Flocking Behavior: Multi-Agent Q-Learning, Obstacle Avoidance and Real-World Simulations

Emily Cohen, Jonathan Gomes-Selman, Zane Kashner
Department of Computer Science
Stanford University
ecohen2@stanford.edu, jgs8@stanford.edu, zkashner@stanford.edu

1 Introduction

Our work is inspired by the "Boid" artificial life research program, first proposed by Craig Reynolds in 1986 [4]. This model seeks to capture the complex dynamics of collective motion exhibited in nature by flocks of birds, schools of fish, and other gregarious animals. Through studying the dynamics of such models, we can illuminate and better understand many complex animal behaviors, which can be directly applied to different real-world settings including dynamic multi-agent drones, robot swarms, quad-copters, and other robot systems.

Figure 1: Birds Flocking vs. Boid Simulation

Although drone swarms are typically associated with military activity, there are a wide range of applications for social good. For example, collective drone swarms can be used to monitor landscapes, such as farms, as well as survey war and disaster torn areas that would otherwise be inaccessible [1]. Additionally, drone swarms can be used to deliver food, medicine, and materials to remote or hard to reach areas without compromising the safety of more people than would otherwise be the case [1]. A lab at MIT has proposed using sailing robot swarms to clean up oil spills at an accelerated rate compared to current techniques [2]. Overall, it is clear that the collective power of a group greatly outperforms that of an individual and opens the potential for more effectively solving many real-world problems. The long-range goal of our project is to better understand the dynamics of these collectives, such as flocks, with the hope of being able to apply what we learn to increase the efficiency and feasibility of utilizing robot swarms to help better our surroundings in ways that humans are not able to physically accomplish.

Based on the Boid artificial life program for modeling flocking behavior, we study the use of machine learning techniques to learn flocking behavior. Rather than using traditional rule-based models to dictate the collective motion of flocking agents [2], we propose a reinforcement learning technique to learn this complex task. We then extend our model to capture other complex flocking behaviors, such as obstacle avoidance, and ultimately being able to apply our learned flocking behavior to the complex tasks of path following through obstacle mazes.

Our contributions: **(1)** We formulate the problem of **modeling complex flocking behavior.** The central challenge we face in this endeavor is fully defining an unstructured task. **(2)** We propose a **reinforcement learning approach** to tackle the problem, using Q-learning with a function ap-

CS221, Spring 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

proximation. We are tasked with determining relevant features for learning, as well as defining a local reward function to properly shape the learning process. In order to evaluate our results, we experiment with different **quantitative metrics** to measure the success of our flocking algorithm compared to standard approaches. **(3)** For training and testing our learning algorithms, we also **developed a simulation environment,** which captures the dynamics and uncertainty of the learning environment, keeps track of the state of the world, provides feedback about the local environment for each agent, and also allows us to visualize agent interaction dynamics. **(4)** We also provide a **uniform cost search** approach for planning the most efficient path towards the goal destination for the leading bird in an environment containing a series of obstacles (a maze). **(5)** We provide **detailed results showing that Q-learning with a function approximator can learn flocking and related behaviors at least as well as, and in terms of certain key measures, better than more traditional rule-based strategies.** These results indicate that reinforcement learning is a promising direction for the development of multi-agent flocking and swarming strategies. We also provide qualitative insights into learned flocking behaviors and into the **robustness** of the reinforcement learning strategy.

2 Related work

As discussed in the Introduction, the majority of the related work is in "Rule-Based" models such as the original Boid simulation [4]. Research on these models have primarily focused on rule analysis and fine tuning, as well as application of rule based systems to real world settings. Several methods have been proposed and investigated for measuring the success of Boid simulation models for flocking that we further explore in our work [5]. However, we were unable to find any related research that applied learning to mimic the collective flocking behavior, thus separating our research from previous work studying flocking algorithms.

We look to reinforcement learning as a general technique for agent based learning. At its core, reinforcement learning (RL) is concerned with how an agent ought to act in its environment in order to maximize its cumulative reward over a sequence of actions. RL is often applied to game playing. Although our proposed task is not a game, in many ways we can equate our central learning goals with that of a game; therefore, we draw inspiration from different learning techniques applied to game playing domains, specifically those involving an agent moving in an unstructured environment [7].

3 Task Definition

Our task is to model complex flocking behavior as a reinforcement learning problem for independent agents. Based on specific sensor inputs, action rewards, and quantitative measures of performance we look to meet and surpass the performance of rule based approaches traditionally explored in literature[4]. The traditional set of metrics defining flocking behavior are shown in figure 9 below.

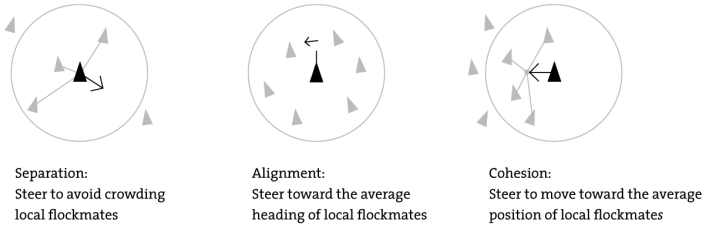

Figure 2: Flocking: Evaluation Metrics

Stated simply, a successful flocking bird (or agent) follows closely the birds around it and an identified leader while avoiding crashing into any other birds. Our goal is to learn such collective behavior in a reinforcement learning setting. In the Approach section below, we define in much greater detail the different state representations, rewards, and evaluation functions that we experiment with to actually model the task of collective motion.

We draw inspiration from research on existing flocking algorithms to define quantitative metrics of evaluation [5]. Specifically, we define metrics of evaluation for the three main objectives of a successful flocking system, where for Boid i, $B_i.loc$ represents its relative location and $B_i.angle$ its

angular direction. We define a standard axis for angular direction where north is 0° heading, East is 90°, south is 180°, and West is 270°. B_{center} captures metrics of the entire flock as:

$$B_{center}.loc = \left(\frac{\sum_{i=0}^{n} B_i.x}{n}, \frac{\sum_{i=0}^{n} B_i.y}{n} \right) \qquad B_{center}.angle = \frac{\sum_{i=0}^{n} B_i.angle}{n}$$

Cohesion: Average distance to the centroid of the flock across all birds at time t.

$$\frac{1}{n} \sum_{i=0}^{n} \|B_{center}.loc - B_i.loc\|_2 \tag{1}$$

Separation: Number of crashes at time t, where a crash is defined as a bird being within a given radial distance of another bird (30 units).

Alignment: Average difference between angular direction of each bird and the entire flock at time t.

$$\frac{1}{n} \sum_{i=0}^{n} |(B_{center}.angle - B_i.angle)| \tag{2}$$

All metrics are for a given time t. Additionally, note that we cannot use the angle of direction directly because we are comparing direction of motion, meaning that going in the opposite direction or having an angle difference of 180° is the largest difference that we want to consider. Therefore, we scale the heading angle as follows

$$B.angle' = \begin{cases} B.angle & B.angle < 180 \\ 180 - (B.angle \ mod \ 180) & otherwise \end{cases} \tag{3}$$

Although we utilize these quantitative metrics for precise evaluations of the learning success, we also consider the qualitative performance of our flock in the simulated environment. An example evaluation compares both the quantitative metrics defined above and the qualitative success of the flock seen through observing our simulation through time (Figure 3).

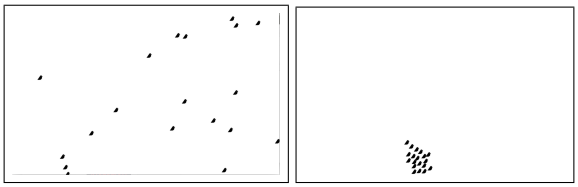

Figure 3: Flocking: before and after

In addition to the general flocking behavior, we explored the challenge of applying reinforcement learning in the context of following a single knowledgeable agent through a maze, which is representative of a more "real-world" experience. For this additional task, evaluation criteria include optimizing to avoid collisions with obstacles while following the leader closely and not getting lost. We use a similar technique to that described above by defining sensor inputs and rewards for our reinforcement learning agent (defined concretely later). While the majority of our results can be obtained by visually evaluating the simulation, we measure success through analyzing a cumulative reward function that we define in detail in the description of our approach.

4 Approach

4.1 Game Simulation

As a test harness and training environment, we implemented a full simulation environment from scratch, using Python and the Pygame module. This simulation environment can be run in headless mode to return states and rewards as well as display flocking agents in real time. While, in the end, this simulation environment is only used as a framework for our experiments, getting a simulation up and running was a significant starting task.

4.2 Proposed Method

4.2.1 Q-Learning

We will use Q-learning to learn the behavior of an individual member of the flock, which can then be applied in a multi-agent setting to mimic flocking motion. Q-learning is a good fit for this task because the task domain can be modeled as an MDP where a given bird must make a decision based on its current state and can learn appropriate actions by attempting to maximize its expected reward over time in the simulated environment. Our method applies Q-learning to learn the behavior of a bird (boid or "agent"). The agent's state is a representation of the sensor data that it receives about itself and its surrounding. Moreover, the expected reward for a given boid reflects the goal of following its neighbors closely while avoiding crashing into any other boids. We will explain in detail both the state representation and reward function later in this section.

In Q-learning, the goal is to learn a function that given a state, s, and an action, a, returns the Q-value (or the expected reward) of taking that action in the state, while subsequently following the optimal policy; i.e., maximizing the (discounted) future reward. The Q-value function can be used by the agent to take optimal actions simply by considering all possible actions in the current state and selecting the action with the highest Q-value. More concretely, we will try to estimate the Q-value for an optimal policy given by

$$Q_{opt}(s,a) = \sum_{s'} T(s,a,s')[r(s,a,s') + \gamma \max_{a' \epsilon A} Q_{opt}(s',a')] \tag{4}$$

where $T(s,a,s')$ is the probability of transitioning to state s' from state s when taking action a, $r(s,a,s')$ is the reward associated with this transition, and γ is the discount factor, which accounts for how much we discount the value of future rewards ($0 \leq \gamma \leq 1$). For each training step, we set the discount factor determining the importance of future rewards to $\gamma = 0.75$.

Traditional Q-learning tries to learn the Q-value for every independent state action pair. However, this approach often leads to issues in generalization when the state and action space is quite large, as the learning algorithm is unable to handle unseen states. Therefore, because of the large, continous state space of our problem (details below), to ensure a manageable representation of the Q-value function, we will define our states with a set of features and functional approximation to parameterize $\hat{Q_{opt}}$ as:

$$\hat{Q}_{opt}(s,a;w) = w \cdot \phi(s,a) \tag{5}$$

where $\phi(s,a)$ is a vector consisting of the feature values for a given state action pair, and w is a weight vector used to compute a weighted linear combination of those features. From this formulation, we define our training objective function for every (s,a,r,s') as:

$$L(w) = \frac{1}{2}(\hat{Q}_{opt}(s,a;w) - (r + \gamma V_{opt}(s')))^2 \tag{6}$$

where, $V_{opt}(s')$ is the expected reward when executing the optimal policy from state s' onwards. In other words, it's $Q_{opt}(s',a)$ for the action a in state s' that maximizes this value. We can now apply stochastic gradient descent on each (s,a,r,s') tuple to update the weight parameters w:

$$w \leftarrow w - \eta[\hat{Q}_{opt}(s,a;w) - (r + \gamma \hat{V}_{opt}(s')] * \phi(s,a) \tag{7}$$

where η is the learning rate. We use a variable learning rate $\alpha = 0.1/\sqrt{t}$ for learning step t.

A remarkable aspect of the weight update equation for learning the Q-values is that the learning agent does not need to know explicitly the state transition probabilities or the reward function. The agent will perform a so-called ϵ-greedy exploration/exploitation strategy, while updating the weight vector after each state transition using information returned by the simulator (the environment). Note that in a multi-agent setting, the agent has no explicit model of state transition probabilities for its actions because those transitions depend in part on the actions of the other agents. In an ϵ-greedy strategy, the agent takes with probability $(1 - \epsilon)$ the action with the highest expected reward according to the current estimate of the Q-value function ("exploitation"), and with probability ϵ, the agent takes a random action ("exploration") to further explore its environment.

A key issue in the Q-learning approach is the definition of the feature vector and the reward function. We will now discuss those in our setting.

4.2.2 Features

In Q-value function approximation, we must define a feature extractor for a given state and action pair, $\phi(s, a)$. Our general approach leverages hand labeled features to represent meaningful information about the environment of a learning bird, such as the location of the bird itself, the location of the leader, and neighboring objects.

When learning the flocking behavior for an individual bird, we define the state for that bird as follows: 1) The location of the bird in space, 2) the angle that the bird is flying (relative to the a central coordinate system), 3) the location of the leader that is leading the flock, 4) the angle that the leader is flying, and 5) the location of the other birds in the flock. For a given state, the actions that the bird can take are defined as a series of angular adjustments and velocity changes: $[-45°, -35°, -20°, -10°, -5°, -2°, 0°, 2°, 5°, 10°, 20°, 35°, 45°]$ and [-.3, -.2, -.1, 0, .1, .2, .3], where the velocity of the bird is bounded between [0, 3].

Given a state and action pair, we define the feature extractor to capture the relevant information needed for the bird to learn flocking behavior. The feature extractor considers four primary features: `Crash-bird1`, `Crash-bird2`, `leader-delta`, and `centroid-delta`. The features `Crash-bird1` and `Crash-bird2` are very similar. First we consider all the birds that fall within the 'too-close' radius around the learning bird (30 units). `Crash-bird1` is a feature that takes the value of 1/distance(B_x) where B_x is the closest bird that falls within this radius (if one exists). `Crash-bird2` takes the value of 1/distance(B_y) where B_y is the second closest bird that falls within this radius (if one exists). In both cases, the feature take on values of zero if a bird matching the feature criteria does not exist (e.g. `Crash-bird2` = 0 if there is no second closest bird that falls within the crash radius). These features are used to discourage the learning bird from getting too close to its neighbors. `Leader-delta` represents the change in distance to the leader bird after the given action is taken. `Centroid-delta` represents the change in distance to the centroid of the flock after the given action. These features place importance on the task of following the leader as well as being aware of the center of the flock to remain a collective unit.

It is important to note that these features do not involve any absolute coordinates or angles, and can be obained *locally* by each bird. In particular, the bird's perceptual system should be able to obtain estimates of the distance to closeby birds and the leader. Moreover, the bird's actions involving turning a certain angle are relative its current body orientation. (Again, the bird does not need to know its orientation in absolute terms.)

For the task of following a leader through a maze we define a slightly different feature extractor. This task occurs in a domain in which there are certain locations that are designated to be obstacles which the boid seeks to avoid colliding with. Within the feature extractor, we allow for the bird to know if it's colliding with one of these obstacles. Other than this, the other features are exactly the same.

4.2.3 Rewards

Another important aspect of Q-learning is defining an appropriate reward function to shape the learning process. This reward function provides a measure of 'success' to the learning algorithm, providing a signal (often quite weak or noisy, and possibly many steps delayed) about how positive or negative a particular action was.

For learning the flocking behavior, we define the reward for a particular action as follows. Our reward function considers the state of the world before and after a particular action taken by the learning bird. Firstly, it is important for the birds to avoid crashing into each other. We therefore give a fairly high penalty for a crash: -20 if the bird is crashing into one other bird, and -40 if it is crashing into two or more birds. Next, we look to provide incentive to go toward the center of the flock. Therefore, we add to a reward of 2, if the bird moves toward the flock's centroid; otherwise 0 reward is added. To encourage the bird to pay close attention to the leader, a bit more so than the flock itself, we give a reward of 5, if the bird takes a step closer to the leader; otherwise, we penalize the bird with a negative reward of -1. The total reward is calculated as a sum of the individual rewards.

As we will see below, this fairly basic reward scheme already enabled our Q-learning approach to learn overall good flocking behavior. We also found that the performance is not that sensitive to the actual values in the reward scheme.

For the maze we use a slightly adapted reward scheme that takes into account avoid obstacles. Firstly, we let 'dist(s)' stand for the Euclidean distance to the leader bird in state s. We heavily penalize

running into an obstacle with a negative reward of -1000. We assign negative reward of $-2/dist(s)$ if the follower steps too close to the leader (within radius of 30) or $-dist(s)/2$ if the follower does not follow the leader (i.e. by stepping in the wrong direction). Lastly, we give a positive reward of $dist(s)/8$ if the follower correctly follows the leader by stepping closer or staying the same distance.

4.2.4 Implementation and Learning Strategies

Given the features capturing the bird's perception of its environment and the rewards resulting from the bird's actions, we can now evaluate the extent to which we can Q-learn various behaviors. To learn the flocking behavior, we propose the following learning strategy. We introduce a leader bird (B), which travels in a random but smoothed flight path at a constant speed. Next, we introduce a second bird (F) that has been trained (Q-learned) to follow the leader at a close following distance of around 30 units. (Note: we first train this bird to follow the leader using simplified procedure to what we describe here). Lastly, we include a third bird (L), which uses Q-learning to learn the flocking behavior by learning to follow the leader while staying near the second bird but without colliding with either the leader of the other bird. We initially hypothesized that training an individual agent how to fly in a flock would be sufficient for learning the collective flocking behavior. As we demonstrate later, applying a Q-learned behavior for a single flocking bird to a large flock of independently acting birds results in the desired flocking behavior.

5 Results and Analysis

5.1 Evaluation and Experimental procedures

Our evaluation metrics measure the success of our reinforcement learning agents to learn flocking behavior from sensory input of their environment. We also consider other challenges, such as learning to follow a leader through a maze while avoiding colliding with other birds and objects ("walls"). In our analysis and performance evaluation, we incorporate both a *quantitative* evaluation, based on metrics described in the Section 3, Task Definition, and *qualitative* observations of the different learned behaviors in our boids virtual world.

5.2 Experiments and Quantative Evaluation

5.2.1 Following the Leader and Feature Comparison

Our experimental setup consists of an implementation of the Boid Q-learning algorithm with features and rewards described in Section 4 coupled with a GUI for simulation and visualization. As a first task, we consider the problem of learning to follow a "leader" Bird, where the leader has different flight-paths precoded (such as a circular flying path and a random smooth path). In this basic, our goal was to validate our Q-learning approach and consider the effect of different features on the learning performance.

For our first training, we began with a feature extractor that included the change in distance to leader, as well as a binary feature about whether the bird was too close or not. For our second iteration, we added the inverse distance to the leader, as well as changing to non-binary if the bird was too close. We were able to compare the stability of running Q-learning using both of those two feature extractors. You can see these results in Figure 4. From the figure, we see that both feature sets lead to successful following behavior. However, adding extra information in terms of a non-binary feature (based on inverse distance) when a bird is too close leads to a more stable learning curve.

This experiments highlights the importance of studying the feature selection in Q-learning, especially when using a linear Q-value function approximator. Note that even though the approximator captures a linear function of the features (a weighted sum), non-linearities in the dynamic domain can still be represented by the feature values themselves.

We now compare the behavior generated by our Q-learning approach with the behavior of a handcoded rule-based boid following a leader. Our handcoded approach follows the strategies discussed in the literature [3,4,5].

In our evaluation, we compare the behaviors with three metrics: `Good Follows`, `Ideal Distance`, `Crashes`, and `Alignment`. We defined `Good Follows` as the number of time steps when the bird took a step closer to the leader. We define `Ideal Distance` as the number of time steps where the

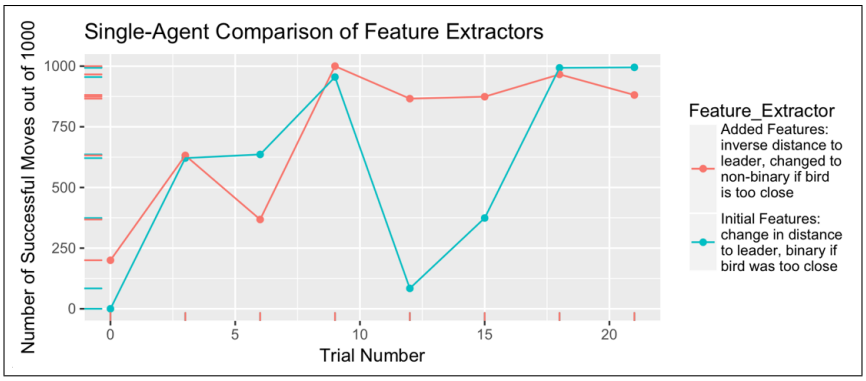

Figure 4: Comparing Features

bird was within a distance of 28 units and 40 units from the leader. Intuitively this metric measures the number of time steps that the following bird followed within the "ideal" follow radius. `Crashes`, is defined as the number of times that the boid "crashed" into the leader. Crashing in this context is defined as being within a distance of 28 units to the leader. Although when training we define the crash radius as 30 units, we relax the crash distance to simulate teaching the learning algorithm to air on the safe side. Lastly, we define `alignment` as the average difference in alignment between the leader and the follower. Specifically, the difference in heading alignment between the follower F and leader L is calculated as:

$$|F.angle' - L.angle'| \tag{8}$$

As discussed in Section 3 (Task Definition) we do not use the angle of direction directly, but instead used a modified metric for comparing heading direction.

Type	Steps	Good Follows	Ideal Distance	Crashes	Alignment
Learned	5000	2539	4773	134	24.1
Rule-Based	5000	2695	2956	68	25.9

Figure 5: Results for Q-learning to follow the leader

Figure 5 shows our results. Overall, we see in terms of `Good Follows`, Q-learned behavior does as well as hand-coded rules. In fact, Q-learning is better in terms of number of `Ideal Distance` steps and the `Alignment` angle. Q-learning does have a somewhat higher crash risk. This can be explained by the fact the other measures show that the Q-learned boid is flying more closely to, and more aligned with, the leader. This results in a somewhat higher risk of a crash. Increasing the penalty for crashing would counter this phenomenon. Overall, we see that the follow-the-leader task can be learned well using the Q-learning framework.

5.2.2 Flocking Behavior

After successfully training a single bird to follow a leader, we consider the more challenging task of training a bird to act as a member of a flock. As described in Section 4.2.4 (Implementation), we experimented with training a bird to follow a leader while exhibiting flocking behavior. To briefly recap, the learning bird is trained in relation to another bird that follows the behavior learned in the follow-the-leader setting (discussed in the previous section) and a leader bird. Using the metrics described in Section 3 (Task Definition), we evaluate the success of training a bird to learn the flocking behavior using Q-learning and compare that to a rule-based bird acting in the same setting over 5000 time steps. Note that in this setting, the flock size is technically three because we include the leader and the follower. We will consider next the behavior of larger flocks.

Multi-Agent (flock size 3)	Q-Learn	Boid Rules
Cohesion	22.01	24.98
Separation	0.0	0.0
Alignment	24.58	20.37

Figure 6: Comparing behavior of training single flocking bird to Boid rules

Figure 6 shows the results. (0.0 for `Separation`, because there were no crashes over the 5000 time steps.) These results show that the Q-learning bird is able to learn the flocking behavior very well. We see that acting as a member of a flock of 3 birds the Q-learn bird performs at the same level as the bird controlled by the Boid rules in all three metrics. We see that the both birds avoid crashing ever crashing into the other two members of the flock, showing that in a small flock the bird "understands" how to follow while maintaining a safe distance. Interestingly, we also note that the alignment scores are quite close with the rule based bird doing better by 4 degrees. Although our features and rewards do not take into account the relative alignment of the birds, somewhat incredibly the bird is able to learn to remain roughly aligned based on its expected rewards from following the leader and attempting to stay near the center of the group. One potential explanation for this is that by learning to keep good cohesion while following the leader's path, naturally the bird will be generally aligned with the overall flock. Nevertheless, it is exciting to see that this particular behavior appears learnable without direct knowledge of it during training.

As hypothesized, when we apply this learned behavior to a "flock" of independently acting birds the birds exhibit fairly robust flocking behavior, comparable to rule-based flocking (even better in some respects). Applying the exact learned behavior from the prior experiment to 15 flocking agents we obtain the following results when compared to 15 rule based agents (flock size 16).

Multi-Agent (flock size 16)	**Q-Learn**	**Boid Rules**
Cohesion	54.768	51.347
Separation	0.164	0.258
Alignment	55.46	42.680

Figure 7: Comparing behavior of independently acting Q-learned birds and rule-based birds

The results demonstrate that the Q-learning algorithm is able to capture the flocking behavior quite effectively. The birds following the Q-learned strategy perform at roughly the same level for cohesion, as a difference of 3 units is roughly the distance covered by one step of a bird. The Q-learned flock performs a bit better than the rule-based model in the separation metric, averaging 0.164 crashes per time step compared to 0.258. This indicates that the learning algorithm learns to be more cautious about avoiding collisions. The tradeoff in cohesion and separation for the Q-learning flock may reflect a safer strategy compared to the rule based birds that place more value in remaining tightly knit as a flock. Lastly, we see that when applying the learned behavior to a large flock of independently acting agents the learning algorithm no longer matches the performance of the rule-based model in the alignment metric. This is not too surprising considering our feature and reward models do not incorporate relative alignment of the flock. While we were able to learn to remain aligned in a small flock, when applied to a much larger flock our learning algorithm has not learned yet the importance of remaining aligned with the flock. It may well be that longer training and smaller learning rate would help uncover the benefits of alignment. This is an interesting question for future work.

5.2.3 Obstacle Avoidance

We implemented two mazes, see Figure 10, for a boid to learn how to maneuver through following a leader. This leader uses Uniform Cost Search (UCS) to plan a most efficient path through the maze. Through 100 trials of 1200 iterations (time steps) each, we learned weights for the features indicating if it was too close, its change in distance, overall distance, as well as if it was too close to the wall. These weights were learned on the train maze, and used for the test maze as well.

In order to evaluate this learned behavior, we compare the average total reward as defined in the task definition over twenty runs through the maze. We decided that since we seek to maximize the reward function, the best way to define the ability of maneuvering through this maze would be the total reward accumulated when moving around. Thus, we define the total reward to be *total-reward* = $\sum_{i=1}^{num-steps} reward(s_i, s_{i-1})$. As a baseline, we collected this metric for a pair of boids following the boid rules. We compared these results with that of the boid following the Q-learned behavior. Because the only time that a positive reward is possible is when we move closer, and to move closer consistently, we also have to move further it is hard to evaluate exactly what the highest possible performance would be. However, if the boid was consistently moving closer incrementally near the border the reward would be *num-iters*(border-dist*/8) = $1200 * (30/8) = 4500$. However, more realistically, an ideal strategy would be to stay exactly on the border of too close, which would result

in a total reward of 0. We found that the Q-learning algorithm vastly outperforms a boid rotely following the hard-coded rules, as is obvious in Figure 8.

Maze	Q-Learn	Boid Rules
Train	-3792.56	-592591.31
Test	-3870.33	-655115.11

Figure 8: Comparing total rewards through a maze

5.3 Qualitative Analysis

For our qualitative analysis, we implemented a graphical user interface to observe the actions of the flocking birds . The leader and the learner have different icons, making it obvious which is which. We are able to use this to observe the ease with which the bird follows the leader and qualitatively measure success of learning.

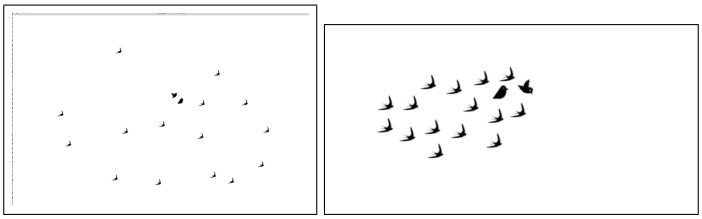

Figure 9: Flocking: before and after

Qualitative analysis is especially useful for analyzing the success of flocking agents. As shown in Figure 9, qualitative analysis of the flocking behavior of a multi-agent system can inform about many aspects of the flock that are much harder to quantify in numerical metrics, such as general flock shape, stability of the flock, and odd behaviors amongst learned agents. One really interesting behavior observed qualitatively was an unexpected behavior expressed by flocking agents where every so often a group of two or three birds would all of a sudden head in the complete opposite direction from the flock before turning back. It appears that this phenomenon is caused in part because in our evaluation runs, we pick the best possible move according to the learned Q-value function (exploration is done during learning). However, it appears that in certain rare configurations, this may lead to rather unexpected behavior. To resolve this issue, we decided to use an epsilon greedy algorithm ($\epsilon = 0.2$) also during evaluation (after learning). Doing shows that the introduction of slight randomness into the movement of the birds actually made the flock more robust because it helped prevent certain birds from getting "stuck" in an unfavorable action sequence. The observed phenomenon is example of unexpected behavior that can arise in reinforcement learning, which is a current concern in the AI Safety research community. Introducing randomization may be a useful counter strategy.

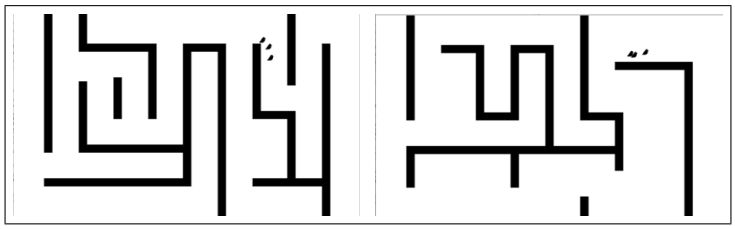

Figure 10: Follow leader through a maze

We also implemented a graphical representation of the mazes that we created. Using this, we can see – as in Figure 10 – how the boids following the leader can navigate the maze and follow at an appropriate distance while avoiding obstacles. The ability of a boid to follow through a maze is immediately apparent when watching these simulations. The boid can effectively keep a buffer distance between them and the leader, while effectively maneuvering through the maze and avoiding running into walls.

6 Conclusion and Future Work

6.1 Conclusion

The goal of our work was to consider whether reinforcement learning can be used to obtain flocking behaviors in multi-agent settings. The first computational studies on flocking behavior was done as part of the "Boids" research program on Artificial Life [1]. Most work in the area has considered various types of hand-coded rules. With the incredible recent advances in reinforcement learning, a reasonable question is whether such behavors can be learned.

Our results show that we can learn behavior for boid agents that can match or outperform more traditional rule-based approaches for flocking behavior. In particular, using Q-learning with function approximation, boids were able to follow a leader effectively, follow a leader through a maze, and move as a cohesive group. All of these behaviors were learned based upon basic features and local rewards (the features can be obtained via sensory input from the boid's environment), and encode the same amount of environment information as was available to the rule-based boids.

We found that a single boid following a leader over 5000 steps was able to stay within the ideal range of separation 61.5% more of the time than its rule-based counterpart. Additionally, when that boid was trained within a domain containing obstacles, it significantly outperformed a boid following the hand-coded rules (average overall total reward -3870 vs. -655115). Lastly, a flock of boids that moves according to learned behavior was somewhat more cohesive, with fewer crashes than their rule-based counterpart. Moreover, for all three of these behaviors, we observed in the graphical simulation that the learned behavior was far "smoother" with much less sudden accelerations and decelerations. We also compared different types of features and local rewards. In general, more basic features and rewards led to the most robust performance.

As discussed in our results section on Q-learning for flocking, we also observed that injecting some randomization in the final learned policy can improve robustness of the flocking strategy, and can avoid some unexpected behaviors. This may be a good general strategy to minimize unexpected behavior in a reinforcement learning setting, a problem recently addressed by the AI Safety research community.

Overall, these results show that Q-learning provides an effective learning paradigm for studying flocking and more general swarm robotics.

6.2 Future Work

Our results show that the use of reinforcement learning in studying flocking behavior or general robot swarm control is a promising area for further research. There are many directions for future work. A first question would be to study the learnability of more complex flocking behaviors, e.g., a flock streaming around an obstacle by splitting up and then merging together again. Also, a comparison with different flocking behaviors in biological systems would be of interest. A comparison with the effectiveness of a deep Q-Learning approach on the tasks we considered would shed light on the effectiveness of the Q-value function approximation in this context. It would also be interesting to study the robustness of learning under various noise models for the sensory inputs to the boids.

References

[1]"Drone Swarms Could Be Lifesaver in Disasters". NBCNews. 25, Jan. 2014. https://www.nbcnews.com/tech/innovation/drone-swarms-could-be-lifesaver-disasters-n15906

[2] "A Swarm Of Robots To Clean Up Oil Spills. MIT School of Architecture and Planning". Dec. 2012. https://sap.mit.edu/article/standard/swarm-robots-clean-oil-spills

[3] Sebastian. "Flocking Like It's 1999". http://coffeepoweredmachine.com/flocking-like-its-1999/

[4] Craig W. Reynold's home page, http://www.red3d.com/cwr/

[5] Kattas, Graciano Dieck, Xiao-Ke Xu, and Michael Small. "Dynamical modeling of collective behavior from pigeon flight data: flock cohesion and dispersion." PLoS computational biology 8.3 (2012): e1002449.

[6] Tan, Ying, and Zhong-yang Zheng. "Research advance in swarm robotics." Defence Technology 9.1 (2013): 18-39.

[7] Szita I. (2012) Reinforcement Learning in Games. In: Wiering M., van Otterlo M. (eds) Reinforcement Learning. Adaptation, Learning, and Optimization, vol 12. Springer, Berlin, Heidelberg