

Consensus, cooperative learning, and flocking for multiagent predator avoidance

Zachary Young and Hung Manh La

Abstract

Multiagent 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 article, a hybrid multiagent system that integrates consensus, cooperative learning, and flocking control to determine the direction of attacking predators and learns 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 multiagent coordination tasks, but this article 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 multiagent tasks.

Keywords

Distributed algorithms for multirobot coordination, mobile robots and multirobot systems, multiagent robot teams, mobile sensor networks, swarm robotics, multiagent learning, function approximation, consensus, flocking control

Date received: 6 May 2020; accepted: 18 August 2020

Topic Area: Mobile Robots and Multirobot Systems

Associate Editor: Changjoo Nam

Topic Editor: Nak-Young Chong

Introduction

Motivation

Multiagent cooperative learning has been continuing to be a large research interest in the field of robotics with a wide range of applications.^{1–3} Tracking wildfires using multiple agents communicating together to better handle the fire, so it does not destroy as much, is one such possibility.^{4,5} Another one is that using multiple agents to better map the structure of a pipeline that if it structurally fails could cause large damage and loss of money.⁶ An additional possibility is mapping and exploring unknown environments.⁷ This article aims to solve the task of

multiagent predator avoidance^{8,9} with an intelligent hybrid system.

Most current multiagent research incorporates some form of consensus,¹⁰ movement control,¹¹ or reinforcement

Department of Computer Science and Engineering, Advanced Robotics and Automation (ARA) Laboratory, University of Nevada, Reno, NV, USA

Corresponding author:

Hung Manh La, Department of Computer Science and Engineering, Advanced Robotics and Automation (ARA) Laboratory, University of Nevada, Reno, NV 89557, USA.

Email: hla@unr.edu



Creative Commons CC BY: This article is distributed under the terms of the Creative Commons Attribution 4.0 License (<https://creativecommons.org/licenses/by/4.0/>) which permits any use, reproduction and distribution of the work without

further permission provided the original work is attributed as specified on the SAGE and Open Access pages (<https://us.sagepub.com/en-us/nam/open-access-at-sage>).

learning,¹² 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 obstacles¹³ or to make measurements in a scalar field.^{14–16} Movement control of multiagent systems can come in the form of cooperatively doing path planning as in the literature.^{17–19} Alternatively, there are means of control through flocking in varying formations^{20–22} to achieve a variety of tasks.^{23,24} Reinforcement learning has been implemented cooperatively in a variety of ways for multiagent environments, such as a GridWorld^{25,26} and box pushing.²⁷ Although all of these use for consensus, movement control, and reinforcement learning that are good in their own right, this article aims to make a more intelligent hybrid system.

In nature, flocking has long been observed in many environments^{28–30} with one possible goal being to defend from predators as can be seen with schools of fish. Even simulated environments^{31,32} 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 article is then to create a hybrid system that combines consensus, flocking, and multiagent reinforcement learning into one intelligent system that can sense and learn to escape from attacking predators.

Literature review

Flocking background. Methods of ensuring agents flocking have been proposed and studied in the literature.^{33–37} Inspired by the natural world of birds and fish flocking together,³⁸ flocking algorithms have been formed. The algorithms allow 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 too 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.^{39,40} There are also cases in which reinforcement learning is implemented to teach agents to go to specific targets, but in cases where learning fails, a flocking algorithm to avoid obstacles is used.⁴¹ 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. Other flocking implementations such as that proposed in the literature⁴² allow agents to flock together with minimal information transfer, but the formation is not ideal for quickly avoiding a predator. Additional flocking features can be added, such as handling faulty robots,⁴³ time delay for information transfer between robots,⁴⁴ or change of formation.⁴⁵ For this

article, these are not the focus but could potentially be added in the future.

Reinforcement learning background. Cooperation is an important part of a flock learning to do a task together. In the literature,⁴⁶ 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-learning⁴⁷ does not achieve the necessary amounts of cooperation required, so a more cooperative approach^{9,48} is required. Unfortunately, the number of agents increases the state space that grows and requires longer amounts of training to effectively learn to flock together to the same destination. For this, function approximation techniques^{49,50} are useful. However, in current research, function approximation in combination with cooperative learning is largely unexplored.^{51,52}

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 the literature.⁵³ Algorithms have been proposed to allow multiagent systems to come to a consensus on measurements from multiple agents that do not necessarily agree with each other.^{54,55} Many uses of consensus, however, are for estimating some kinds of measurements.^{56,57} This work intends to use consensus in combination with reinforcement learning to make a more intelligent system. Some works, such as by Zhang et al.⁵¹ and Xu et al.,⁵² 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.

Contributions

In this article, we combine the benefits of consensus, flocking, and reinforcement learning to create a hybrid system, as shown in Figure 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 toward. That target is then used by a flocking algorithm to give each agent a

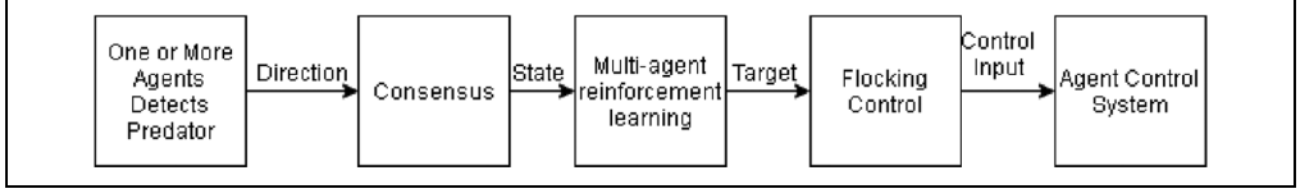


Figure 1. Block diagram of the hybrid system.

control input to move each agent toward the target in a flocking formation.

The contributions of this article 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 a 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.

Paper organization

The organization of the remainder of this article is as follows. In “Flocking control” section, a method for flocking is introduced. The “Multiagent learning” section goes over the multiagent learning that used to ensure the agents flock together to the same target. The “Consensus for multiagent state approximation” section details a method of state approximation called consensus for the agents. This is followed by “Simulation and results” section, 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, “Conclusion and future work” section covers the conclusion with analysis and potential future development.

Flocking control

Flocking control algorithm

In this section, the flocking algorithm used for the hybrid system is presented. To learn to avoid predators, the agents must be able to flock together. Using flocking methodologies presented in the literature,³³ a network topology consisting of a graph G that is a pair (V, E) with a set of vertices $V = \{1, 2, \dots, n\}$ and edges $E \subseteq \{(i, j) : i, j \in V, j \neq i\}$. In this graph, the agents are considered vertices, and the edges are communication links between neighboring agents. Using agents modeled as particles, the equations of motion are given by

$$\begin{cases} \dot{\mathbf{q}}_i = \mathbf{p}_i \\ \dot{\mathbf{p}}_i = \mathbf{u}_i \end{cases} \quad (1)$$

where $\mathbf{q}_i$ is the position of agent i , $\mathbf{p}_i$ is the velocity, and $\mathbf{u}_i$ is the acceleration or the control input.

The neighbors of an agent can be determined by

$$N_i = \{j \in V : \|\mathbf{q}_j - \mathbf{q}_i\| < r\} \quad (2)$$

where $\|\cdot\|$ is the Euclidean norm and r is the interaction range of an agent.

There are many formations that flocking can take, but the formation used here is an α -lattice formation in which

$$\|\mathbf{q}_j - \mathbf{q}_i\| = d \quad \forall j \in N_i(\mathbf{q}) \quad (3)$$

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 $\mathbf{f}_i^g$ given by³³

$$\mathbf{f}_i^g = c_1 \sum_{j \in N_i} \phi_\alpha(\|\mathbf{p}_j - \mathbf{p}_i\|_\sigma) \mathbf{n}_{ij} \quad (4)$$

where c_1 is a positive constant, $\mathbf{n}_{ij} = \sigma_\varepsilon(\mathbf{q}_j - \mathbf{q}_i) = (\mathbf{q}_j - \mathbf{q}_i) / \sqrt{1 + \varepsilon \|\mathbf{q}_j - \mathbf{q}_i\|^2}$, and $\phi_\alpha(\cdot)$ is a pairwise attractive/repulsive force to maintain the desired distance d between agents. With σ -norm, $\|\cdot\|_\sigma$ is given by $\|x\|_\sigma = 1/\varepsilon [\sqrt{1 + \varepsilon \|x\|^2} - 1]$ that is differentiable everywhere for $\varepsilon > 0$. An obstacle avoidance term is given by $\mathbf{f}_i^o$ that is the repulsive force of $\mathbf{f}_i^g$ using points on obstacles as virtual neighbors N_i^β given by

$$\mathbf{f}_i^o = \sum_{j \in N_i^\beta} b_{ij}(\mathbf{q})(\mathbf{p}_j - \mathbf{p}_i) \quad (5)$$

where $b_{ij}(\mathbf{q}) = \mathbf{p}_h(\|\mathbf{q}_j - \mathbf{q}_i\|_\sigma / \|r\|_\sigma)$ is an element at row i and column j of an adjacency matrix over the interval $[0, 1)$ for virtual neighbors j . A velocity consensus term $\mathbf{f}_i^d$ is given by

$$\mathbf{f}_i^d = c_2 \sum_{j \in N_i} a_{ij}(\mathbf{q})(\mathbf{p}_j - \mathbf{p}_i) \quad (6)$$

where $a_{ij}(\mathbf{q}) = \mathbf{p}_h(\|\mathbf{q}_j - \mathbf{q}_i\|_\sigma / \|r\|_\sigma)$ is an adjacency matrix over the interval $[0, 1)$, and c_2 is a positive constant. $\mathbf{p}_h(\cdot)$ is a bump function that smoothly varies between 0 and 1. One possible definition is given by

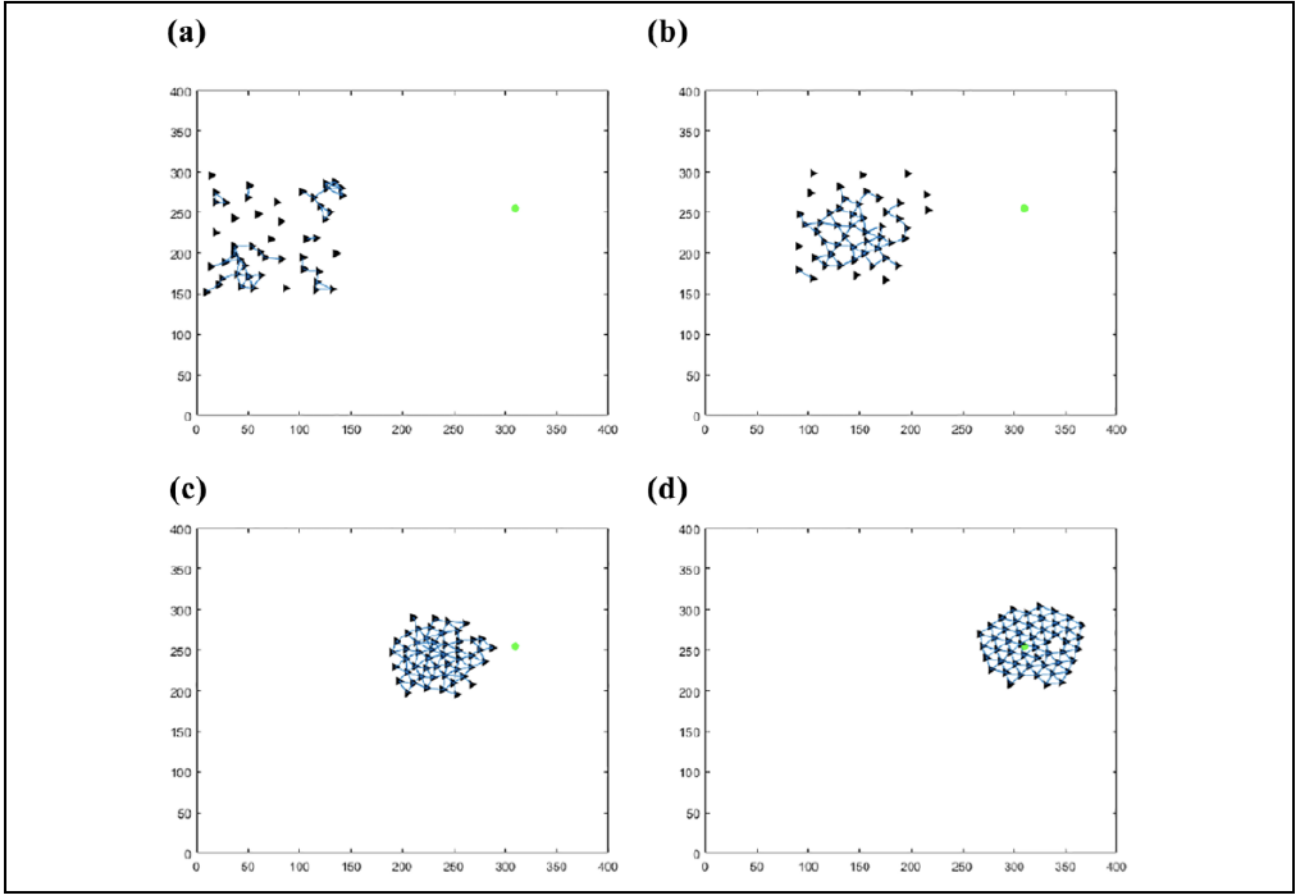


Figure 2. (a–d) Fifty agents flocking to the green target.

$$\mathbf{p}_h(z) = \begin{cases} 1 & z \in [0, h) \\ \frac{1}{2} [1 + \cos(\pi \frac{(z-h)}{(1-h)})] & z \in [h, 1] \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

where $h \in (0, 1)$. A navigational term, $\mathbf{f}_i^\gamma$, that determines the direction the agents should be moving toward, is given by

$$\mathbf{f}_i^\gamma = -c_{1t}(\mathbf{q}_i - \mathbf{q}_t) - c_{2t}(\mathbf{p}_i - \mathbf{p}_t) \quad (8)$$

where c_{1t} and c_{2t} are positive constants, and $\mathbf{q}_t$ and $\mathbf{p}_t$ are the position and velocity of the target, respectively. These equations can be combined to find the control input for each agent $\mathbf{u}_i$ given by

$$\mathbf{u}_i = \mathbf{f}_i^g + \mathbf{f}_i^d + \mathbf{f}_i^\gamma + \mathbf{f}_i^o \quad (9)$$

This method allows the agents to flock together in an α -lattice formation toward a target location.

Results of flocking algorithm

The results of this flocking algorithm ($\mathbf{u}_i$) can be seen in Figure 2, where the agents are flocking to the green dot without an obstacle, and in Figure 3 with an obstacle. The

agents are initialized randomly over a 120×120 area and flock toward 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 α -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.

Multiagent learning

In this section, 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.

Learning model

The model of the learning algorithm is similar to that proposed in the literature.⁹ Using a state, action, and reward

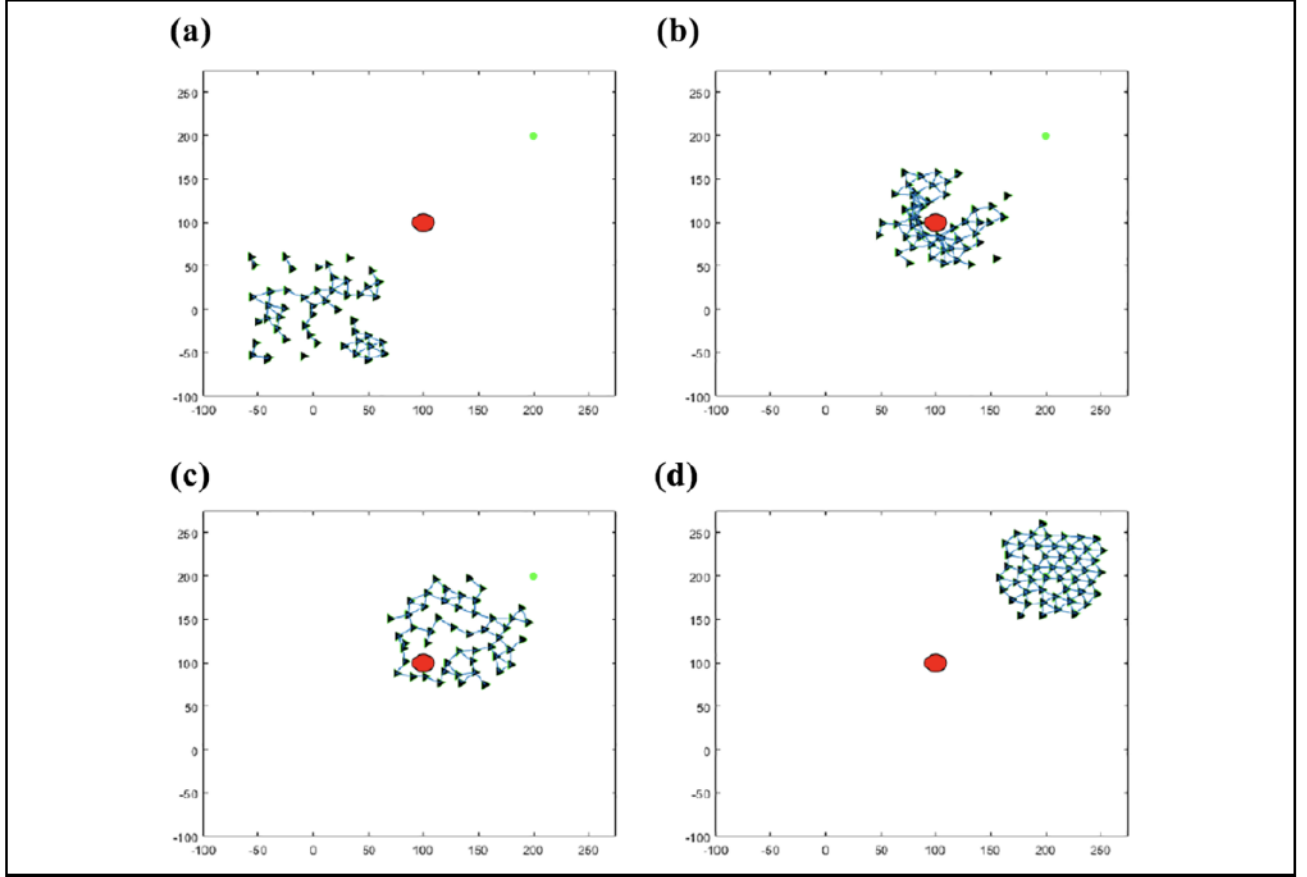


Figure 3. (a–d) Fifty agents flocking to the green target while avoiding the red obstacle.

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 s'_i and a'_i , respectively.

State. The state can be defined as $s_i = [\text{dir}_p, |N_i|]$, where 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 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 dir_p state for each predator.

Action. For actions, the agents want to move in one of eight cardinal directions to escape a predator depending on the direction 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 toward 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 toward.

Reward. The flocking algorithm used provides flocking in an α -lattice formation. This formation ensures that 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

$$r_i = \begin{cases} |N_i^a| \cdot D_r & |N_i| < 6 \\ 6 \cdot D_r & \text{otherwise} \end{cases} \quad (10)$$

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 factor 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 Figure 4. Agents choosing the action corresponding to the best target have their reward equal to the reward defined in equation (10). D_r is a scale factor that is determined as follows: Actions corresponding to good targets are scaled down to 75% of the reward above, average targets to 50%, bad targets to 25%, and the worst target to 0%. This is done to encourage the agents to learn the optimal target to go toward while maintaining the importance of flocking

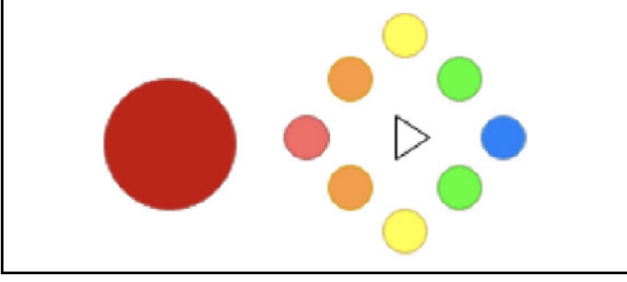


Figure 4. Visualization of the direction classifications. The large red circle represents the predator while the triangle represents the agent. The blue circle represents the best target, green circles are good targets, yellow circles are average targets, orange circles are bad targets, and the small red circle is the worst target in this scenario.

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% twice. 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.

Cooperative learning

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 the literature.⁹ However, to cooperatively learn, each agent must first do independent learning⁴⁷ for an individual table, Q_i , as follows:

$$Q_i^{k+1}(s_i, a_i) \leftarrow Q_i^k(s_i, a_i) + \alpha[r_i^k + \gamma \max_{a'_i \in A_i} Q_i^k(s'_i, a'_i) - Q_i^k(s_i, a_i)] \quad (11)$$

where α is a learning rate and γ 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 following⁹:

$$Q_i^{k+1}(s_i, a_i) \leftarrow wQ_i^k(s_i, a_i) + (1 - w) \frac{\sum_{j=1}^{|N_i|} Q_j^k(s_j, a_i)}{|N_i|} \quad (12)$$

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 .

Action selection

The action selection of an agent is based on the maximum Q -value approach^{48,58} 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 ϵ -greedy.⁴⁷ 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

$$a_i = \begin{cases} a_{\max} \in A_i & \epsilon_g < \text{random}(0, \dots, 1) \\ a_{\text{random}} \in A_i & \text{otherwise} \end{cases} \quad (13)$$

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 θ parameter vector.

Function approximation

Despite the cooperative Q learning algorithm performing better than independent learning, as seen in the literature,⁹ it still can be improved upon to get better results and faster convergence. The direction of the predator 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 α -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.

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 θ as^{49,50}

$$Q_i(s_i, a_i) = \sum \phi_l(s_i, a_i) \theta_{i,l} = \phi^T(s_i, a_i) \theta_i \quad (14)$$

where the RBF kernel ϕ is a column vector of length $l \cdot |\{A\}|$. The output of the l 'th RBF kernel is given as

$$\phi_l(s) = e^{-\frac{\|s-\bar{s}_l\|^2}{2\mu_l^2}} \quad (15)$$

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

Function approximation learning. The cooperative learning algorithm from equations (11) and (12) is still used to learn, however, it is modified to account for the parameter vector θ and RBF kernel ϕ in equation (15) rather than Q -values.⁴⁹ The independent part of RBF learning is then given as

$$\begin{aligned} \theta_i^{k+1} \leftarrow & \theta_i^k + \alpha[r_i^k + \gamma \max_{a' \in A_i} (\phi^T(s'_i, a'_i) \theta_i^k) \\ & - (\phi^T(s_i, a_i) \theta_i^k) \phi(s_i, a_i)] \end{aligned} \quad (16)$$

with the same learning rate α and discount factor γ as before. The cooperative portion is as follows

$$\theta_i^{k+1} = w\theta_i^k + (1-w) \frac{\sum_{j=1}^{|N_i|} \theta_j^k}{|N_i|} \quad (17)$$

The key difference here is that each agent must now communicate a θ vector rather than just a single Q table value since the entire θ vector approximates the state. In standard reinforcement learning, the learning is conducted over multiple episodes. The episodes here consist of iterations of agents flocking toward 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 learned θ 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 Algorithm 1.

Comparison of learning algorithms

If we use 50 agents $n = 50$ with eight discrete directions $|\text{dir}_p| = 8$ and eight actions corresponding to those directions $|\{A\}| = 8$, the Q table would be of size $50 \times 8 \cdot 50 \times 8 = 1.6 \times 10^5$. 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, eight RBF kernels were used to approximate the space although less or more will probably perform similarly. The θ table is then of size $50 \times 8 \cdot 8 \times 8 = 2.56 \times 10^4$, which is approximately one-sixth of the original size. In a state space in which the direction of predators is ignored $|\text{dir}_p| = 1$, the Q size is 2×10^4 while θ is 3.2×10^3 . The results of this learning in this state space are shown in Figure 5. It is clear that 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

Algorithm 1. Function approximated distributed cooperative learning

Initialization

Set parameters $\alpha, \gamma, \epsilon_g$

Initialize θ and position of static targets

for each episode do

Initialize position of agents and predator

for each iteration k do

for each agent i do

Initialization Phase: Observe current state s_i

Select action based on eq. (13)

Action Phase: Each agent performs it's action updating the states by eq. (9)

Update Phase: Observe next state (s'_i)

Select next action ($\hat{a}_i$) based on max θ_i

Compute Reward (r_i)

Compute θ_i value using eq. (16)

Update θ_i based on its neighbors using eq. (17)

end

The number of iterations k is determined by the number of iterations required for the agents to flock to a target.

end

Training is terminated after all agents choose the same target for all predator directions.

end

Output a learned θ table that agents use to know which target to go to depending on the direction of the predator.

function approximation is used for the larger state space, where $|\text{dir}_p| = 8$.

Consensus for multiagent state approximation

In this section, 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 that 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.

Predator sensing

A method of agents achieving consensus proposed in the literature⁵⁶ is used as a start point for the weighted voting procedure. The algorithm is split into two components: a

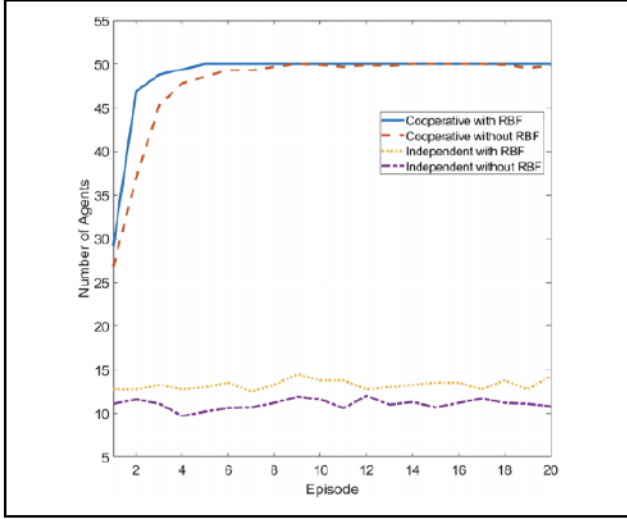


Figure 5. Comparison of convergence between RBF and Q learning with and without cooperative learning. All four algorithms were run 10 times for 20 episodes, and the results were averaged. Over the course of these runs, cooperative RBF was able to converge within four episodes while cooperative Q learning was unsuccessful in fully converging within 20 episodes. Both independent algorithms were not able to learn. RBF: radial basis function.

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. As mentioned previously, the direction is discretized into eight evenly split directions. These directions are assigned to an information vector info_i as

$$\text{info}_i = \begin{cases} 1 & 0 \leq w_p < 22.5, 337.5 \leq w_p < 360 \\ 2 & 22.5 \leq w_p < 67.5 \\ 3 & 67.5 \leq w_p < 112.5 \\ 4 & 112.5 \leq w_p < 157.5 \\ 5 & 157.5 \leq w_p < 202.5 \\ 6 & 202.5 \leq w_p < 247.5 \\ 7 & 247.5 \leq w_p < 292.5 \\ 8 & 292.5 \leq w_p < 337.5 \end{cases} \quad (18)$$

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 Figure 6. 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

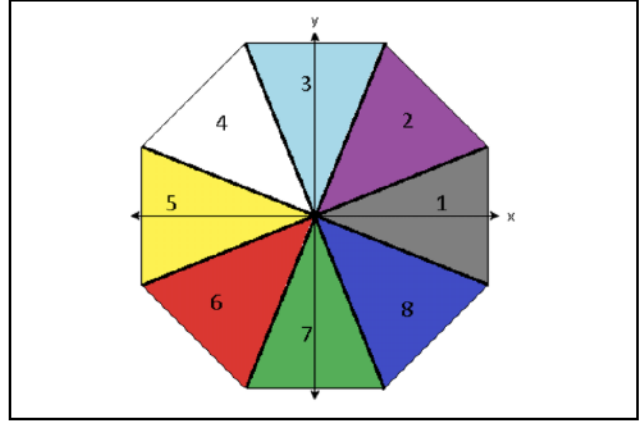


Figure 6. Direction and associated information values for an agent observing a predator. Each color corresponds to an information value 1 through 8 with 1 representing east, 3 representing north, and so forth.

for this application were chosen as eight evenly divided directions such that they are all 45° in width, $360/8 = 45$, and they are aligned to the cardinal directions.

Each agent's information vector is assigned a weight or a belief factor $\text{weight}_{i,d}$. This weight vector is of size number of agents by the number of directions $n \cdot |\text{dir}_p|$ and is determined from an agent's measurement given by equation (19), where w_m is the middle angle of the direction measured, $\text{info}_i + 1$ is the next direction counterclockwise, and $\text{info}_i - 1$ is the next direction clockwise. For example, if $\text{info}_i = 1$, then $\text{info}_i - 1 = 8$. The scale factor $\frac{w_m - w_p + 45}{45}$ splits the distance weight $1 - \frac{\|\mathbf{q}_p - \mathbf{q}_i\|}{r_p}$ into two directions of the weight vector. This scale factor is between 0 and 1, 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° , and if the measured angle w_p is 0° , then $\frac{w_m - w_p + 45}{45} = 1$. Thus, the $\text{weight}_{i,1} = (1 - \frac{\|\mathbf{q}_p - \mathbf{q}_i\|}{r_p}) \cdot 1$. 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, $\text{weight}_{i,1} = \text{weight}_{i,2} = 1 - \frac{\|\mathbf{q}_p - \mathbf{q}_i\|}{r_p} \cdot 0.5$. 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

$$\text{weight}_{i,d} = \begin{cases} \left(1 - \frac{\|\mathbf{q}_p - \mathbf{q}_i\|}{r_p}\right) * \left(\frac{w_m - w_p + 45}{45}\right) & \|\mathbf{q}_p - \mathbf{q}_i\| < r_p, d = \text{info}_i \\ \left(1 - \frac{\|\mathbf{q}_p - \mathbf{q}_i\|}{r_p}\right) * \left(1 - \frac{w_m - w_p + 45}{45}\right) & \|\mathbf{q}_p - \mathbf{q}_i\| < r_p, d = \text{info}_i + 1, w_p > w_m \\ \left(1 - \frac{\|\mathbf{q}_p - \mathbf{q}_i\|}{r_p}\right) * \left(1 - \frac{w_p - w_m + 45}{45}\right) & \|\mathbf{q}_p - \mathbf{q}_i\| < r_p, d = \text{info}_i - 1, w_p < w_m \\ 0 & \text{otherwise} \end{cases} \quad (19)$$

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 are updated for all agents for a set amount of iterations c_m in this manner, and then, the weight for each agent is updated to the maximum weight among 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.

Using this algorithm, $info_i$ is found for each agent, and given 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.

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 Figure 7. 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. The number 20 was arbitrarily chosen as a large number to ensure that the agents have enough iterations to achieve a consensus. One run of the average time it took to converge for each of the 900 iterations can be seen in Figure 8. Figure 9 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 a 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 Table 1. However, if one does not do the maximum weight sharing by setting c_m to an arbitrarily high value, the algorithm will not fully converge even given up to 40 iterations, as can be seen in Figures 10 and 11.

From Table 1, we can see 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

Algorithm 2. Consensus on direction of predator.

```

Initialize  $r_p$ 
Initialize  $c_m$ 
Measurement Phase:
for Each agent  $i$  do
    if  $\|q_p - q_i\| < r_p$  then
        Find  $info_i$  from eq. (18)
        Find  $weight_{i,d}$  from eq. (19)
    else
         $weight_i = 0$ 
    end
end
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

```

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 additional iterations to converge for each weighted voting iteration added. It 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 $c_m = 3$. By letting $c_m = 10$, it can be seen that there is about an 1.5% increase in performance compared to $c_m = 2$ but at a much higher computational cost. The error in accuracy can be attributed to the formation of the flock, as can be seen in Figure 12. 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 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, the time it takes to reach a consensus is at least two to three times the number of iterations that the

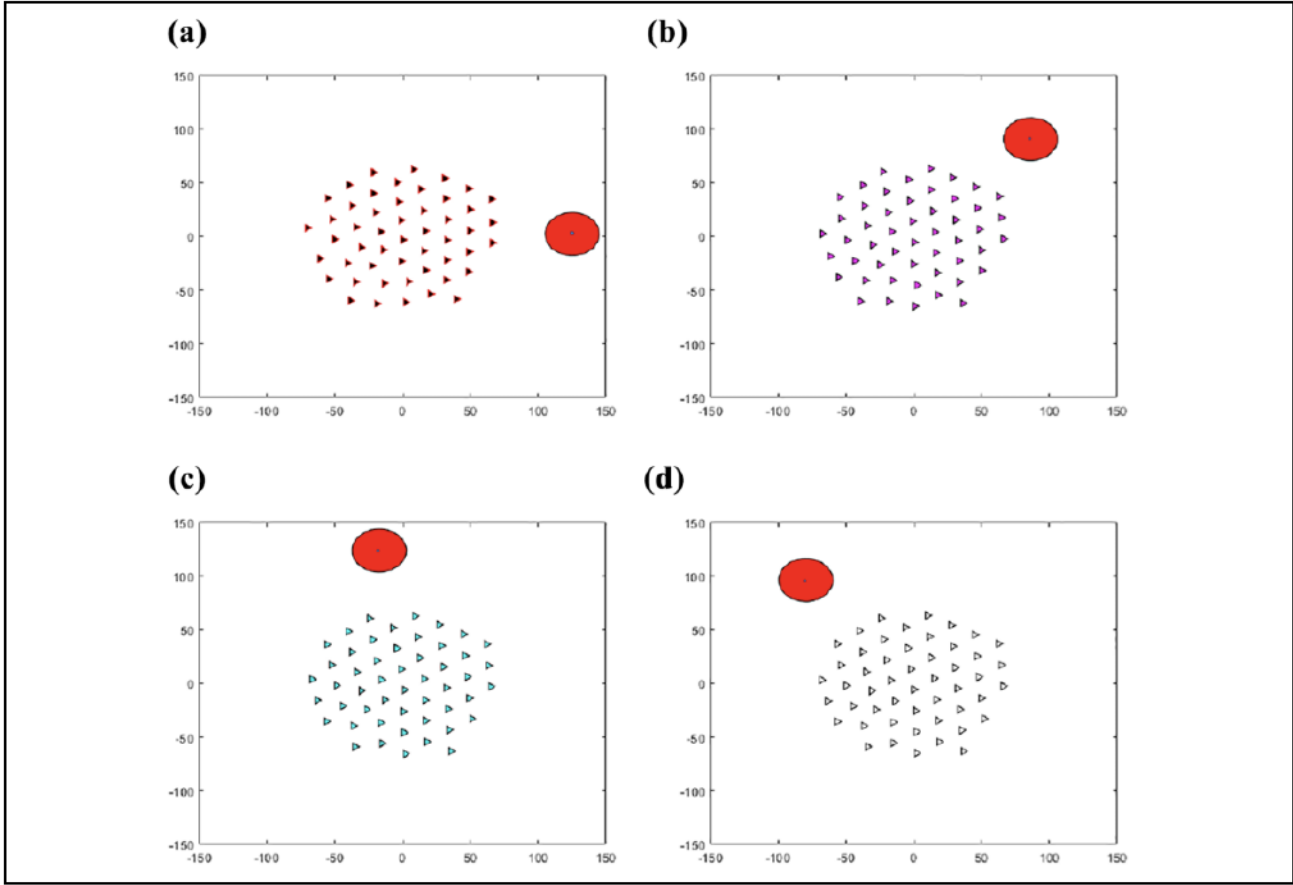


Figure 7. (a–d) Predator moving in circle around flock with consensus updating the state of the agents based on the direction of the predator.

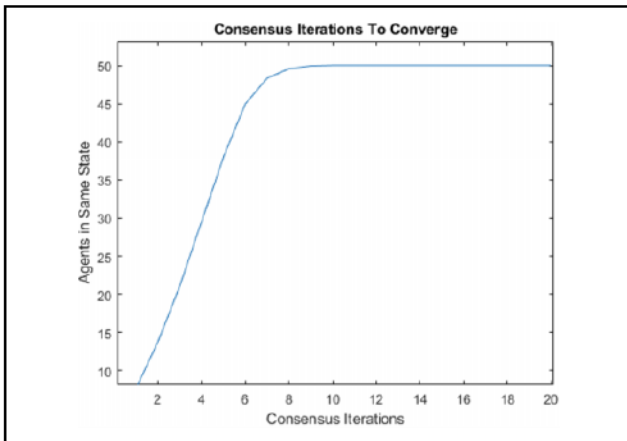


Figure 8. The number of consensus iterations for all agents to converge to the same state.

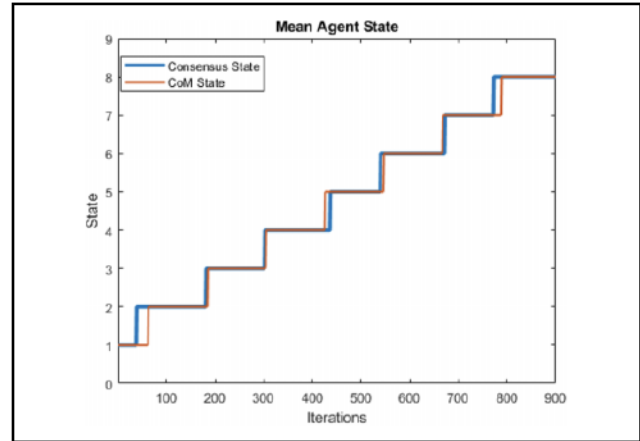


Figure 9. The average info_i for the agents in each iteration. Some states occur for a longer duration due to the shape of the flock.

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 may have a

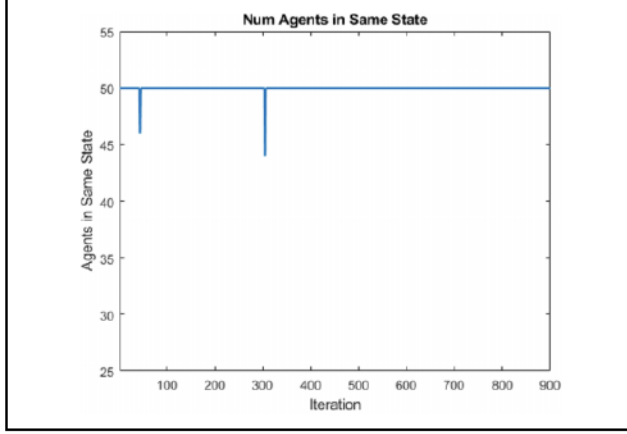
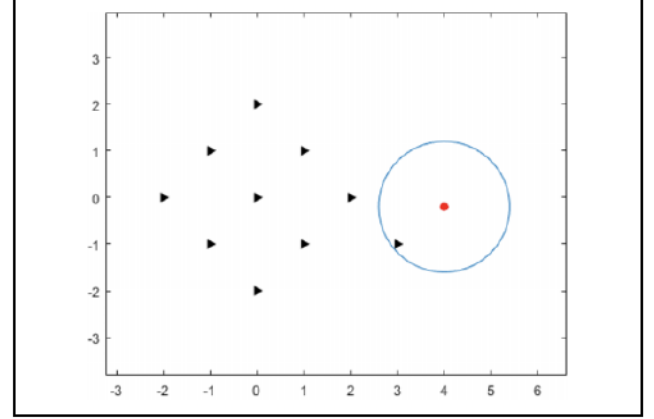
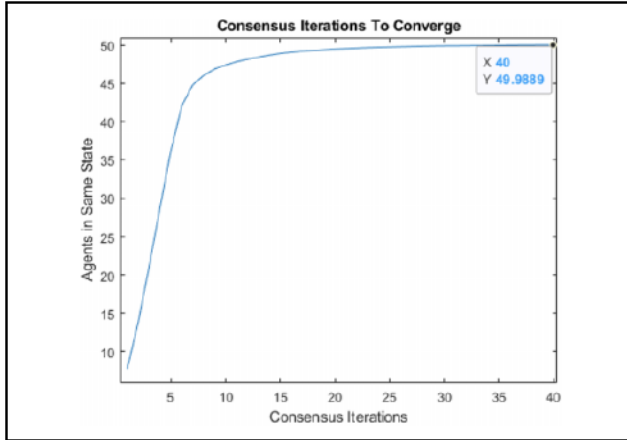
large amount of noise associated with it, so this approach was not used.

Simulation and results

In this section, we go over implementation details and results found for the hybrid learning system. We use

Table 1. The iterations to converge and number of incorrect directions found over 900 iterations. $c_m = 2$ produces the best results.Effects of change in c_m

	$c_m = 0$	$c_m = 1$	$c_m = 2$	$c_m = 3$	$c_m = 4$	$c_m = 10$
Iterations to converge	8.2	8.8	9.8	10.6	11	16.2
Incorrect directions	114 (12.7%)	75.2 (8.4%)	68.6 (7.6%)	73.8 (8.2%)	70.4 (7.8%)	54.4 (6.04%)

**Figure 10.** The number of agents in the same state when c_m is arbitrarily high. There are multiple predator positions that result in the agents not fully converging even given 40 iterations.**Figure 12.** 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 northeast causing the error seen in Table 1.**Figure 11.** The convergence of the agents state when c_m is arbitrarily high. Given 40 consensus iterations, it is not able to fully converge.

consensus to determine the direction of the predator dir_p . This state is then used in the multiagent learning for the θ table. The multiagent learning then produces an action, which is a target to flock toward 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. The simulation was developed using MATLAB. Each learning episode consists of a

certain time duration that is long enough for agents to flock to the determined target. Within each episode, the number of iterations that it is run for is determined by the time step. A smaller time step gives longer episodes but faster learning due to more communications occurring over the episode. Smaller time steps also provide for smoother flocking. Each agent is represented by plotting its position as a triangle at each time step in a two-dimensional MATLAB plot.

Simulation environment

The learning environment is set up in a manner, as shown in Figure 13, where 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 120×120 area and the predator in one of eight directions. The predator then moves toward 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 that 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.

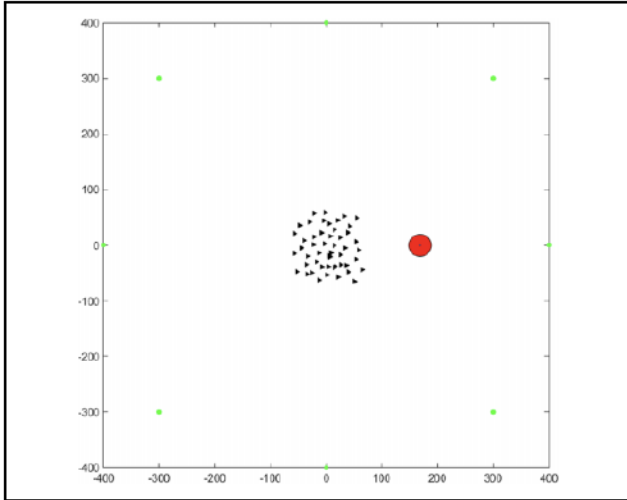


Figure 13. Initialization of an episode with agents randomly distributed.

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 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 ϵ -greedy action selection in equation (13), an ϵ value of 0.1 is chosen to allow the agents to explore other actions more quickly while not hindering the smoothness of flocking too extremely.

Results

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 eight episodes can be seen in Figures 14 and 15. We can see that the agents are able to fully converge for a single direction state in four 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 four 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.

Single predator. Six runs were conducted and averaged for the single predator environment. In Figure 16, 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 Figure 17. Some agents are in different states

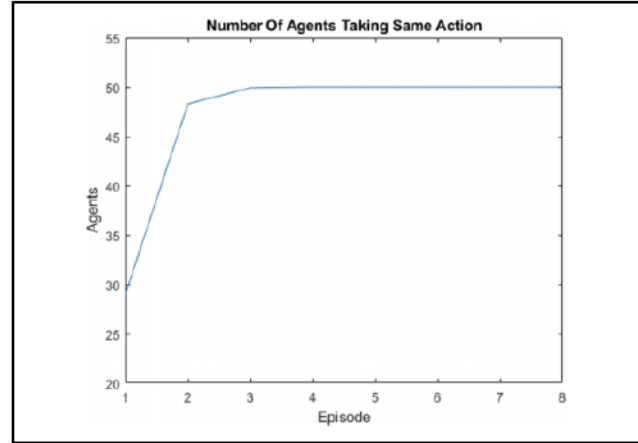


Figure 14. The number of agents choosing the same action for each episode for a single direction state by episode. The agents are able to fully converge by four episodes.

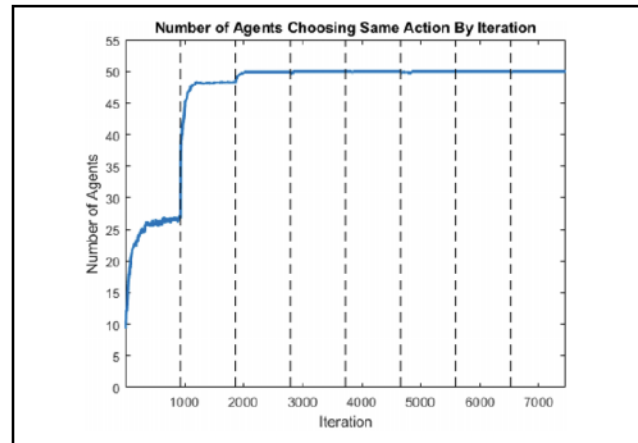


Figure 15. The number of agents choosing the same action for each episode for a single direction state by iteration with the dashed lines representing the start of an episode. It can be seen that most learning is done in the beginning of an episode while the flock is connected.

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 Figure 18, the agents are all in the same state and have learned to go to the same target in an α -lattice formation.

In Figure 19, 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 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 ϵ -greedy.

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 eight times the amount of episodes that must be performed for all



Figure 16. The number of agents choosing the same action for each episode for a single predator environment averaged over six runs. It can be seen that the agents converge by each direction being encountered six times.

direction combinations to be encountered. Unfortunately, this cannot 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 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 eight 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, and the results of which can be seen in one run in Figure 20. It can be seen that by 144 episodes, or all combinations of directions being encountered four times the agents have almost converged. By 180 episodes or each direction being seen five times, the agents have completely converged to the same action for each state.

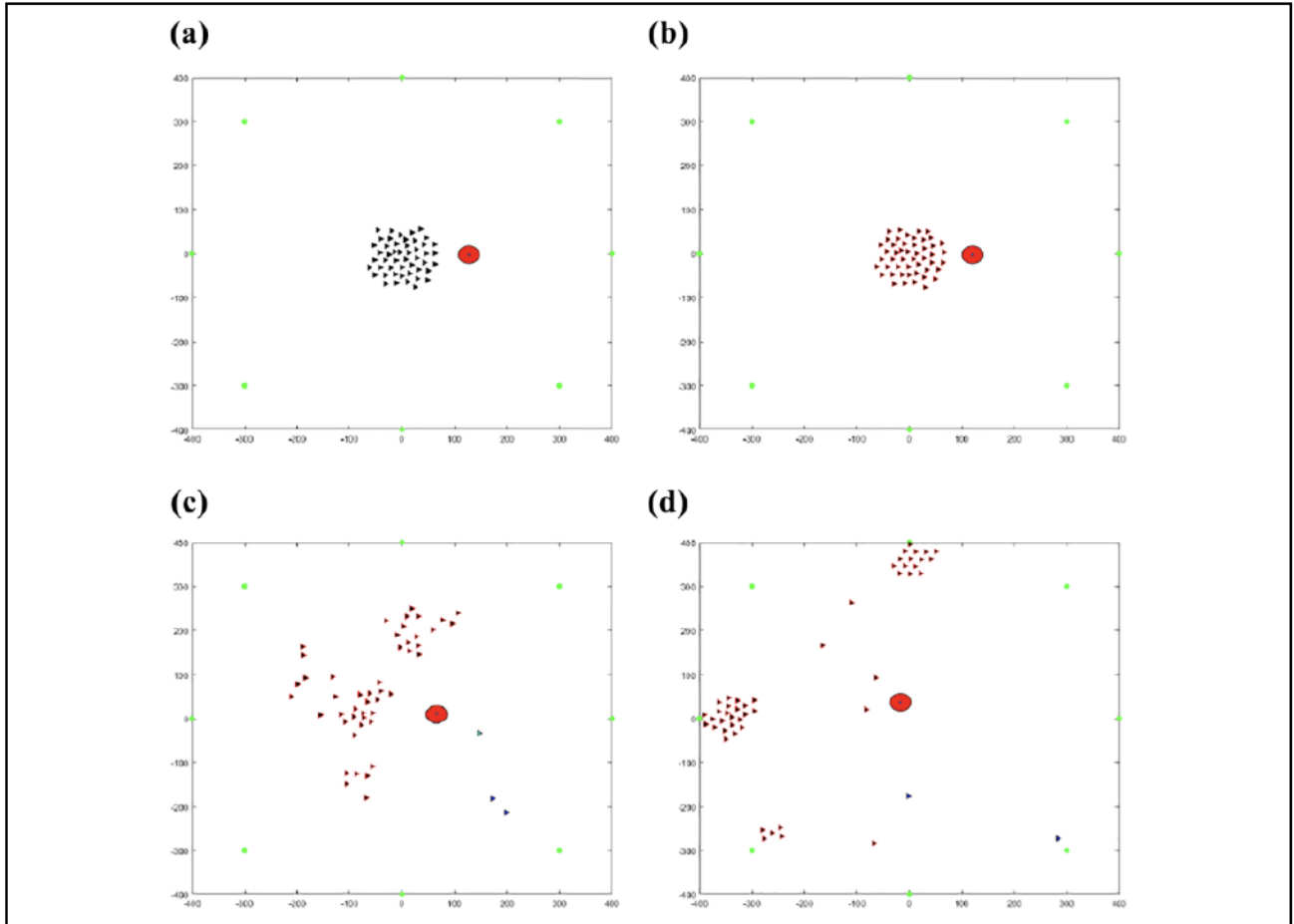


Figure 17. (a–d) Fifty agents flocking away from one predator before learning the same target.

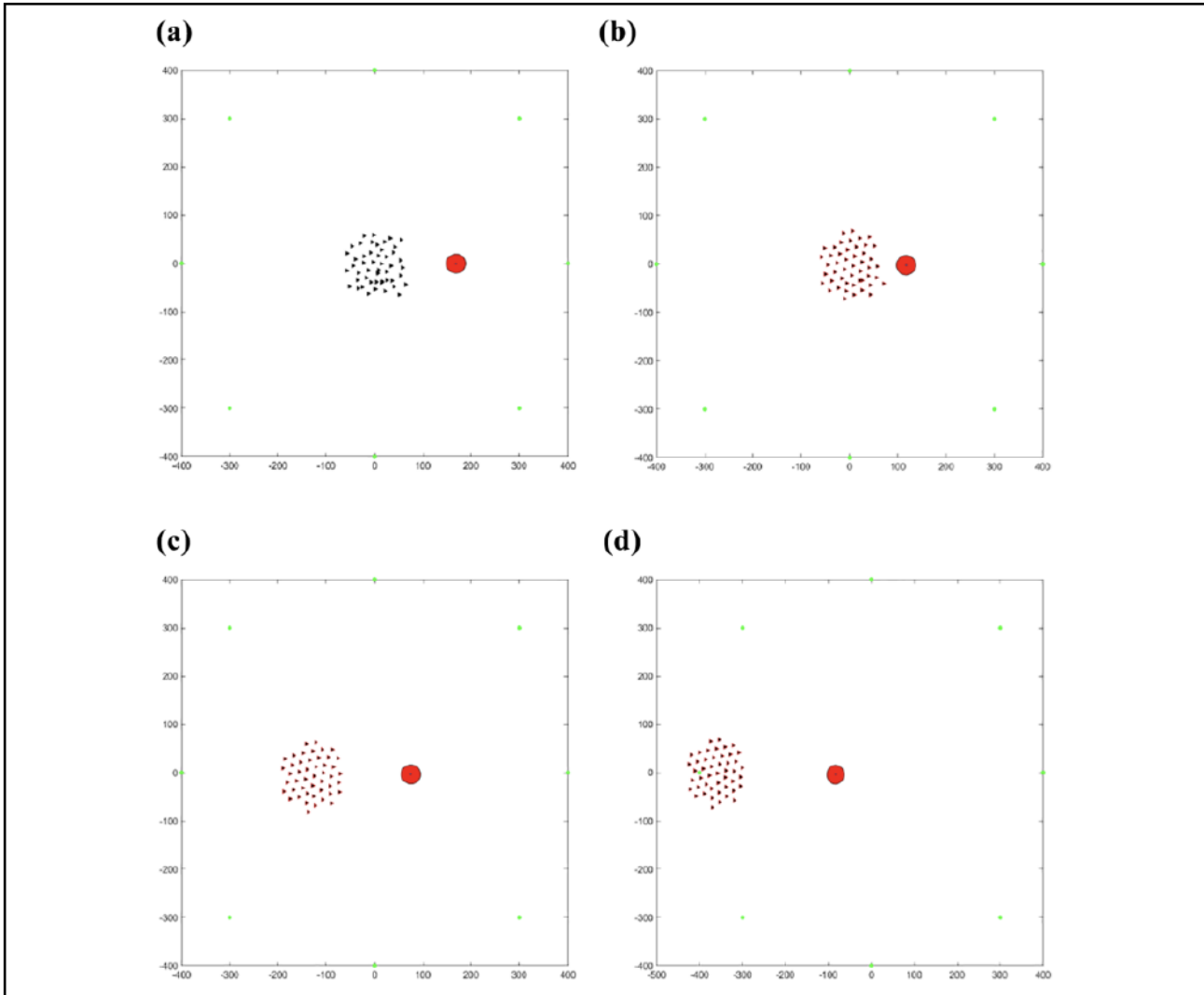


Figure 18. (a–d) Fifty agents flocking away from one predator after learning the same target.

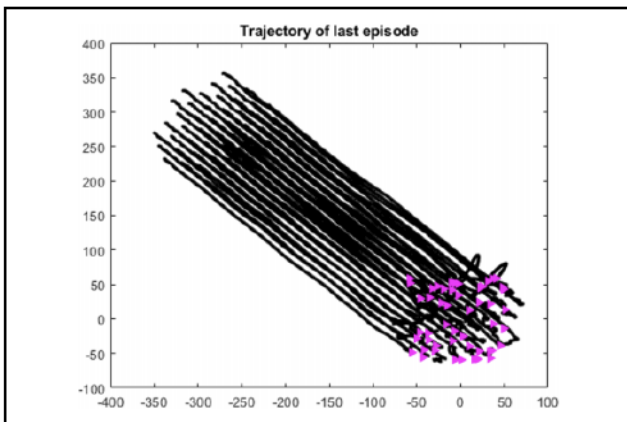


Figure 19. Trajectory the agents take in the last episode where the triangles are the initial position of the agents. The flocking can be seen to be smooth to the target after agents get into flocking formation despite the ϵ -greedy random action selection.

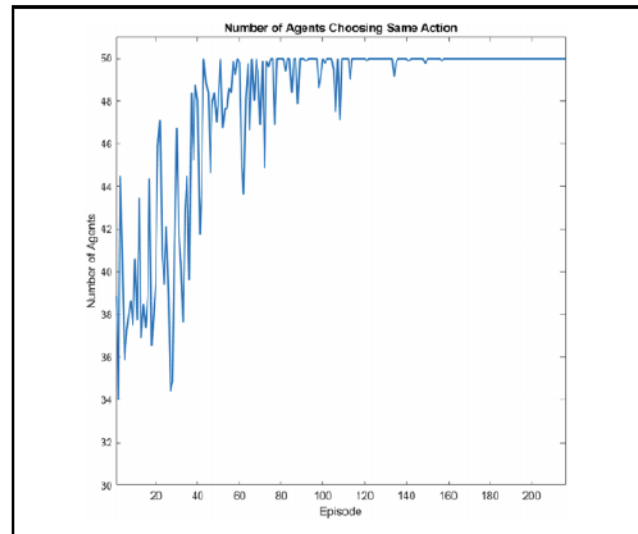


Figure 20. Number of agents choosing the same action for two predators over 216 episodes. It can be seen that after 180 episodes, all agents choose the same action for each state.

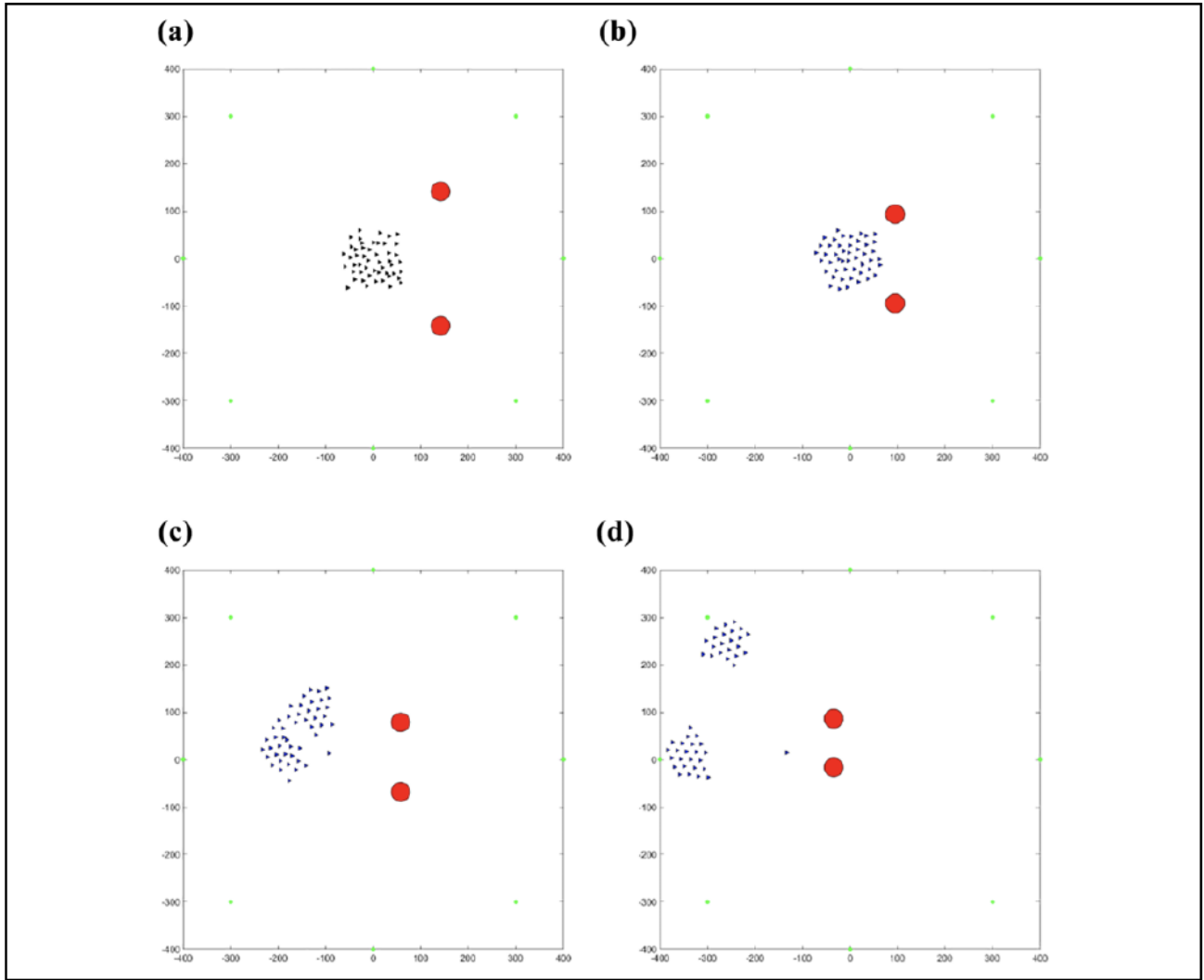


Figure 21. (a–d) Fifty agents flocking away from two predators before learning the same target.

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 Figures 21 and 22, respectively, which looks similar to that of the single predator.

Discussion

In Figure 21, it can be seen that the agents split into two subgroups. This was a fairly common occurrence for unlearned episodes caused by the communication between neighbors. In this case, a lot of the agents may have found the west direction to be the best while others may have found the northwest direction to be best. Eventually, their close neighbors would agree with them and move with them to those targets but not enough to cause the other majority to agree. In this case, the neighborhoods are pretty split between the top half and the bottom half visually with agents on the border picking one or the other depending on the ϵ -greedy learning algorithm. The best direction is then learned

in later episodes when the agents are randomly positioned so that the neighborhoods become mixed together.

Another aspect to consider is what would happen if agents were surrounding a predator. In particular, if agents were equally distributed around the predator, then the predator is not in any particular cardinal direction relative to the flock. Currently, there is no check for that occurring but that would be easily adaptable to stop the agents from moving until the predator moves into a position, where there is a cardinal direction. Alternatively, currently, the agents will move in either the first or last direction depending on implementation at which point the predator will likely obtain a direction after one time step.

Conclusion and future work

Conclusion

This article presented a hybrid system that achieves an efficient cooperative learning behavior. The system is

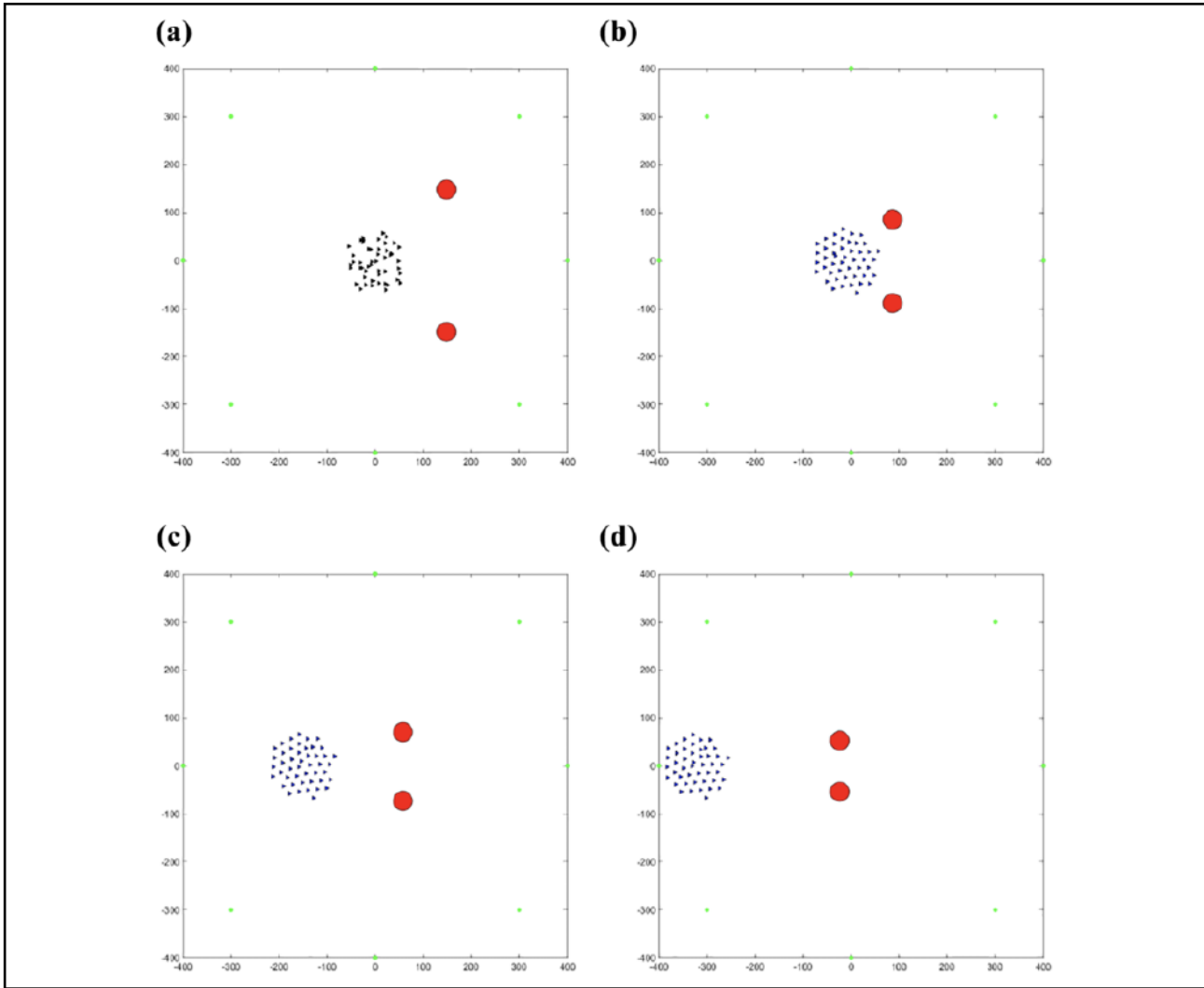


Figure 22. (a–d) Fifty agents flocking away from two predators after learning the same target.

applied to the task of escaping attacking predators while maintaining a flocking formation.

Flocking is used to move the agents in an α -lattice formation to a target location. 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.

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 multiagent 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.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This material was based upon work partially supported by the National Aeronautics and Space Administration (NASA) under grant RRR no. 80NSSC19M0170 and grant no. NNX15AI02H issued through the NVSGC-RI program under subaward no. 19-21, the RID program under subaward no. 19-29, and the NVSGC-CD program under subaward no. 18-54. This work was supported by the U.S. National Science Foundation (NSF) under grant nos NSF-CAREER: 1846513 and NSF-PFI-TT: 1919127 and the U.S. Department of Transportation, Office of the Assistant Secretary for Research and Technology (USDOT/OST-R) under grant no. 69A3551747126 through INSPIRE University Transportation Center. The views, opinions, findings, and conclusions reflected in this publication are solely those of the authors and do not represent the official policy or position of the NSF, NASA, and USDOT/OST-R.

ORCID iD

Hung Manh La  <https://orcid.org/0000-0003-2183-2634>

Supplemental material

Supplemental material for this article is available online.

References

- Jin L, Li S, La HM, et al. Dynamic task allocation in multi-robot coordination for moving target tracking: a distributed approach. *Automatica* 2019; 100: 75–81.
- Harris N, Liu S, Louis SJ, et al. A genetic algorithm for multirobot routing in automated bridge inspection. In: *Proceedings of the genetic and evolutionary computation conference companion*, Prague, Czech Republic, 2019, pp. 369–370. New York, NY, USA: Association for Computing Machinery.
- Konda R, La HM, and Zhang J. Decentralized function approximated q-learning in multi-robot systems for predator avoidance. *IEEE Robot Autom Lett* 2020; 5(4): 6342–6349.
- Pham HX, La HM, Feil-Seifer D, et al. A distributed control framework for a team of unmanned aerial vehicles for dynamic wildfire tracking. In: *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. Vancouver, BC, Canada, 24–28 September 2017, pp. 6648–6653. IEEE.
- Pham HX, La HM, Feil-Seifer D, et al. A distributed control framework of multiple unmanned aerial vehicles for dynamic wildfire tracking. *IEEE Trans Syst Man Cyber: Syst* 2020; 50(4): 1537–1548.
- Tavakoli M, Cabrita G, Faria R, et al. Cooperative multi-agent mapping of three-dimensional structures for pipeline inspection applications. *Int J Robot Res* 2012; 31(12): 1489–1503.
- Fox D, Ko J, Konolige K, et al. Distributed multirobot exploration and mapping. *Proc IEEE* 2006; 94(7): 1325–1339.
- La HM, Lim RS, Sheng W, et al. Cooperative flocking and learning in multi-robot systems for predator avoidance. In: *2013 IEEE international conference on cyber technology in automation, control and intelligent systems*. Nanjing, China, 26–29 May 2013, pp. 337–342. IEEE.
- La HM, Lim R, and Sheng W. Multirobot cooperative learning for predator avoidance. *IEEE Trans Control Syst Technol* 2014; 23(1): 52–63.
- Muñoz F, Espinoza Quesada ES, La HM, et al. Adaptive consensus algorithms for real-time operation of multi-agent systems affected by switching network events. *Int J Robust Nonlinear Control* 2017; 27(9): 1566–1588.
- Dang AD, La HM, Nguyen T, et al. Formation control for autonomous robots with collision and obstacle avoidance using a rotational and repulsive force-based approach. *Int J Adv Robot Syst* 2019; 16(3): 1729881419847897.
- Nguyen H and La HM. Review of deep reinforcement learning for robot manipulation. In: *2019 Third IEEE international conference on robotic computing (IRC)*. Naples, Italy, 25–27 February 2019, pp. 590–595. IEEE.
- Nguyen MT, La HM, and Teague KA. Compressive and collaborative mobile sensing for scalar field mapping in robotic networks. In: *2015 53rd annual allerton conference on communication, control, and computing (allerton)*, Monticello, IL, 2015, pp. 873–880.
- La HM. (2020). Multi-Robot Swarm for Cooperative Scalar Field Mapping. In: Management Association, I. (Eds.), *Robotic Systems: Concepts, Methodologies, Tools, and Applications*. IGI Global, 2020, pp. 208–223.
- La HM, Sheng W, and Chen J. Cooperative and active sensing in mobile sensor networks for scalar field mapping. *IEEE Trans Syst Man Cybernet: Syst* 2015; 45(1): 1–12.
- Nguyen MT, La HM, and Teague KA. Collaborative and compressed mobile sensing for data collection in distributed robotic networks. *IEEE Trans Control Network Syst* 2018; 5(4): 1729–1740.
- Le C, Pham AQ, La HM, et al. A multi-robotic system for environmental dirt cleaning. In: *2020 IEEE/SICE international symposium on system integration (SII)*, Honolulu, HI, USA, 2020, pp. 1294–1299.
- Connell D and La HM. Extended rapidly exploring random tree-based dynamic path planning and replanning for mobile robots. *Int J Adv Robot Syst* 2018; 15(3): 1729881418773874.

19. Connell D and La HM. Dynamic path planning and replanning for mobile robots using RRT. In: *2017 IEEE international conference on systems, man, and cybernetics (SMC)*. Banff, AB, Canada, 5–8 October 2017, pp. 1429–1434. IEEE.
20. Dang AD, La HM, and Horn J. Distributed formation control for autonomous robots following desired shapes in noisy environment. In: *2016 IEEE international conference on multisensor fusion and integration for intelligent systems (MFI)*. Baden-Baden, Germany, 19–21 September 2016, pp. 285–290. IEEE.
21. Nguyen T and La HM. Formation control of multiple rectangular agents with limited communication ranges. In: G Bebis, R Boyle, B Parvin, et al. (eds) *Advances in visual computing*. Cham: Springer International Publishing, 2014, pp. 915–924. ISBN 978-3-319-14364-4
22. Han T, La HM, and Dinh BH. Flocking of mobile robots by bounded feedback. In: *2016 IEEE international conference on automation science and engineering (CASE)*. Fort Worth, TX, USA, 21–25 August 2016, pp. 689–694. IEEE.
23. Hung PD, La HM, and Ngo TD. Adaptive hierarchical distributed control with cooperative task allocation for robot swarms. In: *2020 IEEE/SICE international symposium on system integration (SII)*. Honolulu, HI, USA, 12–15 January 2020, pp. 1300–1305. IEEE.
24. Jafari M, Sengupta S, and La H. Adaptive flocking control of multiple unmanned ground vehicles by using a UAV. *Adv Visual Computing* 2015; 9475: 628–637.
25. Pham HX, La H, Feil-Seifer D, et al. Performance comparison of function approximation-based Q learning algorithms for autonomous UAV navigation. In: *The 15th International Conference on Ubiquitous Robots (UR)*, Hawaii, USA, 26–30 June 2018, pp. 1–3.
26. Pham HX, La HM, Feil-Seifer D, et al. Reinforcement learning for autonomous UAV navigation using function approximation. In: *2018 IEEE international symposium on safety, security, and rescue robotics (SSRR)*. Philadelphia, PA, USA, 6–8 August 2018, pp. 1–6. IEEE.
27. Rahimi M, Gibb S, Shen Y, et al. A comparison of various approaches to reinforcement learning algorithms for multi-robot box pushing. In: Fujita H, Nguyen DC, and Vu NP (eds) *Advances in engineering research and application*. Cham: Springer International Publishing, 2018, pp. 16–30. ISBN 978-3-030-04792-4
28. Roberts G. Why individual vigilance declines as group size increases. *Animal Behav* 1996; 51: 1077–1086. DOI:10.1006/anbe.1996.0109.
29. Krause J, Ruxton GD, and Rubenstein D. Is there always an influence of shoal size on predator hunting success? *J Fish Biol* 1998; 52(3): 494–501.
30. Milinski M and Heller R. Influence of a predator on the optimal foraging behaviour of sticklebacks (*Gasterosteus aculeatus* L.). *Nature* 1978; 275(5681): 642–644.
31. Hahn C, Phan T, Gabor T, Belzner L, and Linnhoff-Popien C. Emergent escape-based flocking behavior using multi-agent reinforcement learning. In: *Artificial life conference proceedings*, July 2019, pp. 598–605. Cambridge, MA: MIT Press.
32. Sunehag P, Lever G, Liu S, et al. Reinforcement learning agents acquire flocking and symbiotic behaviour in simulated ecosystems. In: *Artificial life conference proceedings*, 2019, pp. 103–110.
33. Olfati-Saber R. Flocking for multi-agent dynamic systems: algorithms and theory. *IEEE Trans Autom Control* 2006; 51(3): 401–420.
34. Singh P, Tiwari R, and Bhattacharya M. Navigation in multi robot system using cooperative learning: a survey. In: *2016 international conference on computational techniques in information and communication technologies (ICCTICT)*. New Delhi, India, 11–13 March 2010, pp. 145–150. IEEE.
35. La HM and Sheng W. Flocking control of multiple agents in noisy environments. In *2010 IEEE international conference on robotics and automation*. Anchorage, AK, USA, 3–7 May 2010, pp. 4964–4969. IEEE.
36. La HM and Sheng W. Flocking control of a mobile sensor network to track and observe a moving target. In: *2009 IEEE international conference on robotics and automation*. Kobe, Japan, 12–17 May 2009, pp. 3129–3134. IEEE.
37. La HM, Nguyen TH, Nguyen CH, et al. Optimal flocking control for a mobile sensor network based a moving target tracking. In: *2009 IEEE international conference on systems, man and cybernetics*. San Antonio, TX, USA, 11–14 October 2009, pp. 4801–4806. IEEE.
38. Reynolds CW. Flocks, herds and schools: a distributed behavioral model. *SIGGRAPH Comput Graph* 1987; 21(4): 25–34.
39. Durve M, Peruani F, and Celani A. Learning to flock through reinforcement. *Phys Rev E* 2019; 102: 012601.
40. Semnani SH, Liu H, Everett M, et al. Multi-agent motion planning for dense and dynamic environments via deep reinforcement learning. In: *IEEE Robotics and Automation Letters*, (vol. 5, no. 2), April 2020, pp. 3221–3226.
41. Mehmood U, Roy S, Grosu R, Smolka SA, Stoller SD, and Tiwari A. Neural flocking: MPC-based supervised learning of flocking controllers. In: *International conference on foundations of software science and computation structures*, 25 April 2020, pp. 1–16. Cham: Springer.
42. Martinez-Clark R, Cruz-Hernández C, Pliego-Jimenez J, et al. Control algorithms for the emergence of self-organized behaviours in swarms of differential-traction wheeled mobile robots. *Int J Adv Robot Syst* 2018; 15(6): 1729881418806435.
43. Shiliang S, Ting W, Chen Y, et al. Distributed fault detection and isolation for flocking in a multi-robot system with imperfect communication. *Int J Adv Robot Syst* 2014; 11.
44. Jiang L and Zhang R. Stable formation control of multi-robot system with communication delay. *Int J Adv Robot Syst* 2012; 9(1): 4.
45. La HM and Sheng W. Adaptive flocking control for dynamic target tracking in mobile sensor networks. In: *2009 IEEE/RSJ international conference on intelligent robots and systems*. St. Louis, MO, USA, 10–15 October 2009, pp. 4843–4848. IEEE.

46. Afifi AM, Alhosainy OH, Elias CM, et al. Deep policy-gradient based path planning and reinforcement cooperative Q-learning behavior of multi-vehicle systems. In: *2019 IEEE international conference on vehicular electronics and safety (ICVES)*. Cairo, Egypt, 4–6 September 2019, pp. 1–7. IEEE.
47. Sutton RS and Barto AG. *Reinforcement learning: an introduction*. Cambridge, MA, USA: MIT Press, 2018.
48. Busoniu L, Babuska R, and De Schutter B. A comprehensive survey of multiagent reinforcement learning. *IEEE Trans Syst Man Cyber Part C (Appl Rev)* 2008; 38(2): 156–172.
49. Busoniu L, Babuska R, De Schutter B, et al. *Reinforcement learning and dynamic programming using function approximators*. Cleveland, OH, USA: CRC Press, 2010. ISBN 1439821089.
50. Geramifard A, Walsh TJ, Tellex S, et al. A tutorial on linear function approximators for dynamic programming and reinforcement learning. *Foundation and Trends[®] in Machine Learning* 2013; 6(4): 375–451.
51. Zhang K, Yang Z, and Basar T. Networked multi-agent reinforcement learning in continuous spaces. In: *2018 IEEE conference on decision and control (CDC)*. Miami Beach, FL, USA, 17–19 December 2018, pp. 2771–2776. IEEE.
52. Xu Y, Zhang W, Liu W, et al. Multiagent-based reinforcement learning for optimal reactive power dispatch. *IEEE Trans Syst Man Cyber Part C (Appl Rev)* 2012; 42(6): 1742–1751.
53. Nowzari C, Garcia E, and Cortés J. Event-triggered communication and control of networked systems for multi-agent consensus. *Automatica* 2019; 105: 1–27.
54. Olfati-Saber R, Fax JA, and Murray RM. Consensus and cooperation in networked multi-agent systems. *Proc IEEE* 2007; 95(1): 215–233.
55. Kokiopoulou E and Frossard P. Distributed classification of multiple observation sets by consensus. *IEEE Trans Signal Process* 2011; 59(1): 104–114.
56. La HM and Sheng W. Distributed sensor fusion for scalar field mapping using mobile sensor networks. *IEEE Trans Cyber* 2013; 43(2): 766–778.
57. La HM and Sheng W. Cooperative sensing in mobile sensor networks based on distributed consensus. In: Drummond OE (ed) *Signal and data processing of small targets 2011*, Vol. 8137. SPIE, 2011, pp. 328–341.
58. Kaelbling LP, Littman ML, and Moore AW. Reinforcement learning: a survey. *J Art Intell Res* 1996; 4: 237–285.