Learning Flocking Control in an Artificial Swarm

Ovunc Tuzel, Chloe Fleming, Gilberto Marcon
Collaborative Robotics and Intelligent Systems Institute
Oregon State University
Corvallis, OR 97331
tuzelo | flemichl | marcondg @oregonstate.edu

Abstract

In this paper, we introduce a control strategy for artificial swarms that leverages a small number of trained leaders to influence the trajectory of the flock in spatial navigation tasks. Artificial swarms have potential to solve spatially distributed real-world problems, such as search and rescue, monitoring, and cleanup tasks, because they are robust and scalable with decentralized control. However, controlling the trajectory of a swarm requires communicating navigational goals to the swarm agents, which is infeasible for large swarms or agents that have limited sensing and communication abilities. We address this challenge with a biologically-inspired approach, where informed individuals that are able to exert influence over their peers steer the group towards its destination. Deriving a control strategy for these leaders is a challenging problem, because artificial swarms have complex emergent behaviors that are difficult to model and predict. Our key contribution is a neuro-evolutionary learning-based control method in which a subset of swarm agents are trained to strategically influence the behavior of other agents. We apply this control method in increasingly complex navigational tasks, observing that its effectiveness depends on the proportion of leaders in the swarm, and that a learning-based approach may outperform a simple heading bias policy with a sufficient proportion of leaders.

1 Introduction

Large-scale multi-robot teams have the potential to accomplish a wide variety of distributed field applications, such as search and rescue, mapping, and scouting, but controlling large teams of robotic agents is a challenging problem. Biologically-inspired swarms of robotic agents, called artificial swarms, are well-suited to real-world spatial coverage tasks because they are designed to exhibit emergent behavior based on relatively simple local interactions between agents. Artificial swarms can implement biomimetic flocking behavior when each agent uses a simple control policy to choose its heading based on observations of other nearby agents. Artificial swarms are based on distributed local interactions; thus, they do not have a single point of failure, and they are scalable to very large numbers.

Although this decentralized control paradigm is robust, it is difficult to plan and execute complex tasks with an artificial swarm because control over the trajectory of the group is limited. Prior work has demonstrated that inserting a few leader agents with knowledge of the destination improves the success of a biological swarm at migrating towards a goal, even with a small number of leaders, relative to the group size [9]. Examples from biology suggest that often only a few members of biological swarms can lead the group towards the goal [9]. In our approach, a small number of leaders are trained to influence the behavior of the rest of the swarm, allowing the group to achieve complex navigational tasks.

---

1This project is closely related to our Multi-Robot Systems (ROB 599) course project, but will have a distinct goal and deliverables. The focus for ROB 537 will focus on formulating a learning approach that can produce behaviors that extend to increasingly complex navigation scenarios. For ROB 599, we are exploring the relationship between agent communication model, leadership behaviors, and minimum number of leaders.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.
migratory groups are informed of the goal. The leaders use movement behaviors that vary from their neighbors in order to more effectively drive the swarm towards its objective [4, 17].

Flocking behavior can be realized by artificial swarms with an attraction-orientation-repulsion policy, known as Reynold’s flocking rules [5, 20]: agents move towards neighbors within an attraction radius, determine heading by averaging the direction of neighbors within an orientation radius, and maintain an appropriate distance from neighbors within a repulsion radius. These distributed interactions are the means for communicating navigational cues between agents, and leaders leverage these properties to influence nearby agents in the swarm.

Biological flocks of diverse animal groups such as fish, birds, bees, and cattle all demonstrate leadership behaviors in which a few informed individuals guide migratory trajectories from within the swarm. When fish that have been trained to forage in brightly lit regions of their environment are inserted into a shoal of naive fish, the trained fish are able to guide the shoal to the food source, even though the majority of the shoal has no knowledge of the foraging preference [19]. The fastest pigeons in a flock become the leaders, and then these pigeons learn to be better at navigating, which improves the quality of their leadership [17]. Swarms of honeybees relocating to their new hives appear to be guided by a few fast-flying members that provide visual signals to guide the trajectory, and these individual honeybees are presumed to be scouts that visited the new nest site as part of the process of selecting a new home [1]. Frontal fish tend to have much greater influence on the direction of the shoal compared to rear fish; thus, food deprived individuals in a fish shoal tend to swim faster and occupy frontal positions in the shoal, in order to improve chances at reaching a food source [12].

Inserting leader agents into a swarm offers greater control over the swarm without compromising the scalability and robustness of the swarm. Although leaders play an important role in the migratory behavior of biological flocks, it is unclear how they tune their behaviors to maximize their influence on neighbors and ultimately steer the swarm. It is challenging to analytically derive explicit strategies to exert control over the swarm, and designing fixed rules for leaders to influence the swarm is a naive approach. An adaptive strategy in which leaders are trained to steer the flock can be applied broadly to spatial navigation tasks in order to improve performance of the swarm. Trained leaders can incorporate behaviors adapted to emergent flocking dynamics and influence the trajectory of the swarm to successfully achieve high-level tasks, such as reaching waypoints and avoiding danger zones. This project applies neuro-evolutionary reinforcement learning to develop control policies for leader agents to control flock navigation.

The main objective of this work is to implement a learning-based approach to flock navigation that leverages designated leader agents as distributed control points. The key contributions include the application of learning-based methods to train intelligent agents to influence other naive agents, as well as an analysis of swarm state dimensions for parameterization of the leader’s control policy. Training is performed with neuro-evolutionary algorithms, and neural network inputs are explored to discover high performance parameters that reflect real world scenarios.

2 Related Work

The application of learning-based control to an artificial swarm involves understanding challenges in multi-agent learning. The following sections provide a brief overview of multi-agent learning, related efforts to automatically generate controllers for swarm agents, and examples of leadership in multi-robot systems.

2.1 Multi-Agent Learning

Panait and Luke [16] divide multi-agent learning approaches into two categories, team learning and concurrent learning, and discuss the advantages of each. In team learning, a single learning process derives policies for multiple agents at once based on global performance, whereas concurrent learning develops policies for individual agents based on their own contributions to the system state. Team learning is often a simpler approach, but is inherently centralized. Team learning may be homogeneous if all agents share a single policy, or heterogeneous if the learning approach identifies different policies for different agents in the multi-agent system.

In concurrent learning or heterogeneous team learning, a key challenge is assigning appropriate rewards to agents based on their actions. Colby et al. combine global performance and difference
evaluations to estimate each agent’s contribution to the team performance, leveraging both metrics to compute fitness in neuro-evolutionary training in a multi-robot POI surveillance problem domain [7].

According to a classification of multi-agent systems by Stone and Veloso, swarms executing a flocking behavior fall into the simplest category, with agents that are homogeneous, reactive, and non-communicating [22]. Although the swarms literature often refers to interactions between agents as communication, the extent of this information transfer is limited to observations of neighboring agents. In these types of multi-agent problems, agents must learn to affect each other by observable spatial interactions [22].

Local reinforcement signals, in which agents are rewarded based on their immediate actions, can be used to evolve specialization in swarms. Specialization in large collectives can result in improved task performance [15], but these signals are noisier and it is more difficult to estimate the value of an individual’s action [13]. Li et al. discuss various strategies for tackling the credit-assignment problem in swarms; in a homogeneous or caste-based flock, the agents can often share a global reinforcement signal since each agent contributes to swarm performance in the same way, on average [13].

2.2 Learning Swarm Controllers

Learning-based methods for deriving controllers are commonly applied to artificial swarms, because generating agent controllers based on a top-down behavior requirement is a difficult and mostly unsolved problem for systems with emergent behaviors [5]. In most cases, the agents are trained using a team learning approach in which all agents use identical controllers, which eliminates challenges such as multi-agent credit assignment [1–3, 11, 18].

Ampatzis et al. [1] apply a neuro-evolutionary approach to explore communication mechanisms between robots, using a population of fully-connected 13-neuron CTRNNs to evolve a controller utilized by all robots in the system. Although the agents in this work use simple decentralized controllers characteristic of swarm agents, the authors only conducted tests with two robots. A similar work explored using neuro-evolutionary methods to derive controllers for up to 16 simulated agents to converge around an object and transport it to a target location; however, this approach searched for a controller over a large state-action space that permitted a wide variety of behaviors but required several weeks of computation time to train [11].

Baldassarre et al. [2] demonstrate an early effort to evolve controllers for swarm agents to aggregate around a light signal, in which light, sound, and infrared sensor readings are mapped directly to motor commands using a neural network. The controllers were trained based on global fitness of the swarm based on group speed and compactness. In a later work, this approach was applied to a more complex scenario in which the robots must form a linear structure after starting in random orientations; the controller evolved in simulation was applied to real robots with a small performance degrade [3].

Pugh and Martinoli [18] focus on analyzing the quality of learned controllers for a swarm of real robots with heterogeneous infrared sensors, comparing the effectiveness of a genetic algorithm against a particle swarm optimization approach. The controllers generated by particle swarm optimization consistently performed better in their experiments; the researchers attributed this to the fact that the particle swarm method promotes significantly better population diversity.

These works suggest that neural network controllers are an effective means of controlling artificial swarms, but indicate that role-based training of castes within the swarm is a relatively unexplored area.

2.3 Leaders & Shepherds in Multi-Robot Systems

Insight into leadership mechanisms can help guide the design of learning-based leadership in artificial swarms. Çelikkanat and Şahin [6] conducted experiments with inserting informed leaders into a swarm of naive agents, proposing that a flock is steerable if it forms an aggregate containing the leaders, and if the leaders are additionally able to propagate heading information to members of the aggregate. Their work suggests that steerability improves with the size of the flock and when each agent perceives navigational cues from a larger number of neighbors. Ferrante et al. [10] propose a novel method for using flocking interactions to control the agents’ speed as well as heading, demonstrating that this extension to flocking dynamics enables migrating flocks to perform better even with smaller ratios of informed leaders.
An early research effort to learn multi-robot shepherding behaviors leveraged a genetic algorithm to select a set of control rules, each of which maps the leader’s sensory inputs to a steering or velocity action, and achieved moderate success rates with a single leader and follower \[21\]. Another preliminary work in shepherding artificial flocks used a model-based approach, consisting of algorithms for driving one or more leader agents to steering points from which they could influence the position of the flock by moving towards the other agents \[14\]. A major limitation of this approach is that it requires enumerating specific steering points, which means that it does not generalize well to previously unseen environments.

Toshiyuki et al. devise a method to improve flocking cohesiveness in artificial swarms by designing an adaptive method for swarm agents to switch between two roles, leader and follower, with predefined control policies based on Reynold’s flocking rules \[23\]. In this case, agents of both roles are assumed to have some knowledge of the goal location; leader agents are strongly biased towards the goal, and followers are weakly biased towards the goal and strongly pulled towards robots in front of them. This approach requires the goal location to be communicated to all swarm agents, which is unfeasible for systems intended to scale up to very large numbers.

3 Approach

A key contribution of this paper is identifying appropriate swarm state metrics to provide a reinforcement signal for training the leaders to influence the flock. Leaders exert control over the flock by moving among the group and applying attraction-orientation-repulsion forces on agents; thus, a control policy for steering the swarm’s trajectory positions the leader to apply this pressure effectively. The relative distance and heading to the spatial centroid of the flock’s position, the goal, and obstacles are all candidate inputs to the leader’s control policy. Experiments were selected to analyze the relative usefulness of subsets of these parameters, based on the performance of the resulting neural network controller. The leader’s heading is the primary component of its controllable state that is a learned output of the controller.

An evolutionary algorithm is used to generate a population of neural network controllers that perform this mapping between flock configuration and leader control action. The leaders are trained as a homogeneous caste using a team learning approach: thus, all leaders in a given episode will share the same neural network controller. The effectiveness of trained leaders is assessed relative to the performance of simple informed leaders that always bias their heading towards the goal \[9\], with performance measured in terms of average distance between swarm agents and the goal over the course of an episode.

3.1 Neuro-Evolutionary Learning Method

A set of \(n\) nonholonomic robots, \(R = \{r_1, r_2, \cdots, r_n\}\), navigates a continuous 2-D world at constant velocity. Each robot controls its own direction of movement by defining its desired heading angle \(\psi\), a continuous value in the range \((-\pi, \pi)\).

A subset \(L\) of robots \(R, L \subset R\), consists of leader robots, and the remaining robots, \(S = R - L\), consist of swarm robots. Swarm robots and leader robots have the same physical capabilities, but leader robots are driven by a neural network and swarm robots are driven by swarm control rules. The same neural network, with the same weights, is cloned across all leaders in \(L\) at any given time. Whenever a different neural network is used, it is applied to all leaders in \(L\).

A set of \(m\) two-layer neural networks \(A = \{a_1, a_2, \cdots, a_m\}\) is initialized with random weights. The random weight generation for initializing the neural-networks comes from a Gaussian distribution centered at zero and with a standard deviation of one \((\mu = 0, \sigma = 1)\). The hidden layer and the output layer of the neural networks consist of units with arc tangent activation functions. Arc tangent activation functions were adopted because of their symmetry around the origin, and because of their limited output in the range of \((-\pi, \pi)\), consistent with the desired heading angle \(\psi\) output control of the leader robots.

3.1.1 The Neural Networks’ Inputs and Outputs

The neural networks have up to six sensory inputs, derived from the leader’s perception of their environment. The inputs are the polar coordinates of three points of interest to the leader robots,
relative to the robot reference frame. The first point of interest is the location of the goal position, the second point of interest is the centroid of the swarm, and the third is the closest obstacle within the leader’s perception range. The polar coordinates of each point of interest represents the distance between the robot and the point of interest, \( d \), and the angle between the robot and the point of interest, relative to the robot’s heading, \( \theta \). The first pair of inputs to the leader’s neural network are the polar coordinates to the goal position, \( \langle d_g, \theta_g \rangle \). The second pair of inputs to the leader’s neural network are the polar coordinates to the neighboring swarm’s centroid, \( \langle d_s, \theta_s \rangle \). The third pair of inputs to the leader’s neural network are the polar coordinates to the closest obstacle, \( \langle d_o, \theta_o \rangle \). In summary, all possible inputs to the neural network are defined by the input vector

\[
N_{x} = \langle d_g, \theta_g, d_s, \theta_s, d_o, \theta_o \rangle
\] (1)

Various neural networks based on subsets of these inputs are utilized for experiments in this paper. The inputs to the neural network are presented in Figure 1 from the leader reference frame. The output of the neural network is the desired heading to the leader robot

\[
N_{\psi} = \langle \psi \rangle
\] (2)

Figure 1: Leader inputs. The leader is represented by the orange triangle at the origin of the coordinate axis. The goal location is represented by the blue flag at the top-left corner. The nearest obstacle is represented by the rock at the bottom-left corner. The swarm centroid is represented at the bottom-right corner. All neural network inputs are computed relative to the leader reference frame in polar coordinates.

3.1.2 The Learning Process

Each learning epoch consists of simulating the behavior of the team of robots while performing a task for all \( m \) neural networks in the population set \( A \). An episode consists of loading a neural network \( a_i \) into the leader robots \( L \), positioning the leader robots \( L \) and the swarm robots \( S \) randomly within their start location area, and simulating their behavior for a fixed number of simulation steps \( \tau \). At the end of each episode, the performance of the current neural network \( a_i \) is evaluated using the cost function \( E \), presented in Section 3.1.3 and the next neural network \( a_{i+1} \) is loaded for evaluation.

When all \( m \) neural networks are evaluated, the top-performing \( \lambda \) neural networks, called parents, are kept for the next epoch and the remaining \( m - \lambda \) neural networks are eliminated from the set.
Parents are, then, randomly selected to generate the next generation of neural networks. $m - \lambda$ neural networks are generated by randomly sampling (with replacement) from the set of $\lambda$ parents. Copies are generated and then mutated by applying zero-mean Gaussian noise with a fixed standard deviation $\mathcal{N}_{mut}(\mu = 0, \sigma = \mathcal{N}_{mut})$ to every neural network weight.

### 3.1.3 The Cost Function

At each step $t$ of the simulation, the temporal factor $\frac{t}{\tau}$ represents time progress over the episode number of steps $\tau$. The temporal factor is a number in the range $[0, 1]$, where 0 indicates the first step of the episode, and 1 indicates the last step of the episode. The temporal weight $w_t$ is defined as a function of the temporal factor,

$$w_t = 1 - \cos\left(\pi \cdot \frac{t}{\tau}\right),$$

and presented by Figure 2 over its valid input range, $\frac{t}{\tau} \in [0, 1]$. The area under the curve of the temporal weight function, $w_t(t)$, is 1.

The Euclidean distance between each swarm robot $i$ and the goal location at step $t$, $d_i^t$, is evaluated, weighted by the temporal weight $w_t$, and accumulated over every step of the episode to compute the average weighted distance $d_{avg}^i$:

$$d_{avg}^i = \frac{1}{\tau} \cdot \sum_{t=0}^{\tau} d_i^t \cdot w_t$$

(4)

The average weighted distance $d_{avg}^i$ is the cost associated with swarm robot $i$ for the entire episode. The temporal weighting makes the influence of swarm robot distances from the goal heavier later in the episode, when $t$ is closer to $\tau$, and lighter early in the episode, when $t$ is close to 0, in a smooth transition. The temporal weight $w_t(t)$ has the purpose of prioritizing the distance between swarm robots and the goal late in the episode, as opposed to early in the episode for computing cost.

At the end of each episode, the average accumulated weighted distance $d_{avg}^i$ is averaged across all swarm robots in $S$, defining the neural network cost function, $E$:

$$E = \frac{1}{|S|} \cdot \sum_{i \in S} d_{avg}^i$$

(5)

### 4 Training Leaders in a Simulated Swarm

In order to evaluate learning-based leadership in an artificial swarm, an environment for testing and training agents was developed in C++ with OpenGL visualization. This framework is tunable for
a wide variety of experiments involving variable numbers of leaders, agents, interaction radii, and obstacle densities. In each episode, the agents, obstacles, and goal are randomly redistributed within a 2D test region, and the performance of the leaders at guiding the flock to the goal accumulates over every time step in the episode. All experiments use a repulsion radius of $r_r = 40$, an orientation radius of $r_o = 50$, and an attraction radius of $r_a = 80$; the world size is 1000x1000 and the agents occupy a 16x16 square. The agents are simulated as point objects without dynamics.

Figure 3: A snapshot of the flocking simulator environment used in this project. Obstacles are shown as square gray blocks, and the goal location is marked by a green flag. All agents are simulated as point objects without dynamics, and are drawn as triangles. The red agents are leaders, and the blue agents are naive.

The primary motivation of this project is to assess the potential for trained leaders to serve as a mechanism for controlling swarms in real-world applications; thus, the experiments selected for this project explore important considerations for deploying such a system in the real world. In particular, experiments were designed to evaluate the impact of the ratio of leaders and to select appropriate learning parameters in obstacle-filled environments.

4.1 Ratio of Leaders

An objective of introducing leader agents into swarms is to eliminate the need to communicate mission goals to all members in a large-scale system. Communication can be a limited resource in multi-robot systems, so minimizing the number of contact points is desirable. An experiment assessing the relative performance of swarms with variable proportions of trained leaders was performed with swarms consisting of 20 agents (including leaders). For each value of $l \in \{0, 2, 4,...,18\}$, 100 swarms were composed with $l$ leaders. Each swarm’s leaders were trained for 50 epochs using the weighted distance performance metric to assess fitness. Training was conducted in the absence of obstacles, so inputs to the neural network controllers were the relative distance and angle to the goal and swarm’s centroid, $\langle d_g, \theta_g, d_s, \theta_s \rangle$. Each episode consisted of 700 steps, and training was performed for 50 epochs. The results from this experiment are summarized in Figure 4.

4.2 Learning Parameters for Environments with Obstacles

In a real-world scenario, the assumption that swarm leaders can estimate the relative position of the swarm’s centroid may not be realistic, especially if there are obstacles in the environment. Swarm robots are generally equipped with relatively simple sensors, such as ranging systems or low resolution cameras. Nevertheless, the swarm’s state may be a key input to the leader’s control policy, if the leader aims to guide the swarm to a goal destination.
Figure 4: Changes in swarm performance based on the ratio of leaders to naive agents. Performance is measured as sum of weighted distances between agents and the goal over the course of the episode, normalized for the number of agents. In all cases, there are 20 non-leader agents in the swarm and no obstacles, and results are summarized over 100 trials. **Top:** Learning curves indicating the average controller performance as a function of training time. **Bottom:** A summary of swarm performance as a function of the ratio of leaders, averaged over all controllers in the populations for the last 10 epochs.
Figure 5: Evaluation of leader training parameterization in obstacle-filled environments: neural network controllers were trained with only obstacle and goal locations as inputs (yellow/dash) or with obstacle, goal, and swarm centroid inputs (orange/solid). Performance for swarms consisting of only naïve agents (blue/points) and simple informed agents (purple/dash) are shown for comparison. Swarms consisted of 20 agents and training was performed for 150 epochs using episodes of 2000 steps, with 100 repetitions of each combination of parameters. **Top:** 20% leaders. **Bottom:** 40% leaders.
Figure 6: Swarm performance as a function of leader density for swarms of 50 (yellow/light) and 100 (red/dark) agents. Top: Weighted average distance metric. Lower values indicate better performance. Bottom: Proportion (scale [0, 1]) of agents at the goal in the final episode timestep. Higher values indicate better performance.

In order to evaluate this dependency on observing the swarm’s state, swarms were trained with varying inputs to the neural network controllers. Two variations of swarm centroid observability were tested, with 100 repetitions of each scenario: none and perfect. For none, the goal location and obstacle location \((d_g, \theta_g, d_o, \theta_o)\) was given as inputs to the neural networks. The other variation trained controllers with the complete input vector \((d_g, \theta_g, d_s, \theta_s, d_o, \theta_o)\). 75 obstacles were scattered randomly in the test arena at the beginning of each episode. The episode length was increased to 2000 to encourage long term navigation and avoidance strategies to develop, and each population of controllers was trained for 150 epochs. This experiment was conducted for leader ratios of 20% and 40% in a flock of 20 agents, and performance as a function of training time is plotted in Figure
For comparison, flocks consisting only of naive agents (no leaders) and flocks led by simple informed agents (with a constant policy of biasing heading towards the goal) were tested in the same environments, and performance of these groups is plotted for comparison.

4.3 Increasing Swarm Size

A key goal of enabling leader-based control of artificial swarms is to eliminate the need to communicate mission instructions to all agents, because communication dependencies quickly become infeasible as the swarm size grows. In order to investigate the relationship between the size of the swarm and the proportion of leaders required to control it, experiments were conducted with swarms of 50 and 100 total agents with leader ratios of 10%, 20%, and 30%. As in the previous experiment, obstacles were included in the environment, so all six neural network inputs were used and the episode length was set to 2000 steps. Figure 6 reports statistical performance results, summarized over all 15 members of the population for the last 10 epochs of training, and averaged over 100 repetitions of the experiments. In addition to the average weighted distance metric, another metric counting the proportion of agents at the goal location in the final timestep is also reported.

5 Results & Discussion

The results in Figure 4 suggest that flock performance improves and becomes more consistent as the number of leaders increases, and this relationship is nonlinear. In the simple test environments without obstacles, the leaders appear to converge to a stable policy within the first 20 or so epochs, regardless of the ratio of leaders. In the presence of obstacles, however, learned policies do not appear to converge so quickly: Figure 5 shows continued performance improvements well after 100 epochs.

The experiments with neural network parameterization in obstacle-filled environments bring up several interesting results. First, note that the inclusion of swarm centroid as an input to the neural network controllers does not appear to generate an improved leader policy. In the 20% leader case, the leaders trained without swarm centroid observability actually appear to perform slightly better, and there is not an observable difference between the two methods in the 40% leader case.

A second intriguing result is that neither variety of neural network controller performs as well as flocks with simple informed leaders, in the 20% leader case. When 40% of the flock act as leaders, however, the agents are able to collectively surpass the strategy of simple informed leaders. Based on our results, it appears that a critical mass of at least 30-40% leaders is required in order to observe improvements in flock performance. These leader density values are much higher than anticipated prior to conducting experiments. We had hoped that this threshold would drop to a lower leader density as the size of the swarm increases, but the results summarized in Figure 5 suggest that there is still a considerable performance improvement by moving from 20% leaders to 30% leaders. The median values of both performance metrics reported in Figure 6 are similar for swarms of 50 and 100 agents with leader densities ≤ 20%, and the 100-agent swarms only have noticeably better median values for both metrics with 30% leaders.

On the whole, the results of this project suggest that the learning method and parameterization employed here do not fully capture the complexity of interactions between leaders and other members of the swarm. Biological systems are able to accomplish migration and foraging tasks in complex environments when only a very small percentage of the group (under 10% [4]) has information about the goal, which implies that better leadership policies exist than the ones discovered by our neuro-evolutionary approach. The fact that the simple informed leader policy performed better with 20% leader density suggests that this policy has an advantage over the learning-based method; one possible explanation is that the simple informed leaders are more sensitive to feedback from naive agents because they incorporate Reynolds’ flocking rules into their heading policies.

6 Conclusion

In this paper, we proposed and implemented a learning-based control method for leader agents to guide a swarm towards a known goal destination. A neuro-evolutionary approach was used to map estimates of the swarm, goal, and leader state to a heading change policy for the leader agents. Training a caste of leaders, within an artificial swarm, to influence other members of the collective is a
relatively unexplored area. Experiments were conducted to investigate the relationship between neural network input parameters, environment complexity, and leader performance in a flock navigation scenario. The results suggest that the resulting leader policies are effective in this domain when at least 30–40% of the agents in the swarm are leaders, which is higher than anticipated.

Future work will focus on investigating the gap between the simple informed leader policy and the learning-based leader policies. In particular, we will explore the importance of reciprocal influence in communication between swarm agents and leaders, by designing a leader policy that combines Reynolds’ flocking rules with the heading vector produced by the learning-based method. The learning methodology itself may also be a potential source of improvement. The neuro-evolutionary algorithm employed in this project may be converging to local minima in leader policies, and a particle swarm optimization method may locate better policies by promoting higher diversity among neural network populations [18]. Other combinations of neural network inputs or outputs capturing the swarm and leaders’ states may be considered. In summary, the results presented in this paper confirm the validity of the general approach, but additional improvements are needed in order to generate effective leaders in artificial swarms.

References


