Protection algorithms using fault resilient fish swarming behaviour

Sebastian Deaconu
Supervisor(s): Ashutosh Simha, Suryansh Sharma
Professor: Ranga Rao Venkatesha Prasad
EEMCS, Delft University of Technology, The Netherlands
23 June 2022

A Dissertation Submitted to EEMCS faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering

Abstract

This paper analyzes how flocking behavior in fish can be used to develop target protection algorithms. This starts from the hypothesis that fish aggregate into coordinated flocks in order to protect themselves from predatory attacks. In order to test the protection capabilities of fish, a Prey-Predator instance is developed in which faults are introduced. Prey fish try to protect their faulty flock-mates and themselves from predatory attacks while Predators hunt the fish in order to stay alive. The simulated fish are developed using Boids that emerge in a 70 prey versus 7 predators ecosystem [4]. Results are then extracted using multiple attacking and protection strategies as well as different faulty Boid configurations.

The results initially show no improvement when flocking around faulty prey but when a genetic approach is introduced, the prey gains a clear advantage against the predators. This implies that fish flocking (as opposed to individualistic behavior) is an optimal protection strategy against attacks and could be used in other instances such as military operations or agriculture automation.

Keywords: Swarm Robotics, Multi-Robot System, Boids, Flocking, Fault resilience, Flock Protection, Distributed Learning

1 Introduction

Robot swarms have the potential to tackle real-life issues because of their ability to sense and perform actions over large areas. They can be applied in house management systems, agriculture automation, environmental monitoring and search and rescue missions. However, because of the large number of robots, swarms have the downside of losing individuals due to faults which consequently could lead to unexpected and unwanted behavior. For instance, in a search and rescue scenario, faulty bots will slow down the foraging capabilities of the swarm which could lead to delayed interventions and possibly unsuccessful rescues altogether. As robots are starting to be used more regularly, the issue of fault resiliency and isolation becomes more and more visible. Because of this, solutions and algorithms have already been developed. Some fault resilience solutions include mathematical models, such as fault-tolerant path planning algorithms using an artificial potential field (APF) module, bond-graph modeling, as well as Kalman filters[2][9][10]. Other approaches are inspired from natural emerging behavior. Such solutions take inspiration from cell cross-regulation to detect faults[15]. Others use a fire-fly inspired algorithm with pulse-coupled oscillators[1]. More state of the art centralised and distributed methods (such as distributed deep learning) are proposed in [12].

This paper will use a fish flocking approach to creating a fault resilient protection algorithm. The goal of developing such an algorithm is not only fault resilience but also to protect stationary targets from incoming heat-seeking missile attacks. The study will correlate the protection capabilities of fish flocking to a single point of interest or individual. In order to test both instances of protection, a prey and predator instance inspired by fish behavior will be used. This instance will use flocking algorithms inspired by Craig Reynolds and individuals called Boids (generic simulated flocking creatures)[4]. The Boids simulate fish or bird-like flocking mechanisms. This model was chosen as it very closely mimics a fish ecosystem where distributed protection or attacking behavior can be added. These behaviors are improved further by using a genetic learning approach. The faults lead to the actual challenge of this research which implies target protection through the action of flocking around faulty prey when predators are attacking.

Some of the current state-of-the-art models are in three dimensions in order to make an in-depth study on the patterns that emerge during predator attacks [5]. But since further research shows that the dimensionality has a low impact on the results and simulations, a two-dimensional model is sufficient [6]. Many authors used the Boid model to prove and simulate natural behavior within flocking. Moškon et al, for example, used fuzzy logic to simulate the foraging behavior of artificial birds [8]. There are also Boid models that implement predator behavior, most of them attacking the flock center[7]. This is one of the functionalities that the predators in this research will use alongside attacking the prey furthest from the flock. This should create a realistic predator model based on the behavior of various fish(for example swordfish). Although there is research on prey and predator behavior, the behavior of target protection has not been tested using such an approach.

This paper is organized as follows. In Sections 2 and 3 we first define the problem as well as the methodology. Section 4 will show how the fault detection algorithm is implemented. The simulation environment as well as the results can be found in section 5. Section 6 reflects on the ethical aspects of the problem. In sections 7 and 8 conclusions and final notes are described.

2 Problem Description

The objective of this study is to implement cutting-edge swarming and platooning algorithms for collaborative operations. The swarming algorithms will be based on decentralized/distributed sensing and control (Imagine a flock of birds or a school of fish swarming together). Moreover, the questions related to this work come in the form of how can we develop protection algorithms using the flocking capabilities of fish. The aim of this research is to develop protection algorithms based on the behavior that emerges from fish flocking as a response to predatory attacks. To achieve this, a deep learning approach will be taken. The protection algorithm is tested in a custom-made simulation environment where the flocking behavior of Boids is ranked based on a Prey-Predator and a Missile-Target instance, both described in Section 4.

3 Methodology

In order to test the efficiency of the protection algorithm, an ecosystem instance was developed where faults can be injected and learning capabilities for the agents are easy to train. The agents are described as artificial life programs developed

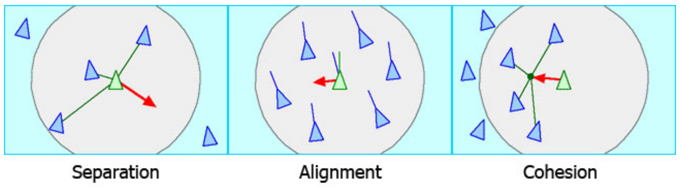

Figure 1: Boid Flocking rules

by Craig Reynolds called Boids [4]. Boids give a simple but reliable bird-like flocking behavior using three main rules (Figure 1) :

- Separation: steer to avoid crowding local flock-mates.
- Alignment: steer towards the average heading of local flock-mates.
- Cohesion: steer to move towards the average position (center of mass) of local flock-mates.

The steering behavior of the Boids is based on the work of Craig W. Reynolds who states “The physics of the simple vehicle model is based on forwarding Euler integration. At each simulation step, behaviorally determined steering forces (as limited by max_force) are applied to the vehicle’s point mass. This produces an acceleration equal to the steering force divided by the vehicle’s mass.“[17]. The acceleration is then added to the current velocity in order to reach the desired velocity force(algorithm 1). The position of the Boid is then updated with the new force and its angle(Figure 2).

Algorithm 1 Steering Algorithm

```
function UPDATE
    $position \leftarrow position + velocity$
    $velocity \leftarrow velocity + acceleration$
    $velocity.limit(max\_speed)$
    $angle \leftarrow velocity.heading + \frac{\pi}{2}$
end function
```

The Boids are then placed in a predator and prey scenario. The scenario follows a game approach where the predators try to hunt in order to survive whereas the prey runs until the attackers starve. Actuator faults are then introduced in both teams which are then mitigated by the fault-resilience algorithm. Both teams collect data in terms of lost/won rounds, time per round, and the ratio of survivors over total Boids. Furthermore, a custom-made efficiency function takes the experimental data and estimates how well fault resilience was done. The main reason for doing this game-inspired setup was to test the efficiency of fish flocking mechanisms as well as their capabilities of protection and flock resilience when it comes to weak individuals. Besides this, the game approach has a clear advantage when it comes to deep learning algorithms. This is because the two teams can train while competing with each other endlessly. This should produce a good learning model without the need for data gathering and engineering. The model will then be tested using a Missile-target instance in order to rank the performance in a more realistic scenario.

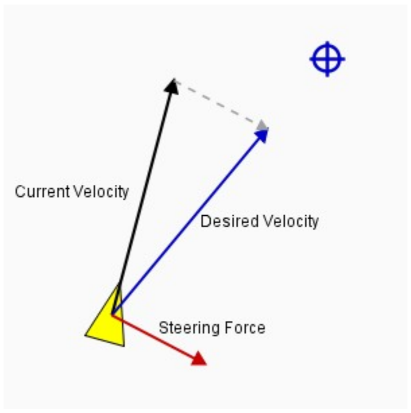

Figure 2: Boid steering behaviour

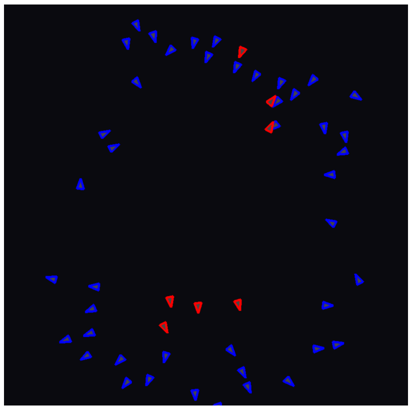

Figure 3: Instance of 70 prey and 7 predator Boids.

4 Attacking and Protection Algorithms

This section describes the behavior and interactions between Boids as well as the learning setup that maximizes the response of these behaviors. The section contains three parts. The first part describes the protection algorithms used by the prey Boids. The second part introduces the attack algorithms of the Predators. And finally, the last part shows how these algorithms are optimized using genetic learning.

4.1 Prey Boids Algorithms

The fault resilience algorithm described in this section aims to improve the flock lifespan (predators or prey) by implementing attacking and protection behavior inspired by nature. Based on the experiments done by Miller on crab aggregation and predators, it was shown that there exists collective response behavior in flocks: prey individuals move toward the center of the flock and the flock curves away from the predator’s attack [16]. With this behavior in mind, the protection algorithm enforces that prey Boids protect their faulty flock-mates by aggregating around them. This is defined by steering towards the faulty Boid, considering it the center of mass inside the flock. This can also be seen with the steering equation 1 that is enforced on Boids when they sense faulty teammates. Besides this, prey Boids try to evade predator attacks using the evasion law from equation 2. When a predator

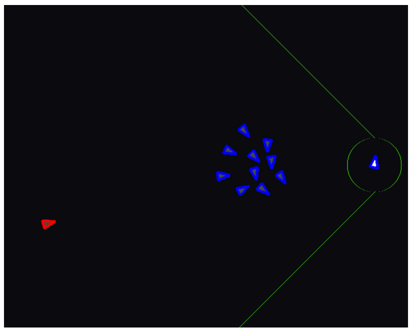
Figure 4: Target-Missile instance with 10 protective Boids

attacks, the functional Boids flee, hoping to lure it away from the faulty prey.

$$acc_{curr} = pos_{fty} - pos_{curr} \tag{1}$$

$$acc_{curr} = pos_{pred} + vel_{pred}^2 - pos_{curr}{}^1 \tag{2}$$

4.2 Predator Boids Algorithms

On the contrary, the predator attack model is based on killer whale attacks, which makes the Boids target the prey that is the farthest from the flock(the most isolated prey) [11]. This translates to hunting the prey furthest from the center of mass(equation 3) which is chosen using equation 4. Besides this, predators will try to attack the prey that is closest to them in hopes of refreshing their hunger value quicker. For all the types of selected prey, predators have the same attack law, trying to intercept the prey position(equation 5).

$$centr = \frac{1}{n}\sum_{i=0}^{n} pos_{c\_prey(i)} \tag{3}$$

$$prey = max(getDistance(list_{prey}, centr)) \tag{4}$$

$$acc_{curr} = pos_{prey} + vel_{prey}^2 - pos_{curr}{}^1 \tag{5}$$

4.3 Genetic Learning Algorithm for Weight Optimisation

The Boid moving laws as well as the fault resilience laws are influenced by weights. This makes the Boids favor certain behaviors over others. Because of this, each behavior label can be tweaked in order to reach a balance between flocking, protecting, fleeing, and attacking. A genetic algorithm is used in order to optimize these weights. The algorithm starts by including genes in the Boid definition. Based on their type, Boids can have two gene definitions:

- Prey: DNA = {"flee": random(0, 1), "protect": random(0, 1)}
- Predator: DNA = {"attack_furthest": random(0, 1), "attack_closest": random(0, 1)}

[1]The description of equation terms can be found in section A.1

The gene values are at first randomly initiated. After the gene creation, a scenario similar to the one in Section 5 is built, the difference being that each Boid now has the ability to reproduce. Having a 1% chance(or double if close to a faulty flock-mate), Boids can produce an offspring that inherits their DNA and mutates it. The mutation rate is defined using algorithm 2:

Algorithm 2 Mutation Algorithm

```
function MUTATE($DNA$)
    for $gene \leftarrow DNA$ do
        $DNA[gene] += random.uniform(-0.1, 0.1)$
        $DNA[gene] = max(min(DNA[gene], 1), 0)$
    end for
    return $DNA$
end function
```

With the mutation algorithm in mind, the ecosystem continues to run until only one team remains. This is considered as one round. After the round ends, the top 10 best performing genes are selected for the next round. The algorithm is then run for several rounds(around 50) until convergence which produces the following approximated weights:

- Prey:{'flee': 0.791, 'protect': 0.576}
- Predator:{'attack_furthest': 0.754, 'attack_closest': 0.548}

After the 50-round mark, the algorithm is over-sampled and produces under-performing results, thus in the experiments, the weights for the trained agents will be the ones above.

5 Experimental Setup and Results

The experiments are set up in a custom environment developed in python and consist of two separate instances. In the first instance, the predatory and flocking behavior of fish is observed. In the second one, this flocking behavior is applied in a more realistic scenario of anti-missile defense, where Boids are used as defensive decoys.

5.1 Prey-Predator Instance

The Prey-Predator instance features a 1920x1080 arena where 70 prey Boids and 7 predators are dropped randomly (Figure 3). The Boids then proceed to attack and protect each other. After 100 game ticks and until 600 all faults are injected randomly into the Boids. Faults can have two strategies: full stop or corrupted movement, which are also chosen randomly. Each game is won by either of the teams and it gets a score based on time, Boids left and faulty Boids left, defined by the scoring function 9. The simulation is then run with all possible configurations in terms of faulty prey and predators.

$$tScore = \frac{50 * fps - fTime}{45 * fps} \tag{6}$$

$$fScore = \frac{fBoid_{rem}}{fBoid_{tot}} \tag{7}$$

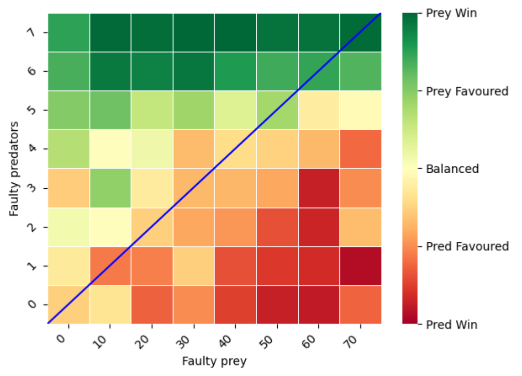

Figure 5: Instance of 70 prey and 7 predator Boids with baseline rules for Prey and improved rules for Predators. Weights: flee: 0.6, protect: 0, attack_furthest:0.2, attack_closest: 0.4

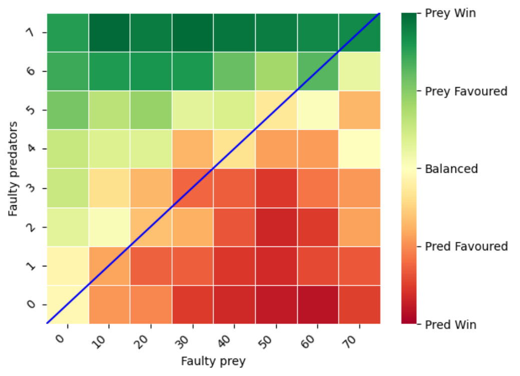

Figure 6: Instance of 70 prey and 7 predator Boids with improved rules for both parties. Weights: flee: 0.6, protect: 0.4, attack_furthest:0.4, attack_closest: 0.4

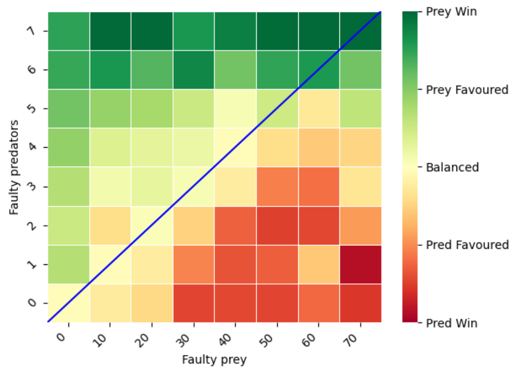

Figure 7: Instance of 70 prey and 7 predator Boids with improved rules for both parties and genetically trained Prey

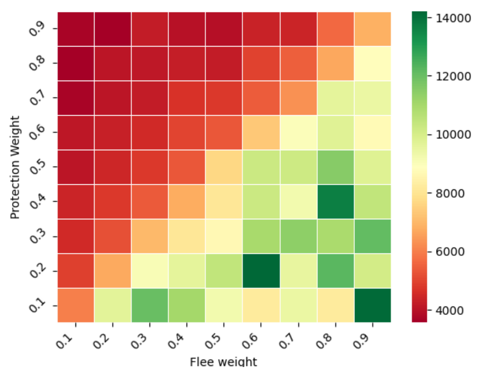

Figure 8: Target-Missile instance with different weight combinations for the protection Boids

$$aScore = \frac{boid_{rem}}{boid_{tot}} \tag{8}$$

$$score = max(min((0.5*tScore+0.5*aScore+fScore),1),0)^{1} \tag{9}$$

The results of each game are then plotted using heat maps. The color of each measurement shows which team is likely to win (red = predators; green = prey) or if the match is balanced(yellow). The configurations tested are:

- Baseline prey(Figure 5). It does not use any protection algorithm.
- Improved protection prey(Figure 6). It uses both protective and evasive behaviors with weights based on heuristics.
- Genetically enhanced prey(Figure 7). It uses both protective and evasive behaviors with weights established by the genetic algorithm.

[1]The description of equation terms can be found in section A.1

All configurations face the improved predator agent, which uses weights based on heuristics, and are run 5 times in order to reduce randomness within the data.

5.2 Target-Missile Instance

The Target-Missile instance features a smaller arena of 1600x900 pixels where 10 defensive Boids have to protect a stationary target from an incoming missile attack (Figure 4). The missile uses the same attack strategies as the predatory Boids, thus attacking the closest as well as the furthest prey from the flock's center of mass. The defensive Boids then disperse the missile away from the target using the same protection strategies as the prey Boids. This means that, in order to protect the target, the Boids flock as close as possible to each other and then disperse quickly enough such that the missile is lured away. Each game instance ends when the target is hit by the missile and is scored based on how much time the target survives.
The results are plotted in a similar fashion to the previous section, again using heat maps. The plot in Figure 8 shows

the scores for the defensive Boids using all possible weights. The scores range from 4000 to 14000 (red to green) milliseconds and are computed by running each measurement 5 times and using the mean. This is done in order to reduce variance within the data and create more realistic results.

6 Discussion and Result Interpretation

Based on the first two experiments tested in the first instance of section 5, it can be reasoned that the improved prey configuration is not very different in terms of performance from the baseline one, but after the genetic algorithm, a clear improvement can be seen. The baseline configuration(Figure 5) has a balanced ratio of wins between the prey and predators, the exception being the extremes where the multitude of faults hinder the capabilities of the teams. For instance, at the measurement of 7 faulty predators and 0 faulty prey, the prey wins by default because the predators cannot move and attack properly. The predators in the improved configuration(Figure 6) have an overall win against the prey (most squares on the blue line are yellow or red). This happens because their rules counter the prey's rules. As presented in section 4 prey Boids try to flock around faulty members in order to protect them by dispersing and luring predators away. This behavior not only protects the prey but paradoxically also attracts predators. Because their weights are not optimized they give predators enough time to find an opening. This results in the predators overrunning the protection behavior of the prey, which gives them the opportunity to disperse the flock long enough such that isolated Boids can be attacked. This attack style beats the prey defensive as faulty Boids are eaten right away and the ones that were luring get captured easily because they are isolated from other flockmates. The third configuration(Figure 7) however, behaves as expected. With optimized weights, the Boids are able to evade the predators in time, thus increasing their win rate when faults appear. This can be seen especially in Figure 7 were between the 10 and 30 Faulty prey mark where prey Boids have more wins with better results than the previous configurations. You could also argue that the prey Boids have an overall win over the predators as all results on and above the diagonal line are either green or yellow. This concludes that as the number of faults increases, the prey Boids have a better chance of survival which means that the flocking and protection behaviors enable them to survive as a flock rather than individually fending off for themselves.
The second instance should give more insight into the protection of a single individual as well as an analysis of the protection and evasion algorithms and their weights. Based on the results in Figure 8, firstly, it can be argued that there is no protection without a good evasion tactic. This can be noticed by looking at the nonperforming results which appear only when the protection weight is higher than the fleeing weight. This is expected behavior as Boids that are inclined to only flock rather than also evade when attacked will be of no use in any kind of protection. This happens because they not only fail to defend but also attract other threats by flocking close together. Secondly, it can be noticed that the best performing weights are $\{0.6, 0.2\}$, $\{0.9, 0.1\}$ and $\{0.8, 0.4\}$. Even though the protection strategies for the first two results have good performances, their protection algorithm has a very low impact in their behavior (0.2 and 0.1). It can be argued that this happens because very high fleeing weights and very low protection weights create the behavior of evading missiles by never returning back to the target for further protection. In the case of multiple attacks and low protection resources, the third set of weights is the best performer as it gives the best results when defensive Boids have the ability to return back to defend for consecutive attacks. It is also important to mention that these weights are very close to the ones extracted using the genetic algorithm, which means that the same fish flocking behavior from the previous experiments is used in order to achieve missile defense. Nevertheless, both the low protection-high evasion and the genetic fish prey strategies are good performers and a combination of the two should be used in order to achieve resilient and efficient missile protection.

7 Ethical Aspects

This section will reflect on the ethical aspects of the research as well as the reproducibility of the simulations and methods used to derive the results. In terms of ethics in swarm robotics, the most relevant disadvantage is the lack of centralized control over the swarm. “Swarms may not be predictable to the enemy, but neither are they exactly controllable or predictable for the side using them, which can lead to unexpected results: (...) a swarm takes action on its own (...)“[14]. This puts a big responsibility on the distributed AI algorithm controlling the swarm robots as it has to mitigate unexpected actions. A solution to this is mentioned in [18], where the swarm chooses to compromise its integrity in order to mitigate unwanted behavior: “A robot can sometimes choose actions that compromise its own safety in order to prevent a second robot from coming to harm. An implementation with both e-puck and NAO mobile robots provides a proof of principle by showing that a robot can, in real-time, model and act upon the consequences of both its own and another robot‘s actions.“ Regarding reproducibility, the simulation environment, as well as the tools for result extraction, will be shared on GitHub. Alongside these, instructions on how to derive the results and remarks for future development will be added.

8 Conclusions and Future Work

The experimental study shows that the action of flocking is indeed a protection mechanism against predation. Not only does this apply to healthy flock individuals but also faulty ones. This results in a good protection mechanism where similar escape patterns to those that emerge in nature can be used [13]. This can help by luring enemies away from vulnerable targets. For example, heat-seeking missile defense.
As mentioned by Demsar and Bajec, the results show that flocking is a paradoxical action because even though it provides the protection it invites predatory attacks(in nature at least)[3]. This suggests that even if flocking can benefit the individuals as a whole, it does not provide absolute protection. However, this does not influence real-life implementa-

tions where the algorithm would be used to protect points of interest rather than escaping predatory attacks. This sifts the interest of protection from the whole flock to a single individual, which is more feasible.
Regarding the continuation of this study, a first step would be to improve the hunting capabilities of the predators as well as the faulty prey protection. Furthermore, an extrapolation of the simulation to a three-dimensional space should be added. This, combined with the advanced hunting and evasion tactics should give a more realistic scenario as well as a better understanding of the Boid movements. The Missile-Target scenario could be extended to multiple missile attacks and configurations. This should make a clearer classification of the protection strategies.

A Appendix

A.1 Equation terms description

Eq no.	Term	Description
1	acc_{curr}	current boid acceleration
	pos_{fty}	faulty boid position
	pos_{curr}	current boid position
2	acc_{curr}	current boid acceleration
	pos_{pred}	predator position
	vel_{pred}	predator velocity
	pos_{curr}	current boid position
3	centr	center of mass
	pos_{c_prey}	nearby prey position
4	prey	selected prey to hunt
	getDistance	get the distance from two points
	list_{prey}	list containing nearby prey
	centr	center of mass
5	acc_{curr}	current boid acceleration
	pos_{prey}	prey boid position
	vel_{prey}	prey boid velocity
	pos_{curr}	current boid position
6	tScore	time score
	fps	frames per second
	fTime	finishing time
7	fScore	fault score
	fBoid_{rem}	remaining faulty boids
	fBoid_{tot}	total faulty boids
8	aScore	alive score
	boid_{rem}	remaining boids
	boid_{tot}	total boids
9	score	total scoring
	tScore	time score
	aScore	alive score
	fScore	fault score

References

[1] A.L. Christensen, R. O'Grady, and M. Dorigo. From fireflies to fault-tolerant swarms of robots. *IEEE Transactions on Evolutionary Computation*, 13(4):754–766, 2009.

[2] Matthew J. Daigle, Xenofon D. Koutsoukos, and Gautam Biswas. Distributed diagnosis in formations of mobile robots. *IEEE Transactions on Robotics*, 23(2):353–369, 2007.

[3] Jure Demšar and Iztok Lebar Bajec. Simulated predator attacks on flocks: A comparison of tactics. *Artificial Life*, 20(3):343–359, 2014.

[4] Craig W. Reynolds Symbolics Graphics Division, Craig W. Reynolds, Symbolics Graphics Division, and Other MetricsView Article Metrics. Flocks, herds and schools: A distributed behavioral model: Proceedings of the 14th annual conference on computer graphics and interactive techniques, Aug 1987.

[5] Charlotte K. Hemelrijk and Hanno Hildenbrandt. Some causes of the variable shape of flocks of birds. *PLoS ONE*, 6(8), 2011.

[6] Andreas Huth and Christian Wissel. The simulation of the movement of fish schools. *Journal of Theoretical Biology*, 156(3):365–385, 1992.

[7] S.-H. Lee, H.K. Pak, and T.-S. Chon. Dynamics of prey-flock escaping behavior in response to predator's attack. *Journal of Theoretical Biology*, 240(2):250–259, 2006.

[8] Heppner F. H. Mraz M. Zimic N. Lebar Bajec I. Moškon, M. Fuzzy model of bird flock foraging behavior. *Proceedings of the 6th EUROSIM Congress on Modelling and Simulation*, 2:1–6, 2007.

[9] Arthur G.O. Mutambara. Decentralized estimation for multisensor systems. *Decentralized Estimation and Control for Multisensor Systems*, page 55–80, 2019.

[10] Ranjith Ravindranathan Nair, Hamad Karki, Amit Shukla, Laxmidhar Behera, and Mo Jamshidi. Fault-tolerant formation control of nonholonomic robots using fast adaptive gain nonsingular terminal sliding mode control. *IEEE Systems Journal*, 13(1):1006–1017, 2019.

[11] Leif Nøttestad and Bjørn Erik Axelsen. Herring schooling manoeuvres in response to killer whale attacks. *Canadian Journal of Zoology*, 77(10):1540–1546, 1999.

[12] Oyinlola Oladiran. Fault recovery in swarm robotics systems using learning algorithms.

[13] T. Pitcher and C. J. Wyche. Predator-avoidance behaviours of sand-eel schools: Why schools seldom split: Semantic scholar, Jan 1983.

[14] P. W Singer. Military robots and the future of war. *PsycEXTRA Dataset*, 2009.

[15] Danesh Tarapore, Anders Lyhne Christensen, and Jon Timmis. Generic, scalable and decentralized fault detection for robot swarms. *PLoS One*, 12(8):e0182058, August 2017.

[16] Steven V. Viscido, Matthew Miller, and David S. Wethey. The response of a selfish herd to an attack from outside the group perimeter, May 2002.

[17] Craig W. Reynolds. Steering behaviors for autonomous characters.

[18] Alan Winfield. From robot swarms to ethical robots: The challenges of verification and validation.