MANCHESTER 1824
The University of Manchester

The University of Manchester Research

Self-Organised Swarm Flocking with Deep Reinforcement Learning

DOI:
10.1109/ICARA51699.2021.9376509

Document Version
Accepted author manuscript

Link to publication record in Manchester Research Explorer

Citation for published version (APA):
Bezcioglu, M., Lennox, B., & Arvin, F. (2021). Self-Organised Swarm Flocking with Deep Reinforcement Learning. In *2021 International Conference on Automation, Robotics and Applications, ICARA 2021* (pp. 226-230). [9376509] (2021 International Conference on Automation, Robotics and Applications, ICARA 2021). https://doi.org/10.1109/ICARA51699.2021.9376509

Published in:
2021 International Conference on Automation, Robotics and Applications, ICARA 2021

Citing this paper
Please note that where the full-text provided on Manchester Research Explorer is the Author Accepted Manuscript or Proof version this may differ from the final Published version. If citing, it is advised that you check and use the publisher's definitive version.

General rights
Copyright and moral rights for the publications made accessible in the Research Explorer are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Takedown policy
If you believe that this document breaches copyright please refer to the University of Manchester's Takedown Procedures [http://man.ac.uk/04Y6Bo] or contact uml.scholarlycommunications@manchester.ac.uk providing relevant details, so we can investigate your claim.

OPEN ACCESS

Download date:09. Jun. 2022

Self-Organised Swarm Flocking with Deep Reinforcement Learning

Mehmet B. Bezcioglu, Barry Lennox, and Farshad Arvin
Swarm & Computational Intelligence Lab (SwaCIL)
Department of Electrical and Electronic Engineering, The University of Manchester, UK
{mehmet.bezcioglu, barry.lennox, farshad.arvin}@manchester.ac.uk

***Abstract*—Optimising a set of parameters for swarm flocking is a tedious task as it requires hand-tuning of the parameters. In this paper, we developed a self-organised flocking mechanism with a swarm of homogeneous robots. The proposed mechanism used deep reinforcement learning to teach the swarm to perform the flocking in a continuous state and action space. Collective motion was represented by a self-organising dynamic model that is based on linear spring-like forces between self-propelled particles in an active crystal. We tuned the inverse rotational and translational damping coefficients of the dynamic model for swarm populations of $N \in \{25, 100\}$ robots. We study the application of reinforcement learning in a centralised multi-agent approach, where we have a global state space matrix that is accessible by actor and critic networks. Furthermore, we showed that our method could train the system to flock regardless of the sparsity of the swarm population, which is a significant result.**

***Index Terms*—Swarm Robotics, Reinforcement Learning, Self-organised Flocking, Multi-agent Learning.**

I. Introduction

Swarm robotics is the study of coordination control in a large number of simple robots using collective mechanisms that are mainly inspired by nature [1]. Many bio-inspired swarm behaviours have been successfully implemented by robotic swarms, e.g. aggregation [2], flocking [3], exploration [4], etc. Flocking is one of the important swarm behaviours that took inspiration from the collective motion of social animals [5]. There are many real-world applications for flocking, e.g. underwater exploration [6] and formation control in autonomous cars [7].

One of the first works in flocking was proposed by Reynolds [8], who defined three behaviours that are important for a swarm of agents to form a flock. Reynolds termed these behaviours as *collision avoidance*, *velocity matching*, and *flock centering*, and demonstrated that flocking can be achieved if each individual has access to the range and orientation of their respective local neighbours. Reynolds' boids simulation was inspired by the mathematical model proposed by Vicsek et al. [9] to describe the phenomena of collective motion from a physics point of view. They modeled swarm flocking in the presence of perturbations and reported a kinetic phase transition from 'disordered to ordered' as the number of perturbations were decreased. Further, Toner and Tu [10] developed theoretical work in flocking from thermodynamics and control points of view. The aforementioned studies were the first works in the field and led to the emergence of other works that relied on effective information exchange between agents.

One such work was [11], which introduced the self-propelled collective motion (CM) mechanism, termed as Active Elastic Sheet (AES), that is based on a two-dimensional active crystal. Agents are coupled by linear attraction-repulsion forces, where each agent's force is a linear combination of the elastic forces that exist between the agent and its set of neighbours. Another study [12] proposed a modified version of the CM mechanism introduced by [11] that was based on spring-like forces between particles that lead to self-organisation of a group of individuals. This method relied on the exchange of relative positions, but not the relative headings between robots.

There are a few challenges in implementing swarm flocking one of which is the requirement of manual tuning of the parameters to their optimal values. This study aims to address this challenge by proposing an automatic tuning of flocking parameters.

In this paper, we proposed a novel flocking mechanism with combination of bio-inspired collective motion and reinforcement learning. The proposed method is based on convolutional Deep Deterministic Policy Gradient (DDPG) algorithm. We chose the elastic model proposed by [12] because their work relied on the relative exchange of positions instead of headings. To the best of our knowledge, this is the first work to study the application of reinforcement learning on an elasticity-based collective motion dynamic model. We trained the model in a particle simulator and showed that DRL can be used to train a group of robots to achieve flocking successfully whilst eliminating the tedious task of hand-tuning CM parameters.

II. Multi-agent Reinforcement Learning

With advances in deep learning, Mnih et al. [13] proposed a deep-Q network for an agent that approximates state-value function in continuous state spaces by taking in high-dimensional sensory inputs and outputting low-dimensional discrete actions for the agent. Their formulation combined reinforcement learning theory with deep learning to select actions that maximise the approximated optimal policy using a

This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) projects RAIN (EP/R026084/1) and RNE (EP/P01366X/1).

loss function. Extensions of DQN-based algorithms have been shown to work on enormous action spaces well, as proposed in [14] and [15]. Learned policies can be in the form of stochastic, or deterministic actions. Stochastic policies sample probability distribution to output an action as $a = P[s|a;\theta]$, where deterministic policies output an action $a = \mu(s)$, which is always deterministic. In [16], David et al. showed that deterministic policies exist and can be learned more efficiently than stochastic policy gradients. Following this idea, [17] proposed DDPG that combined the policy gradient method proposed in [16], with the DQN approach proposed in [13]. Hence, we propose our DRL setting with DDPG as it is designed for continuous action spaces, which satisfies the requirements of our problem formulation.

Multi-agent reinforcement learning for cooperative swarm intelligence is a difficult task, mainly because each agent has partial access to the global state of the swarm, and also due to the curse of dimensionality for a large number of agents in a swarm. Hüttenrauch et al. [18] explored the incorporation of actor-critic approach, where critic has access to the swarm's global state, but actors are based on the locally observed sensory data. They used a variant of DDPG for the simulation of 2D robots. However, their method was decentralised and did not explore the applicability of a centralised application of reinforcement learning to a biologically inspired dynamic model. In [19], further use of mean embedding distributions was introduced, where each agent is considered a sample. They studied global and local cases with a communication protocol proposed for the local case in pursuit evasion and rendezvous problems in swarm systems. In a followup study [20], they proposed a leader-follower mechanism by using inverse RL to recover unknown reward functions in a flock of birds.

III. Collective Motion

In this study, we used Active Elastic Sheet (AES), a dynamic model, to define our flocking mechanism [11]. AES is inspired by nature and provides interaction between agents in an attraction-repulsion manner by having spring-like forces between neighbours, without the exchange of orientation information. This is a true reflection of nature; a flock of birds does not have access to their neighbours orientation.

A. Active Elastic Sheet Dynamics

Active elastic sheet (AES) consists of a force for each robot that influences the linear and angular velocities of each robot in the flock. It consists of summation over the linear spring-like interactions between each robot and its neighbours to compute the new linear and angular velocities of the robot, with noise added. A more detailed explanation of the mechanics of AES can be found in [12].

Individual linear and rotational velocities are given as,

$$\dot{\mathbf{x}}_\mathbf{i}(t) = \Big(v_0(t) + \alpha[(\mathbf{f_i}(t) + D_r\hat{\epsilon}_\mathbf{r}(t)).\hat{\mathbf{n}}_\mathbf{i}(t)]\Big)\hat{\mathbf{n}}_\mathbf{i}(t)\ , \quad (1)$$

$$\dot{\theta}_i(t) = \beta\{[\mathbf{f_i}(t) + D_r\hat{\epsilon}_r(t)]\hat{\mathbf{n}}_\mathbf{i}^\perp(t)\} + D_\theta\epsilon_\theta(t)\ , \quad (2)$$

where $\hat{n}_i(t) = \begin{bmatrix} \cos\theta_i(t) \\ \sin\theta_i(t) \end{bmatrix}$ is the heading vector and $\hat{\mathbf{n}}_\mathbf{i}^\perp(t)$ is the vector perpendicular to it.

Force vector for the flock can be computed by the summation of the individual force vectors for each individual *i* over its respective neighbours in set *S*, where *j* is a member of set *S*,

$$\mathbf{f_i}(t) = \sum \frac{-k}{l_{ij}}(\|\mathbf{r_{ij}}(t)\| - l_{ij})\frac{\mathbf{r_{ij}}(t)}{\|\mathbf{r_{ij}}(t)\|}\ , \quad (3)$$

where $\mathbf{r_{ij}}(t) = \mathbf{x_i}(t) - \mathbf{x_j}(t)$ is defined as the relative position vector of neighbour *j* in frame *i*, l_{ij} is defined as the desired equilibrium spring length that the model should maintain. We used the Euler integration method.

As stated above, when the angle between the force and agent's heading is approximately 0°, the dot product is equal to $\|f\|$, which results mostly in linear translation. When the angle between the desired direction and agent's heading is 90°, the dot product results mostly in the rotation of the agent, as this is equivalent to the dot product of f and $n_i^\perp$ being closer to $\|f\|$. This means that the more the agent's orientation is pointing in the desired direction, the more the linear translation, the less the rotation. Similarly, the more the vector perpendicular to the orientation of the agent is pointing in the direction of the desired movement, the more the rotation and the agent stays at the same (x, y) positions. Ultimately, this means that to have a rapid convergence of the CM model, we need to ensure each agent's orientation is as close as possible to the desired direction of motion for each respective agent. This brings us to the next topic, the degree of alignment, and its use in optimisation.

B. Degree of Alignment

The degree of alignment for agent i is ψ_i that is defined as the dot product of the desired direction movement and the heading of the agent, which is equal to 1 at maximum.

$$\psi_i = |\cos\gamma_i| = \frac{\mathbf{f_i}.\hat{\mathbf{n}}_\mathbf{i}}{\|\mathbf{f_i}\|} \quad (4)$$

Alignment of the entire flock, ψ, can be calculated by the summation of each individual, and then by averaging the sum, as shown below:

$$\psi = \frac{1}{N}\|\sum_{i=1}^{N}\psi_i\| \quad (5)$$

The degree of alignment for an agent can be 1 at maximum and 0 at minimum, with 0 indicating no alignment between the unit force vector and unit heading vector of the agent. For the fastest rate of convergence, we need the degree of alignment of the entire flock to be 1, i.e. we need the force vector and heading vector of each pointing in the same direction to avoid the individual from rotating. This is because if an agent keeps rotating it will take a longer time to move in the desired direction until the dot product of the desired direction and the heading of the agent equate to 1. Hence, understanding the influence of the degree of alignment of the flock is important for the convergence of the algorithm.

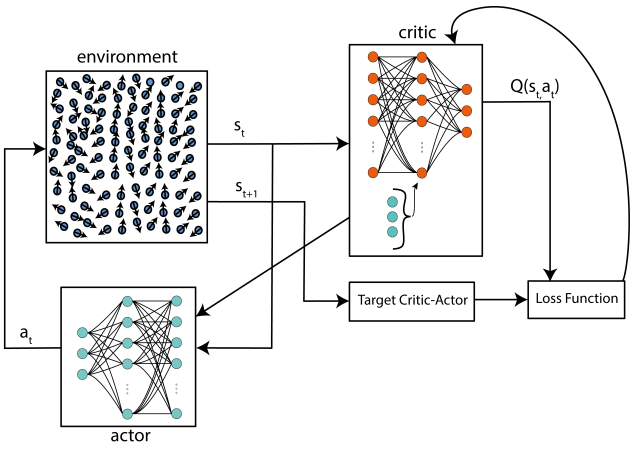

Fig. 1. Architecture overview of DDPG with multi-agent learning. Environment constitutes of *N* robots, actor produces the action a_t, which is observed as s_t. Critic provides the actor with action gradients.

C. Convergence

Raoufi et al. [12] suggested optimising two objectives for converging to the solution,

1) To minimise the sum of individual forces in the swarm
2) To maximise the rate of convergence of the solution, which is achieved when the individuals are aligned with their respective desired direction of motion at each time step.

In our study, we use these two objectives to design our reinforcement learning solution for a self-organised swarm flocking dynamic system.

IV. Swarm Flocking

A. Problem Formulation

Our state-space is a high dimensional continuous vector space. Thus, our actor and critic networks begin with convolutional layers in their first few layers. This enables us to reduce the higher-dimensional state space to a low-dimensional representation which is useful for combining the actions with the intermediate layers in the critic.

State: The state-space of a swarm of robots is represented as a combination of three matrices, where each matrix is composed of x, y and θ values of each robot in the swarm. Each agent has a spot in the global state matrix but with pixel values replaced with pose and orientation of each robot. In our case, the first channel is *x*-, the second channel is *y*-, and the last channel is θ values of each robot. We implemented two swarm population cases, with $N \in \{25, 100\}$ robots, where the former is represented as 5×5, and the latter as 10×10 grids. Our states are normalised channel-wise to have a mean of zero and a variance of one to ensure speed up the training of both networks.

Action: We designed our action space with two scenarios – individual and collective. *Individual* action involves each robot in the swarm having its own set of parameters $\{k, \beta, \alpha\}$ and *collective* action involves one set of parameters for CM model that is being shared across the swarm. Hence, for the collective action case, our action set that is applied to CM is composed of 3 parameters, corresponding to $\{k, \beta, \alpha\}$. With the individual-wise action case, each robot has its own set of $\{k, \beta, \alpha\}$ parameters, hence, our actions are a set of $3N$ parameters. To ensure the dynamics of the CM converges, we tune the $\{k, \beta, \alpha\}$ parameters, which in turn influences the heading and force vectors appropriately. These are simply weights given to the dot product of force and heading vectors. The last layer of the actor-network is passed through a `tanh` layer to ensure symmetry in actions. We found out that when the $\{k, \beta, \alpha\}$ parameters become negative, the flock fails to converge. Hence, we shift the output of the `tanh` layer to ensure values are between 0 and 1. We also tried using sigmoid, since this is what a `sigmoid` does, but we obtained better performance with `tanh`. Our action space scaling vector is given as $\{2.0, 3.0, 0.07\}$, corresponding to scaling boundary for $\{k, \beta, \alpha\}$.

Reward: Reward design is composed of the degree of alignment and the global force for the entire swarm, which is obtained by summing over the individual forces for each robot in the swarm, given in Eq. (7). As explained in Section 3, the degree of alignment needs to be maximised, and the global force is to be minimised. Hence, we formulated a reward function that is the negated global force function with a weighted sum of the degree of alignment and the mean of actor output. We then maximise this reward to ensure that a swarm of flocking robots is formed, and the rate of formation is maximised. We incorporated actor-network output in the design of the reward function because during learning actor can sometimes output values around zero, which halts efficient learning. This happens when the actor outputs a zero action, the gradient of which is then multiplied with the gradient of the critic to be fed back during backpropagation, which causes a deadlock in learning, as is explained by the work proposed in [21]. Hence, our reward function is given in 6 as,

$$R(F_g, \psi) = -\omega_F F_g + \omega_\psi \psi + \omega_a \frac{1}{N_a} \sum_{j=1}^{N_a} a_j \ , \qquad (6)$$

where F_g is the global force obtained as the norm of the sum of individual forces of each robot in a swarm of *N* robots,

$$F_g = \frac{1}{N} \| \sum_{i=1}^{N} \mathbf{f_i} \| \qquad (7)$$

and ψ is the degree of alignment of the swarm.

B. System Architecture

As shown in Figure 1, the first layers of our actor and critic networks were composed of convolutional layers, the output of which was flattened out and fed as input into the three fully connected layers. For both networks, the first fully connected layer was composed of 200 neurons, the second layer of 200 neurons, and the third layer of 1 neuron for the critic, which is the estimated Q-value, and N or $3N$ for the actor depending on the scenario being studied. The first five layers had ReLU

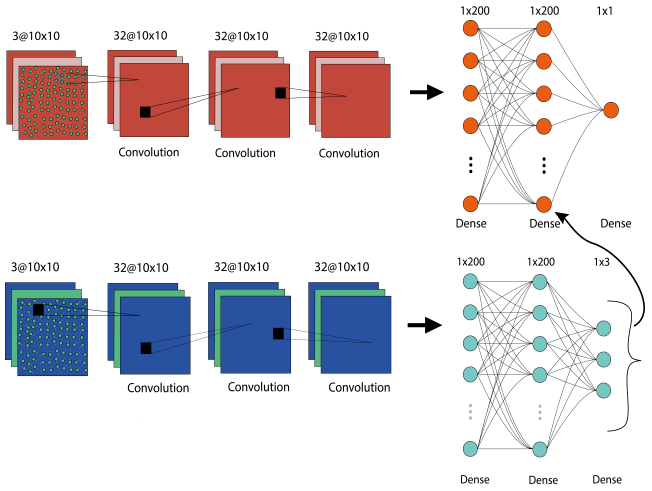

Fig. 2. Actor (bottom) and critic (top) network architectures.

as an activation function, with the sixth layer having a linear activation function. The actions were not fed until the fifth layer of the network.

We used a filter size of 3 for the first convolution layer and 2 for the following convolution layers. The final action outputs were then scaled with their respective values. Our system architecture and parameter selection were chosen empirically, as it enabled us to explore with various settings for our experimentation.

C. Weight Initialisation

For the critic network, the weight initialisation for the first 5 layers was drawn from a uniform distribution with a standard deviation of $1/\sqrt{fan\ in}$ and mean of zero where *fan in* is the number of input neurons from the previous layer. The last layer was drawn from a uniform distribution with a standard deviation of $3e^{-4}$ and mean zero. For the actor-network, the weight initialisation for the first 5 layers was drawn from the uniform distribution with a standard deviation of $1/\sqrt{fan\ in}$ and a mean of zero. The last layer was drawn from a uniform distribution with a standard deviation of $3e^{-3}$ and a zero mean.

D. Learning Schedule

We trained for 100000 steps, where each episode had a duration of 1000 steps. We used momentum optimiser with stochastic gradient descent for both networks, where network weights were updated with a learning rate of 0.0001 and a momentum of 0.3 for both networks. Our minibatch size was 32. We evaluated the performance of our method with various alignment and force weights, which highly affect the convergence.

V. Results & Discussion

We conducted experiments with swarm populations of $N \in \{25, 100\}$ robots, where a multi-agent problem was formulated as a single-agent learning problem by composing sparse reward function as one dense reward function. We used a natural spring length, l_{ij} of 0.2 for each case. This means the agents should maintain a length of 0.2 when the solution has converged.

Figure 3 shows the plots generated after training for 100 episodes. It is important to note that the global force in Figure 3(b) is averaged across the swarm network. This indicates that the initial starting point of the global force for the swarm is smaller when the swarm population is small. This is indicated by the green and blue lines in Figure 3(b). Our results also showed that the force converges to the global minimum as it reaches zero over time. Comparing N=100 robots and N=25 robots cases, we can add that the smaller the swarm, the faster the convergence of the force, as blue and green lines converge faster compared to orange and purple. This could be explained by the sparsity of the reward function in a population of N=100 robots compared to N=25 robots. The greater the population size the more sparse the reward function and the harder to solve the problem.

Next, we analyse the reward function. In Figure 3(a), it is clear that both swarm populations demonstrated learning by maximising the reward function. If we look in detail, we can further see that the convergence for individual cases was much faster than that of the collective cases, regardless of the population of the swarm. Even though the collective case with N=25 robots (green line) starts above the orange line (collective case with N=100 robots), they both seem to converge around the same number of episodes. This indicates that the starting value of the reward has almost no impact on the rate of finding the solution, which is an interesting topic for further analysis.

Furthermore, we showed that our Q-value was increasing during training and approaches to zero as shown in 3(c), which further proves that the system was learning to form a collective motion. Once again, blue and purple lines, which are both individual-wise action spaces, had a much faster Q-value learning experience compared to the collective version of their respective populations.

However, neither case leads to reward greater than 0 and force equal to zero, which could be due to (1) training did not suffice long enough, (2) team learning simplification inspired by [22] in the formulation of the reward function did not suffice to address the credit assignment problem, as the reward function is very sparse in a swarm population of 100. It can be seen that the plots for the individual cases are much more oscillatory than the shared action case. This could be due to (1) each robot having its own set of action parameters, so the convergence of the global force function might take longer, (2) actor-network is much larger due to the last layer of 300 neurons, and so it takes longer to train the network.

Besides, we ran experiments with the weights given in Eq. (6). We found that the optimal values of weights were in the range $[20, 10]$ for the global force, 1 for swarm alignment, and $[1, 3]$ for the actions weights. These weights can be thought of as priorities given to certain blocks of the system in the reward function. Increasing too much of alignment weight leads to an unstable response, as the priority is given to the alignment of each individual in the swarm, neglecting

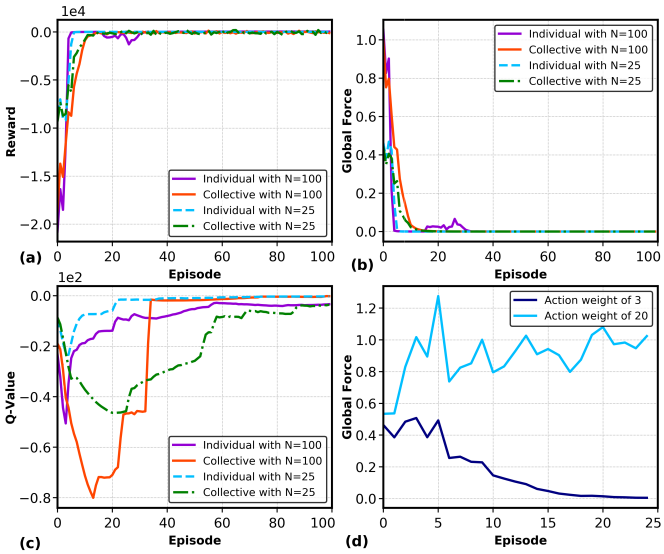

Fig. 3. Learning a centralised policy during 100 episodes of training, (a) presents the maximisation of reward during learning, (b) shows the convergence of the global force, (c) shows the progression of Q-value during learning for all cases, and (d) shows the impact of action weight on the minimisation of force.

the need to decrease the force to its minimum. Similarly, reducing alignment weight too much led to a very slow learning experience. This is shown in figure 3(d). Overall, we believe the tuned parameters should balance both cases. Lastly, we found that increasing the action weight significantly improves the rate of convergence. This could be explained by the fact that the actor-network might get stuck in local minima during learning and hence might give an output that is around 0 in centre. This would then halt the learning for some time until the critic provides non-zero gradients. However, incorporating the mean of actions in the reward design, enables us to maximise the mean of actions which in turn forces the actor to give non-zero output. This significantly improves the reduction of force function and the rate at which this happens. However, increasing this weight can cause the force to become unstable during learning.

VI. Conclusion

In this paper, we proposed two different mechanisms for achieving collective motion in a swarm of homogeneous robots. By treating the swarm as one agent and using a global reward, we could then feed in the individual-wise actions $\{k, \beta, \alpha\}$ for each robot, or have the actor-network generate one set of $\{k, \beta, \alpha\}$ which would then be shared across each agent. We formulated a reward function for our problem and showed that RL can be effectively used in the optimisation of the parameters of the CM by i) minimising the global force which means that a flock is formed and maintained and ii) maximising the rate of convergence of CM, which means that each agent is aligned while forming the flock. Furthermore, it shows that DRL can enable us to tune systems of collective motions in a way that would not be as easy and as simple using conventional optimisation algorithms.

References

[1] M. Schranz, G. A. Di Caro, T. Schmickl, W. Elmenreich, F. Arvin, A. Şekercioğlu, and M. Sende, "Swarm intelligence and cyber-physical systems: Concepts, challenges and future trends," *Swarm and Evolutionary Computation*, vol. 60, p. 100762, 2021.

[2] F. Arvin, A. E. Turgut, T. Krajník, and S. Yue, "Investigation of cue-based aggregation in static and dynamic environments with a mobile robot swarm," *Adaptive Behavior*, vol. 24, no. 2, pp. 102–118, 2016.

[3] A. E. Turgut, H. Çelikkanat, F. Gökçe, and E. Şahin, "Self-organized flocking in mobile robot swarms," *Swarm Intelligence*, vol. 2, no. 2-4, pp. 97–120, 2008.

[4] J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin, "Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, 2020.

[5] H. Oh, A. R. Shirazi, C. Sun, and Y. Jin, "Bio-inspired self-organising multi-robot pattern formation: A review," *Robotics and Autonomous Systems*, vol. 91, pp. 83–100, 2017.

[6] J. S. Jaffe, P. J. Franks, P. L. Roberts, D. Mirza, C. Schurgers, R. Kastner, and A. Boch, "A swarm of autonomous miniature underwater robot drifters for exploring submesoscale ocean dynamics," *Nature communications*, vol. 8, no. 1, pp. 1–8, 2017.

[7] J. Hu, P. Bhowmick, F. Arvin, A. Lanzon, and B. Lennox, "Cooperative control of heterogeneous connected vehicle platoons: An adaptive leader-following approach," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 977–984, 2020.

[8] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 1987, pp. 25–34.

[9] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, and O. Shochet, "Novel type of phase transition in a system of self-driven particles," *Physical review letters*, vol. 75, no. 6, p. 1226, 1995.

[10] J. Toner and Y. Tu, "Flocks, herds, and schools: A quantitative theory of flocking," *Physical review E*, vol. 58, no. 4, p. 4828, 1998.

[11] E. Ferrante, A. E. Turgut, M. Dorigo, and C. Huepe, "Collective motion dynamics of active solids and active crystals," *New Journal of Physics*, vol. 15, no. 9, p. 095011, 2013.

[12] M. Raoufi, A. E. Turgut, and F. Arvin, "Self-organized collective motion with a simulated real robot swarm," in *Annual Conference Towards Autonomous Robotic Systems*. Springer, 2019, pp. 263–274.

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[14] A. Tavakoli, F. Pardo, and P. Kormushev, "Action branching architectures for deep reinforcement learning," *arXiv preprint arXiv:1711.08946*, 2017.

[15] T. Van de Wiele, D. Warde-Farley, A. Mnih, and V. Mnih, "Q-learning in enormous action spaces via amortized approximate maximization," *arXiv preprint arXiv:2001.08116*, 2020.

[16] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," 2014.

[17] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[18] M. Hüttenrauch, A. Šošić, and G. Neumann, "Guided deep reinforcement learning for swarm systems," *arXiv preprint arXiv:1709.06011*, 2017.

[19] M. Hüttenrauch, S. Adrian, G. Neumann *et al.*, "Deep reinforcement learning for swarm systems," *Journal of Machine Learning Research*, vol. 20, no. 54, pp. 1–31, 2019.

[20] R. Pinsler, M. Maag, O. Arenz, and G. Neumann, "Inverse reinforcement learning of bird flocking behavior," in *ICRA Swarms Workshop*, 2018.

[21] G. Matheron, N. Perrin, and O. Sigaud, "The problem with ddpg: understanding failures in deterministic environments with sparse rewards," *arXiv preprint arXiv:1911.11679*, 2019.

[22] C. Versino and L. M. Gambardella, "Learning real team solutions," in *Distributed Artificial Intelligence Meets Machine Learning Learning in Multi-Agent Environments*. Springer, 1996, pp. 40–61.