Consensus, Cooperative Learning, and Flocking for Multi-agent Predator Avoidance

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Engineering

by

Zachary Young

Dr. Hung M. La - Thesis Advisor
May 2020
THE GRADUATE SCHOOL

We recommend that the thesis prepared under our supervision by

Zachary Young

entitled

Consensus, Cooperative Learning, and Flocking for Multi-agent Predator Avoidance

be accepted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Hung M. La, Ph.D.
Adviser

Sushil Louis, Ph.D.
Committee Member

Jun Zhang, Ph.D.
Graduate School Representative

David W. Zeh, Ph.D., Dean
Graduate School

May, 2020
Abstract

Multi-agent coordination is highly desirable with many uses in a variety of tasks. In nature the phenomenon of coordinated flocking is highly common with applications related to defending or escaping from predators. In this thesis a hybrid multi-agent system that integrates consensus, cooperative learning, and flocking control to determine the direction of attacking predators and learn to flock away from them in a coordinated manner is proposed. This system is entirely distributed requiring only communication between neighboring agents. The fusion of consensus and collaborative reinforcement learning allows agents to cooperatively learn in a variety of multi-agent coordination tasks, but this thesis focuses on flocking away from attacking predators. The results of the flocking show that the agents are able to effectively flock to a target without collision with each other or obstacles. Multiple reinforcement learning methods are evaluated for the task with cooperative learning utilizing function approximation for state space reduction performing the best. The results of the proposed consensus algorithm show that it provides quick and accurate transmission of information between agents in the flock. Simulations are conducted to show and validate the proposed hybrid system in both one and two predator environments resulting in an efficient cooperative learning behavior. In the future the system of using consensus to determine the state and reinforcement learning to learn the states can be applied to additional multi-agent tasks.
Acknowledgments

I would like to thank my advisor, Dr. Hung La, who provided me with the guidance and assistance needed for my research. Without him and his advice this thesis would not have been possible. I would also like to thank my committee members Dr. Sushil Louis and Dr. Jun Zhang for their advice and time taken to review this thesis. Lastly a thanks to my family who made my education possible.

This material is based upon work partially supported by the National Aeronautics and Space Administration (NASA) Grant RRR No. 80NSSC19M0170, and Grant No. NNX15AI02H issued through the NVSGC-RI program under sub-award No. 19-21, the RID program under sub-award No. 19-29, and the NVSGC-CD program under sub-award No. 18-54.
# Table of Contents

1 Introduction .................................................. 1
   1.1 Motivation ................................................. 1
   1.2 Literature Review ....................................... 2
      1.2.1 Flocking Background ............................... 2
      1.2.2 Reinforcement Learning Background ............... 3
      1.2.3 Consensus Background ............................. 3
   1.3 Contributions ............................................ 4
   1.4 Thesis Organization ..................................... 5

2 Flocking Control .............................................. 6
   2.1 Flocking Control ......................................... 6
   2.2 Results of Flocking Algorithm ......................... 8
   2.3 Summary .................................................. 10

3 Multi-agent Learning .......................................... 11
   3.1 Learning Model .......................................... 11
      3.1.1 State ................................................. 11
      3.1.2 Action ................................................ 12
      3.1.3 Reward .............................................. 12
   3.2 Cooperative Learning ................................... 13
List of Tables

4.1 The iterations to converge and number of incorrect directions found over 900 iterations. $c_m = 2$ produces the best results. .......................... 25
List of Figures

1.1 Block diagram of the hybrid system. ........................................ 5

2.1 50 agents flocking to the green target. ..................................... 9

2.2 50 agents flocking to the green target while avoiding the red obstacle. 10

3.1 Visualization of the direction classifications. The large red circle represents the predator while the triangle represents the agent. The blue circle represents the best target, green circles are good targets, yellow circles are average targets, orange circles are bad targets and the small red circle is the worst target in this scenario. ........................................ 13

3.2 Comparison of convergence between RBF and Q learning with and without cooperative learning. All four algorithms were run 10 times for 20 episodes and the results were averaged. Over the course of these runs cooperative RBF was able to converge within four episodes while cooperative Q learning was unsuccessful in fully converging within 20 episodes. Both independent algorithms were not able to learn. ................ 19

4.1 Direction and associated information values for an agent observing a predator. Each color corresponds to an information value 1 through 8 with 1 representing east, 3 representing north, and so forth. .................. 22
4.2 Predator moving in circle around flock with consensus updating the
state of the agents based on the direction of the predator. . . . . . . . 25

4.3 The number of agents in the same state (the same $info_i$) for each iter-
ation. For all positions of the predator the agents are able to converge
to the same state. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 26

4.4 The number of consensus iterations for all agents to converge to the
same state. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 26

4.5 The average $info_i$ for the agents in each iteration. Some states occur
for longer durations due to the shape of the flock. . . . . . . . . . . . 27

4.6 The number of agents in the same state when $c_m$ is infinite. There are
multiple predator positions that result in the agents not fully converg-
ing even given 40 iterations. . . . . . . . . . . . . . . . . . . . . . . . 27

4.7 The convergence of the agents state when $c_m$ is infinite. Given 40
consensus iterations it is not able to fully converge. . . . . . . . . . . 28

4.8 The cause of error of the consensus algorithm. Despite the predator
being east of the flock as a whole the only agent in range senses it as
north east causing the error seen in Tab. 4.1. . . . . . . . . . . . . . 28

5.1 Initialization of an episode with agents randomly distributed. . . . . . 32

5.2 The number of agents choosing the same action for each episode for a
single direction state by episode. The agents are able to fully converge
by 4 episodes. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33

5.3 The number of agents choosing the same action for each episode for
a single direction state by iteration with the dashed lines representing
the start of an episode. It can be seen that most learning is done in
the beginning of an episode while the flock is connected. . . . . . . . 34
5.4 The number of agents choosing the same action for each episode for a single predator environment averaged over 6 runs. It can be seen that the agents converge by each direction being encountered 6 times.

5.5 50 agents flocking away from one predator before learning the same target.

5.6 50 agents flocking away from one predator after learning the same target.

5.7 Trajectory the agents take in the last episode where the triangles are the initial position of the agents. The Flocking can be seen to be smooth to the target after agents get into flocking formation despite the \( \epsilon \)-greedy random action selection.

5.8 Number of agents choosing the same action for two predators over 216 episodes. It can be seen that after 180 episodes all agents choose the same action for each state.

5.9 50 agents flocking away from two predators before learning the same target.

5.10 50 agents flocking away from two predators after learning the same target.
Chapter 1

Introduction

1.1 Motivation

Multi-agent cooperative learning has been and continues to be a large research interest in the field of robotics with a wide range of applications. Tracking wildfires by using multiple agents communicating together to better handle the fire so it doesn’t destroy as much is one such possibility [1][2]. Another is using multiple agents to better map the structure of a pipeline that if it structurally fails could cause large damage and loss of money [3]. An additional possibility is mapping and exploring unknown environments [4]. This thesis aims to solve the task of multi-agent predator avoidance [5][6] with an intelligent hybrid system.

Most current multi-agent research incorporates some form of consensus, movement control, or reinforcement learning but a combination of all three to achieve an efficient cooperative learning behavior is largely unexplored. Some uses of consensus are to determine the location of obstacles [7] or to make measurements in a scalar field [8][9][10]. Movement control of multi-agent systems can come in the form of cooperatively
doing path planning as in [11] and [12]. Alternatively there are means of control through flocking in varying formations [13] [14] [15] to achieve a variety of tasks [16] [17]. Reinforcement learning has been implemented cooperatively in a variety of ways for multi-agent environments such as a Grid World [18] [19] and box pushing [20]. Although all of these uses for consensus, movement control, and reinforcement learning are good in their own right, this thesis aims to make a more intelligent hybrid system.

In nature flocking has long been observed in many environments [21–23] with one possible goal being to defend from predators as can be seen with schools of fish. Even simulated environments [24,25] that reward individuals for their own survival result in flocking like formations of the agents. It is thus clear that flocking for survival has clear benefits in both natural and simulated environments. The goal of this thesis is then to create a hybrid system that combines consensus, flocking, and multi-agent reinforcement learning into one intelligent system that can sense and learn to escape from attacking predators.

1.2 Literature Review

1.2.1 Flocking Background

Methods of ensuring agents flocking have been proposed and studied in [26] [28]. Inspired by the natural world of birds and fish flocking together [29] flocking algorithms have been formed. The algorithms allow for agents to flock in different patterns in a distributed manner that requires on communication between direct neighbors rather than the entire flock. The main rules that define flocking are that flockmates maintain a distance from each other without getting to close or far from each other and
match the velocity of the other flockmates. Reinforcement learning approaches have also been applied to flocking in which agents individually or cooperatively learn to flock without the use of specific algorithms [30,31]. There are also cases in which reinforcement learning is implemented to teach agents to go to specific targets but in cases where learning fails to then use a flocking algorithm to avoid obstacles [32]. Flocking by itself is not particularly useful if the agents flock directly to a predator so this work aims to combine learning with flocking to effectively escape from predators.

1.2.2 Reinforcement Learning Background

Cooperation is an important part of a flock learning to do a task together. In [33] agents are not necessarily flocking together but through cooperation they are completing their task effectively. Cooperation is already a large part of flocking algorithms so that they can be done in a distributed manner. Cooperation must also be used to effectively learn in a distributed manner. Using simple Q-learning [34] does not achieve the necessary amounts of cooperation required so a more cooperative approach [6,35] is required. Unfortunately as the number of agents increases the state space grows and requires longer and longer amounts of training to effectively learn to flock together to the same destination. For this, function approximation techniques [36,37] are useful. However in current research function approximation in combination with cooperative learning is largely unexplored [38,39].

1.2.3 Consensus Background

Partial observability of the state space is another issue for purposes of escaping attacking predators. For a school of fish, not all fish will be able to see an attacking predator yet they manage to utilize flocking to maximize their safety anyway. For
our agents this is the same case in that only agents on the outside of a flock will be able to see an oncoming predator. A method of communicating to the other agents that a predator is approaching is thus necessary. This can be seen as a sort of event triggered consensus such as that proposed in [40]. Algorithms have been proposed to allow multi-agent systems to come to a consensus on measurements from multiple agents that do not necessarily agree with each other [41,42]. Many uses of consensus however are for estimating some kind of measurement [43]. This work intends to use consensus in combination with reinforcement learning to make a more intelligent system. Some works such as [38] and [39] implement a hybrid system of consensus and reinforcement learning but they use consensus to determine the global reward of the system. This work uses consensus to determine the state relying on local rewards instead.

1.3 Contributions

In this thesis we intend to combine the benefits of consensus, flocking, and reinforcement learning to create a hybrid system shown in Fig. 1.1. This system assumes partial observability in that only agents on the outside of the flock near an approaching predator are able to see the attacking direction of the predator. They must use consensus to inform the rest of the flock about the attacking direction of the predator which is then used with reinforcement learning to learn a target(a safe place) to move towards. That target is then used by a flocking algorithm to give each agent a control input to move each agent towards the target in a flocking formation. The contributions of this thesis are then as follows:

- Utilization of consensus between agents for state approximation in reinforcement learning.
• Cooperative learning with a large number of agents.

• Implementation of function approximation to reduce state space for the large number of agents used.

• Integration of reinforcement learning and flocking to learn where to flock.

• A fast and accurate consensus algorithm for a large number of agents.

• An entirely distributed system.

1.4 Thesis Organization

The organization of the remainder of this thesis is as follows. In Chapter 2 a method for flocking is introduced. Chapter 3 goes over the multi-agent learning used to ensure the agents flock together to the same target. Chapter 4 details a method of state approximation called consensus for the agents. This is followed by Chapter 5 which details the combination of flocking, reinforcement learning, and consensus into a hybrid system for the agents to learn to flock away from a predator. Lastly, Chapter 6 covers the conclusion with analysis and potential future development.
2.1 Flocking Control

In this section the flocking algorithm used for the hybrid system is presented. In order to learn to avoid predators the agents must be able to flock together. Using flocking methodologies presented in [26] a network topology consisting of a graph $G$ that is a pair $(V, E)$ with a set of vertices $V = \{1, 2, ..., n\}$ and edges $E \subseteq \{(i, j) : i, j \in V, j \neq i\}$. In this graph the robots are considered vertices and the edges are communication links between neighboring robots. Using robots modeled as particles the equations of motion are given by

\[
\begin{aligned}
\dot{q}_i &= p_i \\
\dot{p}_i &= u_i
\end{aligned}
\]  

(2.1)

where $q_i$ is the position of robot $i$, $p_i$ is the velocity, and $u_i$ is the acceleration or the control input.
The neighbors of an agent can be determined by

\[ N_i = \{ j \in V : \| q_j - q_i \| < r \} \]  \hspace{1cm} (2.2)

where \( \| . \| \) is the Euclidean norm and \( r \) is the interaction range of an agent.

There are many formations flocking can take but the formation used here is an \( \alpha \)-lattice formation in which

\[ \| q_j - q_i \| = d \quad \forall j \in N_i(q). \]  \hspace{1cm} (2.3)

for desired distance \( d \) where \( d = r/k \) for a scale factor \( k \).

In flocking each agent determines its control input with a gradient-based term \( f^g_i \) given by \[26\]

\[ f^g_i = c_1 \sum_{j \in N_i} \phi_\alpha(\| p_j - p_i \|_\sigma) n_{ij} \]  \hspace{1cm} (2.4)

where \( n_{ij} = \sigma_\epsilon(q_j - q_i) = (q_j - q_i)/\sqrt{1 + \epsilon\| q_j - q_i \|^2} \) and \( \phi_\alpha(.) \) is a pair-wise attractive/repulsive force to maintain the desired distance \( d \) between robots. With \( \sigma \)-norm, \( \| . \|_\sigma \) given by \( \| x \|_\sigma = 1/\epsilon[\sqrt{1 + \epsilon\| x \|^2} - 1] \) that is differentiable everywhere for \( \epsilon > 0 \).

An obstacle avoidance term given by \( f^o_i \) that is the repulsive force of \( f^g_i \) using points on obstacles as virtual neighbors \( N_i^o \) given by

\[ f^o_i = \sum_{j \in N_i^o} b_{ij}(q)(p_j - p_i). \]  \hspace{1cm} (2.5)

A velocity consensus term \( f^d_i \) given by

\[ f^d_i = c_2 \sum_{j \in N_i} a_{ij}(q)(p_j - p_i) \]  \hspace{1cm} (2.6)
where $a_{ij}(q) = p_h(||q_j - q_i||/||r||)$ is an adjacency matrix over the interval $[0,1)$ and $c_2$ is some positive constant. $p_h(.)$ is a bump function that smoothly varies between 0 and 1. One possible definition is given by

$$p_h(z) = \begin{cases} 1 & z \in [0,h) \\ \frac{1}{2}[1 + \cos(\pi\frac{z-h}{1-h})] & z \in [h,1] \\ 0 & \text{otherwise} \end{cases}$$

(2.7)

where $h \in (0,1)$. A navigational term, $f_i^\gamma$, that determines the direction the agents should be moving towards, given by

$$f_i^\gamma = -c_{1t}(q_i - q_t) - c_{2t}(p_i - p_t)$$

(2.8)

where $c_{1t}$ and $c_{2t}$ are positive constants, and $q_t$ and $p_t$ are the position and velocity of the target. These equations can be combined to find the control input for each agent $u_i$ given by

$$u_i = f_i^g + f_i^d + f_i^\gamma + f_i^o.$$  

(2.9)

This method allows the agents flock together in an $\alpha$-lattice formation towards a target location.

### 2.2 Results of Flocking Algorithm

The results of this flocking algorithm can be seen in Fig. 2.1 where the agents are flocking to the green circle without an obstacle and in Fig. 2.2 with an obstacle. The agents are initialized randomly over a 120x120 area and flock towards the green target. The blue lines represent communication links between agents. It can be seen that the
agents maintain their distance from each other without getting too far away from each other and eventually converging to an $\alpha$-lattice formation. In the case of an obstacle the agents manage to avoid colliding with it. This flocking algorithm thus provides a viable method of escaping from predators as well as provides a communication structure between agents to use for communications required to cooperatively learn.

Figure 2.1: 50 agents flocking to the green target.
Figure 2.2: 50 agents flocking to the green target while avoiding the red obstacle.

2.3 Summary

In this chapter a flocking algorithm was described that allows numerous agents to flock together to specified locations using a random position initialization. The algorithm enables agents to flock in an $\alpha$-lattice formation that is capable of avoiding collisions with obstacles on the way to the specified target though it slightly disrupts the connection of the network. For this reason the algorithm alone is not sufficient for avoiding predators so reinforcement learning is used to determine targets to go to that avoids getting near predators.
Chapter 3

Multi-agent Learning

In this chapter an entirely decentralized reinforcement learning method for a network to learn to flock together to specified targets is presented. Independent and cooperative learning methods are presented. In addition to this a method of cooperative learning with function approximation is evaluated against standard cooperative learning.

3.1 Learning Model

The model of the learning algorithm is similar to that proposed in [6]. Using a State, Action, Reward model for an agent $i$, let current state, action, and reward be $s_i$, $a_i$, and $r_i$ with the next state and next action as $\dot{s}_i$ and $\dot{a}_i$ respectively.

3.1.1 State

The state can be defined as $s_i = [\text{dir}_p, |N_i^a|]$ where $\text{dir}_p$ is the direction of a predator if detected and $|N_i|$ is the number of neighbors in range for agent $i$. The state $\text{dir}_p$ is
set to 1, 2, 3, 4, 5, 6, 7, or 8 for the directions east, northeast, north, northwest, west, southwest, south, and southeast respectively. The directions can further be divided into a larger space or smaller space if desired. In the case of multiple predators this state space can be expanded by adding a \( \text{dir}_p \) state for each predator.

3.1.2 Action

For actions the agents want to move in one of eight cardinal directions to escape a predator depending on the directions that the predator is coming from. These actions can be encoded as 1, 2, 3, 4, 5, 6, 7, and 8 mirroring the possible directions in the state defined above. The action list can then be defined as \( A_i = [1, 2, 3, 4, 5, 6, 7, 8] \). These actions interact with the flocking algorithm in that the actions are targets in the respective direction that the agents then flock towards if chosen. If no predator is detected the agents perform no action and stay where they are. The actions are represented as targets that an agent can choose to flock towards.

3.1.3 Reward

The flocking algorithm used provides flocking in an \( \alpha \)-lattice formation. This formation ensures agents on the inside of the formation have up to six neighbors while agents on the outside have one to five neighbors. To match this formation the reward is then defined as

\[
    r_i = \begin{cases} 
        |N^a_i| \cdot D_r & |N^a_i| < 6 \\
        6 \cdot D_r & \text{otherwise}
    \end{cases}
\]

so that the max reward that an agent can get is 6 if it has all six neighbors to encourage flocking.

The reward is then scaled depending on the direction of the predator. The scaling
Figure 3.1: Visualization of the direction classifications. The large red circle represents the predator while the triangle represents the agent. The blue circle represents the best target, green circles are good targets, yellow circles are average targets, orange circles are bad targets and the small red circle is the worst target in this scenario.

factor $D_r$ is split into five categories consisting of the best target, good targets, average targets, bad targets, and the worst target which can be visualized in Fig. 3.1. Agents choosing the action corresponding to the best target have their reward equal to the reward defined in (3.1). Actions corresponding to good targets are scaled down to 75 percent of the reward above, average targets to 50 percent, bad targets to 25 percent and the worst target to 0 percent. This is done to encourage the agents to learn the optimal target to go towards while maintaining the importance of flocking together.

The addition of more predators multiplicatively scales the reward. For example if there are two predators and an agent chooses an action that is a good direction for both of them the reward will be scaled down by 75 percent twice or 56.25 percent. This would fail if there are eight predators with one in each direction but in that case there is no safe space for the agents to go.

3.2 Cooperative Learning

In order to learn to flock to the same target together a cooperative learning method is implemented. Agents learning independently in this environment will take many learning episodes to converge or never converge at all which can be seen in [6]. However, to cooperatively learn each agent must first do independent learning [34] for an
individual table, $Q_i$, as follows:

$$Q_i^{k+1}(s_i, a_i) \leftarrow Q_i^k(s_i, a_i) + \alpha[r_i^k + \gamma \max_{\hat{a}_i \in A_i} Q_i^k(s_i, \hat{a}_i) - Q_i^k(s_i, a_i)]$$ (3.2)

where $\alpha$ is a learning rate and $\gamma$ is a discounting factor. This independent learning is not capable of converging in any reasonable amount of time for this application so cooperative learning must be used. After performing independent learning the $Q$-table of each agent is further updated by communicating with its neighbors using the following [6]:

$$Q_i^{k+1}(s_i, a_i) \leftarrow wQ_i^k(s_i, a_i) + (1 - w) \frac{\sum_{j=1}^{|N_i|} Q_j^k(s_j, a_i)}{|N_i|}$$ (3.3)

where $w$ is a weight such that $0 \leq w \leq 1$ to determine how much an agent should trust neighbors versus itself. It can be seen $w = 1$ would mean the agent only trusts itself and $w = 0$ would mean the agent only trusts its neighbors. The weight chosen can either be a static value or in this application the weight is defined as $w = \frac{1}{|N_i|+1}$ so that each agent equally trusts each other agent. Dividing the sum by $|N_i|$ is required so that over the course of the learning the Q-values do not converge to infinity too quickly. Note that the update from the neighbors is based on the neighbors state $s_j$ and the agents own action $a_i$

### 3.3 Action Selection

The action selection of an agent is based on the maximum Q-value approach [35] [44] in which the action with the highest Q-value for a given state is the action chosen. This method of choosing the action is highly exploitative with no exploration. To introduce exploration we use $\epsilon$-greedy [34]. We use a small probability $0 \leq \epsilon_g \leq 1$
in which to ignore the highest Q-value and instead select an action at random. This can be modeled as follows

\[
a_i = \begin{cases} 
a_{\text{max}} \in A_i & \epsilon_g < \text{random}(0, \ldots, 1) \\
a_{\text{random}} \in A_i & \text{otherwise}
\end{cases}
\]  

(3.4)

This random action selection allows an agent to explore a new action that might return a higher reward. The same action selection can be used for function approximation learning replacing the Q-value with the \( \theta \) parameter vector.

### 3.4 Function Approximation

Despite the cooperative Q learning algorithm preforming better than independent learning as seen in [6] it still can be improved upon to get better results and faster convergence. The direction of the predator \( \text{dir}_p \) is already discretized into eight directions however the number of neighbors \( |N_i| \) grows in size with the number of agents used. Due to the random initialization of agents at the start of each episode it is possible for each agent to be a neighbor of each other agent. However as the episode progresses and the \( \alpha \)-lattice formation is achieved this state will have one of seven values for either no neighbor or one to six neighbors. This state size is not particularly large but the Q values for higher neighbor amounts are ideally found to ensure smooth flocking to the target. A radial basis function (RBF) method of function approximation is used to achieve quicker learning. A fixed sparse representation method was explored but RBF was found to perform better. The state, action, and reward representations remain the same but the number of neighbors \( |N_i| \) is now being approximated using RBF.
3.4.1 Radial Basis Function

Function approximation allows to approximate a state space rather than just discretize it. There are many methods of doing function approximation but the method that seemed most applicable was a simple RBF approach. The RBF scheme maps the original Q table to a parameter vector \( \theta \) as:

\[
Q_i(s_i, a_i) = \sum l \phi_i(s_i, a_i) \theta_{i,l} = \phi_i^T(s_i, a_i) \theta_i
\]

where the RBF kernel \( \phi \) is a column vector of length \( l \cdot |\{A\}| \). The output of the \( l^{th} \) RBF kernel is given as

\[
\phi_l(s) = e^{-\frac{||s - \bar{s}_l||^2}{2\mu_l^2}}
\]

where \( s \) is the current state, \( \bar{s}_l \) is the center of the RBF kernel \( l \), \( \mu_l \) is the radius of the RBF kernel \( l \) producing the shape of a Gaussian bell. A larger \( \mu_l \) thus produces a flatter RBF.

3.4.2 Function Approximation Learning

The cooperative learning algorithm from equation 3.2 and 3.3 is still used to learn however it is modified to account for the parameter vector \( \theta \) and RBF kernel \( \phi \) in equation 3.6 rather than Q-values. The independent part of RBF learning is then given as

\[
\theta_i^{k+1} \leftarrow \theta_i^k + \alpha \gamma \max_{\hat{a}_i \in A_i} (\phi_i^T(s_i, \hat{a}_i) \theta_i^k) - (\phi_i^T(s_i, a_i) \theta_i^k) \phi_i(s_i, a_i)
\]
with the same learning rate $\alpha$ and discount factor $\gamma$ as before. The cooperative portion follows as
\[
\theta_i^{k+1} = w\theta_i^k + (1 - w)\frac{\sum_{j=1}^{N_i} \theta_j^k}{|N_i|}.
\]  
(3.8)

The main key difference here is that each agent must now communicate a $\theta$ vector rather than just a single Q table value since the entire $\theta$ vector approximates the state. In standard reinforcement learning the learning is conducted over multiple episodes. The episodes here consist of iterations of agents flocking towards the targets corresponding to their actions. Learning is concluded when the agents all learn the same action for each given state. After learning all states the agents will have a

<table>
<thead>
<tr>
<th>Algorithm 1: Function Approximated Distributed Cooperative Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 <strong>Initialization</strong></td>
</tr>
<tr>
<td>2 Set parameters $\alpha$, $\gamma$, $\epsilon_g$</td>
</tr>
<tr>
<td>3 Initialize $\theta$ and position of static targets</td>
</tr>
<tr>
<td>4 for each episode do</td>
</tr>
<tr>
<td>5   Initialize position of agents and predator</td>
</tr>
<tr>
<td>6 for each iteration $k$ do</td>
</tr>
<tr>
<td>7     for each agent $i$ do</td>
</tr>
<tr>
<td>8         <strong>Initialization Phase</strong>: Observe current state $s_i$</td>
</tr>
<tr>
<td>9         Select action based on equ. (3.4)</td>
</tr>
<tr>
<td>10        <strong>Action Phase</strong>: Each agent performs its action updating the states by equ. (2.9)</td>
</tr>
<tr>
<td>11        <strong>Update Phase</strong>: Observe next state ($\dot{s}_i$)</td>
</tr>
<tr>
<td>12        Select next action ($\dot{a}_i$) based on max $\theta_i$</td>
</tr>
<tr>
<td>13        Compute Reward ($r_i$)</td>
</tr>
<tr>
<td>14        Compute $\theta_i$ value using equ. (3.7)</td>
</tr>
<tr>
<td>15        Update $\theta_i$ based on its neighbors using equ. (3.8)</td>
</tr>
<tr>
<td>16     end</td>
</tr>
<tr>
<td>17 The number of iterations $k$ is determined by the number of iterations required for the agents to flock to a target.</td>
</tr>
<tr>
<td>18     end</td>
</tr>
<tr>
<td>19 Training is terminated after all agents choose the same target for all predator directions.</td>
</tr>
<tr>
<td>20     end</td>
</tr>
<tr>
<td>21 Output a learned $\theta$ table that agents use to know which target to go to depending on the direction of the predator.</td>
</tr>
</tbody>
</table>
learned \( \theta \) table that can be used to guide the agents in a flock to safe locations away from predators. The Algorithm for this learning is then given in Alg. 1.

### 3.5 Comparison of Learning Algorithms

If we use 50 agents \( n = 50 \) with 8 discrete directions \( |dir_p| = 8 \) and 8 actions corresponding to those directions \( |\{A\}| = 8 \), the Q table would be of size \( 50 \cdot 8 \cdot 50 \cdot 8 = 1.6 \cdot 10^5 \). Since the directions are already discretized and the number of actions and agents cannot be reduced only the neighbor dimension \( |N_i| \) can be reduced. For this application 8 RBF kernels were used to approximate the space although less or more will probably perform similarly. The \( \theta \) table is then of size \( 50 \cdot 8 \cdot 8 \cdot 8 = 2.56 \cdot 10^4 \) which is approximately \( \frac{1}{6} \) the original size. In a state space in which the direction of predators is ignored \( |dir_p| = 1 \) the Q size is \( 2 \cdot 10^4 \) while \( \theta \) is \( 3.2 \cdot 10^3 \). The results of this learning in this state space is shown in Fig. 3.2. It is clear that by using cooperative learning with function approximation the performance is significantly better than without in both space and training time required. Because of this large gap in learning effectiveness, only cooperative learning with function approximation is used for the larger state space where \( |dir_p| = 8 \).
Figure 3.2: Comparison of convergence between RBF and Q learning with and without cooperative learning. All four algorithms were run 10 times for 20 episodes and the results were averaged. Over the course of these runs cooperative RBF was able to converge within four episodes while cooperative Q learning was unsuccessful in fully converging within 20 episodes. Both independent algorithms were not able to learn.

3.6 Summary

In this chapter a method of learning targets to flock towards was proposed. Previous research has shown the benefit of cooperative learning over independent learning to greatly improve the learning time. This was taken a step further here by applying function approximation methods to approximate the state space providing even better results. Now that there is a method of flocking to a target and a way of learning a target to flock to, a way of sensing predators is required to learn to flock away from them.
Chapter 4

Consensus for Multi-agent State Approximation

In this chapter a way of sensing and communicating the direction of a predator is presented. Each agent has a predator sensing radius $r_p$ that allows them to sense a predator. If a predator is within that radius then the agent is able to know its relative angle to the predator. These angles can be used to determine the direction the predator is coming from. However not every agent will be in range of the predator to see the direction it is coming from and not every agent that is in range will agree on which direction the predator is coming from. To solve this a weighted voting method is introduced for agents to share and achieve a consensus on the direction of the predator.
4.1 Predator Sensing

A means of agents achieving consensus proposed in [43] was the start point for this weighted voting used here. The algorithm is split into two components, a measurement step and a consensus step. For the measurement step, if an agent is in range of the predator then it performs a measurement of the relative direction of the predator. For the purposes of this paper the direction is discretized into eight directions evenly split directions as mentioned previously. These directions are assigned to an information vector $info_i$ as

$$info_i = \begin{cases} 
1 & 0 \leq w_p \leq 22.5, 337.5 \leq w_p \leq 360 \\
2 & 22.5 \leq w_p \leq 67.5 \\
3 & 67.5 \leq w_p \leq 112.5 \\
4 & 112.5 \leq w_p \leq 157.5 \\
5 & 157.5 \leq w_p \leq 202.5 \\
6 & 202.5 \leq w_p \leq 247.5 \\
7 & 247.5 \leq w_p \leq 292.5 \\
8 & 292.5 \leq w_p \leq 337.5
\end{cases} \tag{4.1}$$

where $w_p$ is the angle between the agent and the predator such that $0 \leq w_p \leq 360$. These directions are visualized in Fig. 4.1. These directions do not have to be symmetrical or positioned as have been positioned here. There can also be more or less directions as desired. However the directions for this application were chosen as eight evenly divided directions such that they are all 45 degrees in width, $360/8 = 45$, and they are aligned to the cardinal directions.

Each agents information vector is assigned a weight or a belief factor $weight_{i,d}$. 
Figure 4.1: Direction and associated information values for an agent observing a predator. Each color corresponds to an information value 1 through 8 with 1 representing east, 3 representing north, and so forth.

This weight vector is of size number of agents by number of directions $n \cdot |dir_p|$ and is determined from an agent’s measurement given by

$$weight_{i,d} = \begin{cases} 
(1 - \frac{||q_p - q_i||}{r_p}) \times (\frac{w_m - w_p + 45}{45}) & ||q_p - q_i|| < r_p, d = info_i \\
(1 - \frac{||q_p - q_i||}{r_p}) \times (1 - \frac{w_m - w_p + 45}{45}) & ||q_p - q_i|| < r_p, d = info_i + 1, w_p > w_m \\
(1 - \frac{||q_p - q_i||}{r_p}) \times (1 - \frac{w_p - w_m + 45}{45}) & ||q_p - q_i|| < r_p, d = info_i - 1, w_p < w_m \\
0 & \text{otherwise} 
\end{cases}$$

(4.2)

where $w_m$ is the middle angle of the direction measured, $info_i + 1$ is the next direction clockwise, and $info_i - 1$ is the next direction in the counterclockwise. For example if $info_i = 1$ then $info_i - 1 = 8$. The scale factor $\frac{w_m - w_p + 45}{45}$ splits the distance weight $1 - \frac{||q_p - q_i||}{r_p}$ into two directions of the weight vector. This scale factor is between zero and one and is determined by where the measured angle is relative to the center of the direction. For example the center of direction one $w_m$ is 0 degrees and if the measured angle $w_p$ is 0 degrees then $\frac{w_m - w_p + 45}{45} = 1$. Thus the $weight_{i,1} = (1 - \frac{||q_p - q_i||}{r_p}) \cdot 1$. If the measured angle $w_p$ was 22.5 then $\frac{w_m - w_p + 45}{45} = 0.5$ and $1 - \frac{w_m - w_p + 45}{45} = 0.5$. Thus
weight_{i,1} = weight_{i,2} = 1 - \frac{||q_e - q_i||}{r_p} \times 0.5. The idea here is to assign weight based on closeness to the predator and closeness to the center of the directions. Once the information and weight has been found for all agents we can then run a consensus based on weighted voting.

4.2 Consensus

For consensus each agent updates its information info_i and weight weight_{i,d} based on its neighbors N_i. The goal is for all agents to agree on the same info_i and for that info_i to be as accurate as possible thus achieving consensus on the direction of the predator. To do this a weighted voting method is implemented where the weights for an agent and its neighbors are summed together into the weighted direction vector weight_i such that weight_i = weight_i + \sum_{j=1}^{||N_i||} weight_j. The info_i is then set to the direction that has a maximum weight info_i = max_d(weight_{i,d}). The weight and information is updated for all agents for a set amount of iterations c_m in this manner then the weight for each agent is updated to the maximum weight amongst itself and its neighbors such that weight_i = max_{weight}(weight_{N_i} \cup weight_i). Sharing the maximum weight after the set amount of iterations allows for all agents to converge to the same predator direction in a quick manner. The measurement and consensus steps can be combined as seen in Algorithm 2.

By using this algorithm info_i is found for each agent and given enough enough iterations, the proposed consensus will converge to the same value for all agents. This value is used to determine the state dir_p in the reinforcement learning component.
Algorithm 2: Consensus on Direction of Predator

1. Initialize $r_p$
2. Initialize $c_m$
3. Measurement Phase:
   for Each agent $i$ do
   if $||q_p - q_i|| < r_p$ then
   Find $info_i$ from equ. (4.1)
   Find $weight_{i,d}$ from equ. (4.2)
   else
   $weight_i = 0$
   end
   end
4. Consensus Phase:
   for Each Consensus Iteration $c$ do
   for Each agent $i$ do
   Update the weighted direction vector
   if $c \leq c_m$ then
   $weight_i = weight_i + \sum_{j=1}^{N_i} weight_j$
   else
   $weight_i = \max_{weight}(weight_{N_i} \cup weight_i)$
   end
   Update the $info_i$
   $info_i = \max_d(weight_{i,d})$
   end
   end

4.3 Validation

Algorithm 2 is tested in an environment where 50 agents are flocking to a static position while seeing a predator, denoted by a large red circle, moving in a circle around the flock over 900 iterations as can be seen in Fig. 4.2. Visually the agents that are represented by the triangles change color in association with the direction they perceive the predator to be in after consensus. The consensus component was allowed to run for 20 consensus iterations and was found that all agents converged to the same $info_i$ within that duration which can be seen in Fig. 4.3. One run of the average time it took to converge for each of the 900 iterations can be seen in 4.4.
Figure 4.2: Predator moving in circle around flock with consensus updating the state of the agents based on the direction of the predator.

<table>
<thead>
<tr>
<th>Effects of change in $c_m$</th>
<th>$c_m = 0$</th>
<th>$c_m = 1$</th>
<th>$c_m = 2$</th>
<th>$c_m = 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations to Converge</td>
<td>8.2</td>
<td>8.8</td>
<td>9.8</td>
<td>10.6</td>
</tr>
<tr>
<td>Incorrect Directions</td>
<td>114(12.7%)</td>
<td>75.2(8.4%)</td>
<td>68.6(7.6%)</td>
<td>73.8(8.2%)</td>
</tr>
</tbody>
</table>

Table 4.1: The iterations to converge and number of incorrect directions found over 900 iterations. $c_m = 2$ produces the best results.

Fig. 4.5 shows the comparison of the state found through consensus to the actual state relative to the center of mass of the flock. It can be seen that it is not always perfectly accurate but this can be attributed to lack of full observability and lack of symmetry in the flock. It was always able to fully converge for varying values of $c_m$ which can be seen in Tab. 4.1. However, if one does not do the maximum weight sharing by setting $c_m$ to infinity the algorithm will not fully converge even given up to 40 iterations as can be seen in Fig. 4.6 and 4.7.

By using Tab. 4.1 we can learn that if the largest measured weight is spread from one side of the flock to the other, $c_m = 0$, it takes 8.2 iterations to completely reach every agent in the flock. That amount of iterations to achieve a consensus is thus not able to be made smaller due to the network communication limitations and size of the flock. By adding additional weighted voting iterations $c_m$ we can see that it takes about one additional iteration to converge for each weighted voting iteration added. It
Figure 4.3: The number of agents in the same state (the same $inf_{\alpha_i}$) for each iteration. For all positions of the predator the agents are able to converge to the same state.

Figure 4.4: The number of consensus iterations for all agents to converge to the same state.
Figure 4.5: The average $info_i$ for the agents in each iteration. Some states occur for longer durations due to the shape of the flock.

Figure 4.6: The number of agents in the same state when $c_m$ is infinite. There are multiple predator positions that result in the agents not fully converging even given 40 iterations.
Figure 4.7: The convergence of the agents state when $c_m$ is infinite. Given 40 consensus iterations it is not able to fully converge.

Figure 4.8: The cause of error of the consensus algorithm. Despite the predator being east of the flock as a whole the only agent in range senses it as north east causing the error seen in Tab. 4.1.
is also clear that adding more weighted voting iterations does not necessarily increase
the accuracy of the consensus as can be seen by comparing $c_m = 2$ and $c_m = 3$. The
error in accuracy can be attributed to the formation of the flock as can be seen in Fig.
4.8. Despite the predator being east of the flock as a whole the agents will converge
to northeast due to only one agent being in range of the predator to sense it. Using
this data going forward to the hybrid system we let $c_m = 2$ and let the number of
consensus iterations be 12 to allow some margin of error to account for a poor flocking
structure.

Through testing alternate consensus methods one method was able to achieve a
higher accuracy. This method involved using consensus to determine the center of
mass of the flock and absolute position of the predator. When each agent has that
center of mass position and the position of the predator it can then determine for
itself the direction of the predator but this approach is not used for a few reasons.
First and foremost is the time it takes to reach a consensus is at least two to three
times the number of iterations that the proposed approach uses thus making it take
much longer to both learn and later use practically for predator detection. Secondly
it requires that each agent has an absolute coordinate for itself and the predator
rather than relative directions for those agents in range of the predator. In many
environments this may not be known or if it is may have a large amount of noise
associated with it so this approach was not used.

4.4 Summary

In this chapter a way of detecting and sharing the direction of a predator in a partially
observable environment. This is done in a relatively accurate manner subject to
the structure of the flock that takes very few iterations more than the amount of
iterations required to transmit data from one edge of the flock to the other. Now
we can develop an intelligent hybrid system given a method of flocking to a target, a
method of learning a target to flock towards, and a method of learning the direction
of predators to flock away from.
Chapter 5

Simulation and Results

In this chapter we go over implementation details and results found for the hybrid learning system. We use consensus to determine the direction of the predator $\text{dir}_p$. This state is then used in the multi-agent learning for the $\theta$ table. The multi-agent learning then produces an action which is a target to flock towards that is used by the flocking algorithm for the agents. Finally the flocking algorithm produces a control input for each agent to flock to its chosen target. Using this system we can teach agents to detect and flock away from predators.

5.1 Simulation Environment

The learning environment is set up in a manner shown in Fig. 5.1. As before the triangles represent the agents and the large red circle represents the predator. The eight smaller green circles around the edge represent the eight static targets for the eight actions. Each episode begins by randomly initializing 50 agents in a 120x120 area and the predator in one of eight directions. The predator then moves towards the center of mass of the agents. The predator is placed far enough away that the
agents will be fully connected to each other but not necessarily in perfect flocking formation by the time the predator gets in range. This is done to ensure each agent is able to get the direction of the predator through consensus so that an agent does not get left behind due to being initialized too far away from the rest of the agents.

5.1.1 Learning Configuration

For the single predator environment, the direction of the predator is initially east then northeast and so on in a counter clockwise rotation so that ideally every possible state \( \text{dir}_p \) is encountered once every eight episodes. The learning is conducted over 56 episodes and the results can be seen below. For the two predator environment the learning is conducted over 216 episodes with the reason explained below. For the epsilon greedy action selection in (3.4) an \( \epsilon \) value of 0.1 is chosen to to allow the agents to explore other actions more quickly while not hindering the smoothness of flocking too extremely.
5.2 Results

5.2.1 Single Direction State

We first look at a scenario in which all agents are in the same direction state with the use of consensus. The average results of 10 runs over 8 episodes can be seen in Fig. 5.2 and 5.3. We can see that the agents are able to fully converge for a single direction state in 4 episodes. Thus the theoretical number of episodes required to learn is 8 directions times 4 episodes required or 32. However due to the nature of flocks not being perfect it is possible for each direction to not be seen 4 times within those 32 episodes so 56 episodes are used to learn for the single predator and 216 for the two predator environment to account for slower learning due to consensus.

Figure 5.2: The number of agents choosing the same action for each episode for a single direction state by episode. The agents are able to fully converge by 4 episodes.
Figure 5.3: The number of agents choosing the same action for each episode for a single direction state by iteration with the dashed lines representing the start of an episode. It can be seen that most learning is done in the beginning of an episode while the flock is connected.

5.2.2 Single Predator

Six runs were run for the single predator environment with averages taken over the runs. In Fig. 5.4 it can be seen that by episode 32, or the predator coming from each direction four times, the agents have mostly converged to the same target but there are a few cases in which it is not fully converged until approximately episode 48 or each direction occurring six times which is expected due to potential consensus inaccuracies.

The position of agents during the first learning episode can be seen in Fig. 5.5. Some agents are in different states and the agents in the same state have not learned to go to the same target yet. This produces the messy flocking shape that can be seen. In Fig. 5.5 the agents are all in the same state and have learned to go to the same target in an \( \alpha \)-lattice formation.

In Fig. 5.7 we have the trajectory the agents take in the final learning episode where the pink triangles are the random initialization of the agents. By the last
Figure 5.4: The number of agents choosing the same action for each episode for a single predator environment averaged over 6 runs. It can be seen that the agents converge by each direction being encountered 6 times.

Figure 5.5: 50 agents flocking away from one predator before learning the same target.
Figure 5.6: 50 agents flocking away from one predator after learning the same target.
Figure 5.7: Trajectory the agents take in the last episode where the triangles are the initial position of the agents. The Flocking can be seen to be smooth to the target after agents get into flocking formation despite the $\epsilon$-greedy random action selection.

episode it can be seen that all agents have converged to the same target flocking in a relatively smooth manner despite the random action selection of $\epsilon$-greedy.

5.2.3 Two Predators

Two predators have also been tested to perform well with an expanded information vector to account for the extra predator and thus a larger state space as well. The addition of a second predator increases the number of predator starting positions by a factor of eight from 8 positions to 64. In addition to longer computation times for handling a second predator there is now 8 times the amount of episodes that must be performed for all direction combinations to be encountered. Unfortunately this can not be reduced, without reducing the problem size, by applying a function approximation approach to the learning process. However there is a way to lower the amount of direction combinations. Instead of learning the direction of the two predators separately we treat both predators as the same predator. This way if the first predator is detected in direction 1 and the second predator is detected in direction
Figure 5.8: Number of agents choosing the same action for two predators over 216 episodes. It can be seen that after 180 episodes all agents choose the same action for each state.

2 it is the same as if the first predator is in direction 2 and the second predator is in direction 1. Thus the system learns two state combinations simultaneously. However there are 8 direction combinations where both predators are detected in the same direction which is not reducible. This reduces the number of direction combinations from 64 down to 36. Which reduces the number of episodes and thus the time it takes to learn from 8 times that of a single predator to 4.5 times. For this reason the two predator learning is done over 216 episodes the results of which can be seen in one run in Fig. 5.8. It can be seen that by 144 episodes, or all combinations of directions being encountered 4 times the agents have almost converged. By 180 episodes or each direction being seen 5 times the agents have completely converged to the same action for each state.

The movement of the agents flocking away from the two predators before they have learned and after they have learned to flock away from the predators can be seen in Fig. 5.9 and 5.10 respectively which looks similar to that of the single predator.
Figure 5.9: 50 agents flocking away from two predators before learning the same target.
Figure 5.10: 50 agents flocking away from two predators after learning the same target.

The video of the simulations is given in this link: https://youtu.be/zL3rj5GqbXM

5.3 Summary

In this chapter the simulation design and results of the hybrid system are presented. By looking at a single direction we are able to determine the number of episodes
required to learn all directions. Visuals of the learning are given as well as results computed over multiple runs. Based on the results it is clear this intelligent hybrid-system is capable of learning how to flock away from attacking predators.
Chapter 6

Conclusion and Future Work

6.1 Conclusion

This thesis presented a hybrid system that achieves an efficient cooperative learning behavior. The system is applied to the task of escaping attacking predators while maintaining a flocking formation.

First a flocking algorithm is presented that enables agents to flock to a defined target in a distributed manner while maintaining a lattice formation. The flocking algorithm uses a gradient-based term, attractive/repulsive force, and a velocity consensus term to ensure the agents flock in formation towards a target without colliding with each other or obstacles. The flocking algorithm was shown to effectively allow 50 agents to flock to a target with and without an obstacle in the path.

Then multiple reinforcement learning methods were presented utilizing the flocking targets as actions in the learning algorithm. Independent learning with and without function approximation proved to be unreliable in learning to flock together to the same target. Cooperative learning without function approximation was shown
to be better than both independent learning methods but still took a considerable amount of episodes to learn. Cooperative learning with function approximation was shown to perform the best requiring very few episodes for the agents to converge to the same target.

Finally, a method of detecting predators is used to determine the direction of attacking predators. The method proposed was shown to be very accurate requiring only a few more iterations than it takes to transmit information from one edge of the flock to the other. The direction of the detected predators is used for determining the state of the system for the reinforcement learning component.

The hybrid system was then developed consisting of flocking control, function approximated cooperative learning, and consensus to allow agents to learn the location of a predator and where to flock away from it. The system was tested in one and two predator environments with results showing the success of the system as an efficient cooperative learning method.

6.2 Future Work

Although the hybrid-system proposed here was used to solve the task of avoiding predators by agents flocking together to targets, it is generic enough to be used in a variety of multi-agent tasks. Applying this hybrid-system of using consensus to determine the states and reinforcement learning to learn the states is something that can be looked further into in the future to achieve efficient learning for a variety of tasks. A possible improvement to this work is to create a more parallelized framework for the simulation to allow for faster learning particularly for increased amounts of predators that grow the state space and learning configuration. Further testing can be done with an expanded state size of more predators and a three dimensional simulation
environment. In addition to this implementing smarter predators and different types of agents are also tasks that could be looked into in the future.
Bibliography


Conference on Cyber Technology in Automation, Control and Intelligent Systems, May 2013, pp. 337–342.


putational Techniques in Information and Communication Technologies (ICC-TICT), March 2016, pp. 145–150.


