbass. 2019. Transparent Machine Education of Neural Networks for Swarm Shepherding Using Curriculum Design. In *2019 International Joint Conference on Neural Networks (IJCNN)*. 1–8. https://doi.org/10.1109/IJCNN.2019.8852209

[6] Jorge Gomes, Pedro Mariano, and Anders Lyhne Christensen. 2016. Cooperative Coevolution of Partially Heterogeneous Multiagent Systems. (2016), 9.

[7] Faustino Gomez, Jürgen Schmidhuber, and Risto Miikkulainen. [n. d.]. Efficient Non-linear Control Through Neuroevolution. In *Machine Learning: ECML 2006*, Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou (Eds.). Vol. 4212. Springer Berlin Heidelberg, 654–662. https://doi.org/10.1007/11871842_64 Series Title: Lecture Notes in Computer Science.

[8] Aya Hussein, Eleni Petraki, Sondoss Elsawah, and Hussein A Abbass. 2022. Autonomous Swarm Shepherding Using Curriculum-Based Reinforcement Learning.. In *AAMAS*. 633–641.

[9] Nathan K. Long, Karl Sammut, Daniel Sgarioto, Matthew Garratt, and Hussein A. Abbass. 2020. A Comprehensive Review of Shepherding as a Bio-Inspired Swarm-Robotics Guidance Approach. *IEEE Transactions on Emerging Topics in Computational Intelligence* 4, 4 (2020), 523–537. https://doi.org/10.1109/TETCI.2020.2992778

[10] Duc Thien Nguyen, Akshat Kumar, and Hoong Chuin Lau. 2018. Credit Assignment For Collective Multiagent RL With Global Rewards. (2018), 12.

[11] Tung Nguyen, Jing Liu, Hung Nguyen, Kathryn Kasmarik, Sreenatha Anavatti, Matthew Garratt, and Hussein Abbass. 2020. Perceptron-Learning for Scalable and Transparent Dynamic Formation in Swarm-on-Swarm Shepherding. In *2020 International Joint Conference on Neural Networks (IJCNN)*. 1–8. https://doi.org/10.1109/IJCNN48605.2020.9207539

[12] Aida Rahmattalabi, Jen Jen Chung, Mitchell Colby, and Kagan Tumer. [n. d.]. D++: Structural credit assignment in tightly coupled multiagent domains. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2016-10). 4424–4429. https://doi.org/10.1109/IROS.2016.7759651 ISSN: 2153-0866.

[13] Kenneth O. Stanley, Jeff Clune, Joel Lehman, and Risto Miikkulainen. [n. d.]. Designing neural networks through neuroevolution. 1, 1 ([n. d.]), 24–35. https://doi.org/10.1038/s42256-018-0006-z

[14] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. [n. d.]. Value-Decomposition Networks For Cooperative Multi-Agent Learning. arXiv:1706.05296 [cs] http://arxiv.org/abs/1706.05296

[15] Karl Tuyls and Gerhard Weiss. [n. d.]. Multiagent Learning: Basics, Challenges, and Prospects. 33, 3 ([n. d.]), 41. https://doi.org/10.1609/aimag.v33i3.2426

[16] Ovunc Tuzel, Gilberto Marcon dos Santos, Chloë Fleming, and Julie A Adams. 2018. Learning based leadership in swarm navigation. In *International Conference on Swarm Intelligence*. Springer, 385–394.

[17] Pradnya A. Vikhar. 2016. Evolutionary algorithms: A critical review and its future prospects. In *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*. 261–265. https://doi.org/10.1109/ICGTSPICC.2016.7955308

[18] Chern Han Yong and Risto Miikkulainen. 2000. Cooperative Coevolution of Multi-Agent Systems. (2000), 15.