# Investigating the Impact of a Genetic Algorithm on Inter-Flock Dynamics in a Competitive Environment

Matthew James Hunter

BSc(Hons) Computer Games Technology

2019

School of Design and Informatics
University of Abertay Dundee

# Contents

## List of Figures

## Acknowledgements

To Hayley McCullough, who has somehow tolerated me during this.

I would like to thank Dr.Karen Meyer for her support as a supervisor.

Thankyou to Carol Clark, who has been a great mentor and has given me increased confidence as the year has gone past.

# Abstract

**Context** Flocking algorithms represent a kind of group behaviour that collectively improves the survival chances of individual members and using genetic algorithms to improve a flock is an area already under exploration. However, a research gap is whether a genetic algorithm can improve a flock's competitiveness versus a regular flocking algorithm and result in behaviour changes.

**Aim** The aim is to develop an environment in which a flock that improves over generations, via a genetic algorithm, can compete for resources with a regularly implemented flock, using the expanded boids model. This research will investigate how the genetic algorithm effects the competitiveness of the flock in the environment in survival, potential convergence in evolutionary traits, and emergent situational behaviours.

**Method** A competitive environment with a limited resource pool, insufficient to sustain two flock populations, was developed. For this environment, a flocking algorithm and genetic algorithm were developed. Tests were then carried out on the effectiveness of different selection methods, how the mutation rate affected fitness, and the behaviours the genetic algorithm produced in the boids.

**Results** The quantitative analysis of fitness values displayed no discernable increases or decreases in fitness outside of anomalies. The qualitative analysis displayed some strategic improvements over the regular flocking algorithm, such as having a weaker 'other flock avoidance' force to directly outcompete a regular boid over a resource.

**Conclusion** This project has overall, through the development of a testing environment and collection of data, identified the significance of the qualitative and quantitative results. The improved flock produced strategies to outcompete the other flock, but also failed to quantitatively improve its fitness over generations in any of the simulations.

# 1 Introduction

## 1.1 General Introduction and Background Information

Flocking is an emergent property of social behaviours; it is the common movement of individuals guided by both social and environmental influences and is one of many ways social organisms engage. These flocks of organisms can be found interacting with other flocks in nature, and the way they interact is as varied as it is interesting.

Producing flocking behaviour that recreates those found in nature is an endeavour already undertaken and in constant update. The field, taking off in 1987 with Craig Reynolds' influential paper (Reynolds, 1987), shows how realistic flocking behaviour can already be achieved by applying 3 simple rules to each boid ('boid' as coined in the mentionned paper: this is essentially an agent) in regard to its neighbours:

- Cohesion - movement toward the average position;
- Alignment - movement toward the average direction;
- and Separation - movement to avoid collision.

Since then, the original flocking algorithm has been extended in various ways, with communication techniques, mathematical models for how leadership arises in the flock, as well as models for how consensus is made in a flock. These expansions allow for more complex behaviours and reactions to the environment and surroundings.

Learning behaviours have also been developed. The behaviour and strategies produced by flocks form patterns. This means that if a flock can learn and understand those patterns it has a significant advantage over the other flock – an interesting example would be a group of honey hunters and smoke. Bees flee the nest if they think there is a fire; the first warning sign of this is smoke; as a group (or flock) they take advantage of this by releasing smoke into the hive. The bees evacuate the nest; the hunters get honey.

This dissemination of new knowledge, either through behaviour or communication is interesting because it can increase the complexity of the reactions the flock has to a given situation. This added complexity may lead to new behaviours that may not have been easily predicted or thought of as something a flock could produce.

Imagine the kinds of complex behaviours that might arise out of interactions of different types of flock with these added behaviours in an RTS (Real-Time Strategy) game. Not only would you have to focus on resource management, but on the counter play that each of the flocks would come up with against each other. Maybe a flock has a certain area advantage so moves to that area nearby, or is less in numbers in comparison so has to adapt its strategy to deal with that (possibly by making it seem like its more numerous than it is, forcing that other flock to adapt). This is purely speculative, however much like science fiction is to science, it's sometimes good to think about what might be possible first, and then think of a path to get there.

## 1.2 Aim, Objectives and Research Question

**Aim** To investigate if a flock improved by a genetic algorithm can outcompete a regular flock in a competitive environment, and to identify what changes the genetic algorithm produces, strategies, and the kinds of behaviours produced over time.

**Objectives** The aim and research question of this work will be met through the following objectives:

- To research and evaluate the effectiveness of a genetic algorithm on improving itself in a complex environment, studying the effects produced on flocking behaviour.
- To produce an application that models flocking behaviour and allows the observation of the genetic algorithms flocking strategies against a regular flocking algorithm.
- To analyse the effectiveness of strategies that the genetic algorithm comes up with in its interactions with the regular flocks, its ability to improve against the fitness function, and an evaluation of diiferent methods within the genetic algorithm.

**Research Question** What impact can a genetic algorithm have on the dynamic interaction of flocks with each other, and can it outcompete a regular flocking algorithm?

# 2 Literature Review

This chapter will identify the gap in research that forms the basis for the research question, by reviewing existing work on flocking and genetic algorithms, and focusing down from the wider fields of research it will frame the context of this research paper.

## 2.1 Flocking in Nature - Swarm Behaviour

Flocking algorithms draw a lot of their inspiration from the behaviour of flocks in the natural world (Flake, 1998). As there are many examples of this behaviour in organisms across the planet, there is a lot of information and insight that can be gleaned on how to design these algorithms. Flocks take on a kind of swarm behaviour and looking at the broader set of behaviours can point to the interesting ways a flock could behave under certain conditions.

**Decision Making** The way decisions are made as a flock emerges is varied. The two main ways are via consensus and leadership. An interesting look at this can be found in the behaviour of pigeons. A study conducted into the behaviour of these birds in a flock (Jorge and Marques, 2012) found that leadership initially emerged from younger pigeons in the flock, but as the flight went on older members of the flock led the group; the paper then goes on to discuss how social versus personal information affects the behaviour of the flock. What this demonstrates is how the extra experience of older members of the flock is taken advantage of in determining leadership and therefore which actions are taken by the flock; in this way they build consensus on their leadership through the choices individual flock members make in their group. This is further confirmed in *'Misinformed leaders lose influence over pigeon flocks'* Watts et al. (2016). This work highlights the interaction between consensus and leadership in making decisions and demonstrates that they can be simultaneously present in the process. Consensus building has also been modelled, in this case in swarms (Shang and Bouffanais, 2014). They consider the interaction between individuals and how that ripples out to form a consensus within the flock. In the paper they look at a hybridisation of two types of models that cover this behaviour, one based on metric distance, and the other on topological distance; both of these models have a lot of research behind them. This shows how communication can work in a flock to produce decisions, and if expanded could be used to disseminate more complex information throughout the group.

**Eusocial Behaviour** While not specifically flocking as it pertains to biology, this falls under the bracket of swarm behaviour and has interesting relationships which can inform us on how to expand flocking algorithms in beneficial ways. For example, communication can occur in a variety of ways; ants use pheromones to find shortest paths (Dorigo and Gambardella, 1997), and bees use a waggle dance to inform others in the hive of food sources and potential new nest sites

(Al Toufailia et al., 2013). This eusocial behaviour could incite the ideas that lead to an expansion in flocking and genetic algorithm capabilities.

## 2.2 Flocking Algorithm

Flocking algorithms draw inspiration from the natural world. However, the design of these algorithms involves mathematical approximations of the behaviours involved, the interactions which take place and the systems overall.

**Reynolds Boids** In the often cited paper which informs many key discussions of flocks: *'Flocks, herds and schools: A distributed behavioral model'* Reynolds (1987), we see that we have the three main forces that make up a basic flock: Separation, Cohesion and Alignment. These are approximative forces that represent the aggregate motion of a collection of boids (representing flock members, which in turn can represent different species in nature).

**Expanded Boids** There are many variations of the expanded boids model dependent upon what is necessary for the research conducted. The expanded model presented by Kawabayashi and Chen (2008) is very useful as their implementation is very clear and displays useful expansions to the model for this research. It adds functionality for running away from predators, attraction to food sources, relations as towards other neutral flocks, the ability to search its surroundings, boundary conditions, and obstacle avoidance. All of which are useful in designing more natural flocking behaviour.

**Decision Making** In parallel to the way decisions are made in natural flocks, flocking algorithms can also make decisions. An interesting way of producing leadership in an artificial flock can be found in the paper *'Autonomous Boids'* Hartman and Benes (2006). Here they propose the use of an 'Eccentricity' variable to determine leadership in the group in order to make decisions based on the boids' proximity to the front of the flock: the closer it is, the higher a chance of leading the flock. This mimics the way that some species of bird, such as starlings, decide on leadership within the group. This leadership can then be used to influence decisions by the rest of the flock.

## 2.3 The Interaction of Flocking Algorithms

In the paper *'Simulating Species Interactions and Complex Emergence in Multiple Flocks of Boids with GPUs'* Husselmann and Hawick (2011), multiple flocks, each a different type of boid, are run in a closed environment to see what aspects of species interaction could be reproduced. This paper is important as it covers how the authors created their different types of flocks and what sort of parameters they used when differing them. They also offer an interesting way of getting results; "histogramming in 3D-space" as the authors put it, describing it by writing: "the 3D space is discretised and then each boid's position is mapped to a discrete cube in 3D space". This is used in order to view the patterns of behaviour more clearly, and may come of use in discussing and testing the patterns produced by flocks in the application to be built. The main limitation of the paper is the simplicity of the boids in question, and while it's noted in the paper that they wish to address this, further research into the authors' papers since show no further research into the interactions of multiple flocks.

## 2.4 Genetic Algorithm

The literature on genetic algorithms is also relevant as this is what will be used to expand upon the capabilities of the flock.

### 2.4.1 The Potential Effect of AI on Flocking Algorithms

A paper discussing the use of reinforcement learning in a multi-agent system by Jaderberg et al. (2018) displays the potential of AI applied to a group of agents. It displays the qualitative findings of the research, demonstrating how the agents produced intelligent behaviour, such as camping and other player strategies. This shows that it is at least feasible for agents, or boids to produce strategic behaviour in a fixed environment.

### 2.4.2 The effect of Genetic Algorithms on Boids

The general use of Genetic Algorithms in application for a boids model has been researched, and useful insight is available on how to go about adapting this technique. For example in *'Genetic Algorithms for Optimization of Boids Model'* Chen et al. (2006) the authors describe how they managed to produce an improved boids model using a genetic algorithm that focused on producing more realistic behaviours in the school of fish they were simulating. Their success points toward the potential genetic algorithms have in solving problems in a complex environment, setting the research in a wider, multi-disciplinary context.

## 2.5 The Interaction of an AI Flock with another Flock

What should be noted is, the interaction of an artificially intelligent flock with other flocks is where the research becomes thinner and where the author feels based upon their own background research there is a room for expansion. This paper therefore will seek to investigate quantitatively and qualitatively a new specific use of the Genetic Algorithm, to improve a flock's capabilities in an environment of limited resources in competition with a regular flocking algorithm. This research gap has been identified by evaluating the existing work as detailed above.

## 3 Methodology

An application was produced to test whether a flock improved by a genetic algorithm can outcompete a flock using a regular expanded boids algorithm in a competitive environment. This application used the Games Education Framework (GEF) as the framework, which was chosen based on its lightweight overhead, its portability and recommendations from several university lecturers. It was developed using Visual Studio 2017 in C++, chosen due to the ease of working with them and the flexibility afforded by both tools.

GitHub Desktop was used as the tool for source control, ensuring risks to the project were kept to a minimum. Further to this, a GitFlow tool was developed in order to stem the risk of pushing and pulling work from incorrect branches, and to prevent any conflicts when committing and pushing new work. The tool essentially acted as a smart checklist; starting a programming session with meant everything was set up correctly before opening individual programs, which happened automatically as the correct items were ticked off.

### 3.1 Test Environment Design

The application simulated an environment in which two flocks interact (one of which is improved via a genetic algorithm) within a limited space of scarce resources, with the aim of finding out if a flocking algorithm improved by a genetic algorithm can out-compete one without.

To answer the research question required removing as many base assumptions as possible. To do this, three core decisions were made: the environment would be limited so as to keep the flocks in competition with each other, ensuring the flocks would interact; there would be a limited amount of resources in the environment: less than that required to support two full populations of flocks - this would drive the flocks to compete over resources spawned in the environment; finally, there would be only two flocks: one improved by a genetic algorithm over simulated generations and one produced through a tested expanded flocking algorithm design. This ensured a competitive environment in which the effects of decisions made by genetic algorithm could be singled out, and the data producing more accurate results. Fig.1 displays a screenshot of this environment in action: the red entities representing the regular flock; the blue entities representing the genetic algorithm modified flock; the green entities representing resources in the scene and the white area displaying the space in which the boids can move and interact.

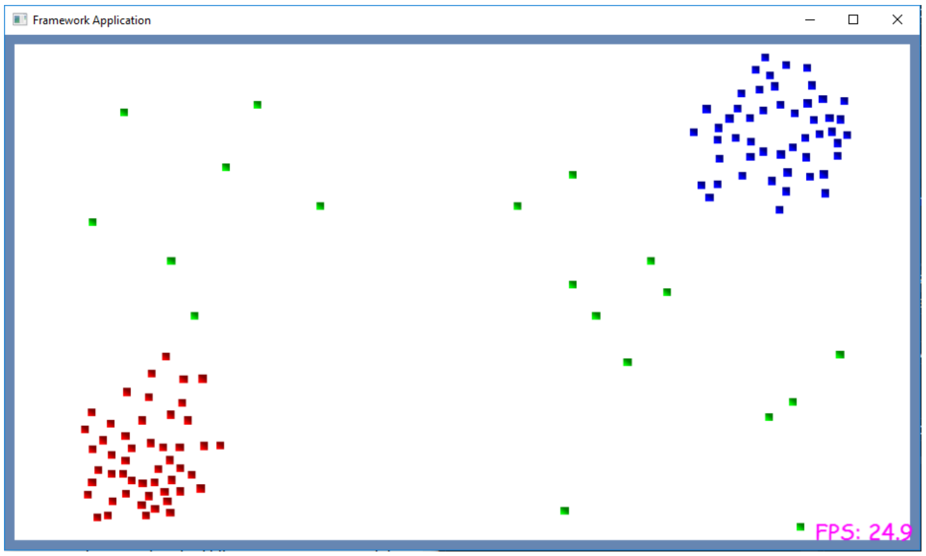

Figure 1: A screenshot of the test environment in action

## 3.2 Flocking Algorithm Design

The original flocking algorithm was designed primarily by adapting the example from Shiffman (n.d.). This provided a very clear representation of how to design a basic flocking algorithm from a code perspective. There were also clear areas for improving the efficiency of the algorithm presented. The most effective way was in combining the calculations for Alignment, Separation and Cohesion. This was to ensure each boid was not checking against all others for each force calculation, and only needed to check against all other boids once instead of three times; doing this was simple and reduced the requirement for N-Body problem solutions like the Barnes-Hut algorithm later on. The other main method lay in combining the calculations required for each force as they used shared variables, so calculating them once instead of multiple times for each force also increased efficiency.

This worked adequately for the original three forces based on Reynolds (1987). However, issues came up when expanding the application to include extra forces. These were heightened by the fact that, in the aforementioned implementation, it wasn't strictly clear how the forces would graph out due to the lack of separation between direction and weighting. This caused issues to accumulate and it was clear a change in the flocking algorithm design was necessary to expand and edit the program appropriately.

The boids' forces as they are in the final program were modelled primarily off Kawabayashi and Chen (2008), which has a very clear mathematical representation of each force. The version of the expanded boids algorithm found in this paper was modified by optimising the algorithm for the environment in which it was placed. The forces each boid experienced were: Cohesion, Alignment, Separation, Food Attraction and Flock Avoidance. Each of these forces were the result of multiplication of a unit vector and their respective weight, mathematically represented in Eq.1

$$\boldsymbol{ForceVector} = \boldsymbol{v} \cdot \boldsymbol{w} \tag{1}$$

Where $\boldsymbol{v}$ is the unit vector which describes the direction of the force, and $\boldsymbol{w}$ is the weight which magnifies the force dependent on its relation to its environment.

The relationship of the weight to the unit vector will be detailed in relation to boid forces below. Separating direction vector and weight meant it was far easier to mathematically model and graph out each force to predict their behaviour; this reduced the amount of work necessary to produce a functioning flocking algorithm. The resulting accelerative force then was the sum of these forces. That sum was then used in the Semi-Implicit Euler method to produce the updated position of each boid, every frame.

### 3.2.1 Cohesion

The first of the boid forces, Cohesion pulls the boid in the direction of flock members - in this case towards its local flock centre. The boids' local flock centre was determined by its neighbours in communicable range and is the average position of those flock members. The further the distance away from its neighbours, the stronger the force. Below is Eq.2 describing the direction vector and weighting:

$$\begin{aligned} \boldsymbol{\hat{v}_{coh}} &= \frac{LFCVector}{|LFCVector|} \\ \boldsymbol{w_{coh}} &= \frac{(|LFCPos - BoidPos|)^2}{30 \cdot FlockSize} \end{aligned} \tag{2}$$

Where $LFCVector$ represents the vector from the boid to the local flock centre, $LFCPos$ is the position of the local flock centre, $BoidPos$ is the position of the boid and $FlockSize$ is the number of flock members as a whole.

This can be seen visually represented in Fig.2. This displays the cohesion force increasing as a boid moved away from the local flock centre.

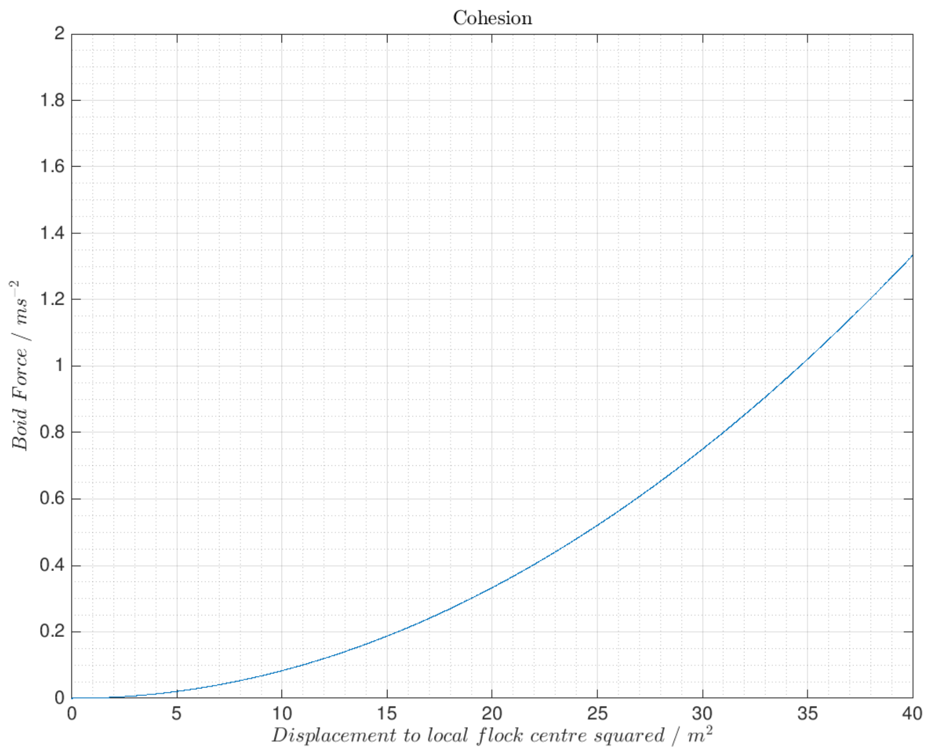

Figure 2: A Graph Displaying the Cohesion Boid Force

### 3.2.2 Alignment

Alignment is a vector which accelerates the boid in the average direction of its neighbouring boids. This produces the common direction of the flock as the boids individually move around. By taking the average velocity of neighbouring boids as a direction vector and multiplying it one over the distance between the local flock centre and the boid itself, a suitable alignment vector was produced. Below is Eq.3 describing the two components of the alignment vector:

$$\begin{aligned} \hat{\boldsymbol{v}}_{\boldsymbol{ali}} &= \frac{LFVelVector}{|LFVelVector|} \\ \boldsymbol{w}_{\boldsymbol{ali}} &= \frac{1}{10 \cdot |(LFCPos - BoidPos)|} \end{aligned} \tag{3}$$

Where $LFVelVector$ is the average velocity of the local flock members, and $LFCPos$ and $BoidPos$ are the same as in Eq.2.

This can be seen visually represented in Fig.3. It displays the alignment force increasing as a boid moved closer to the local flock centre.

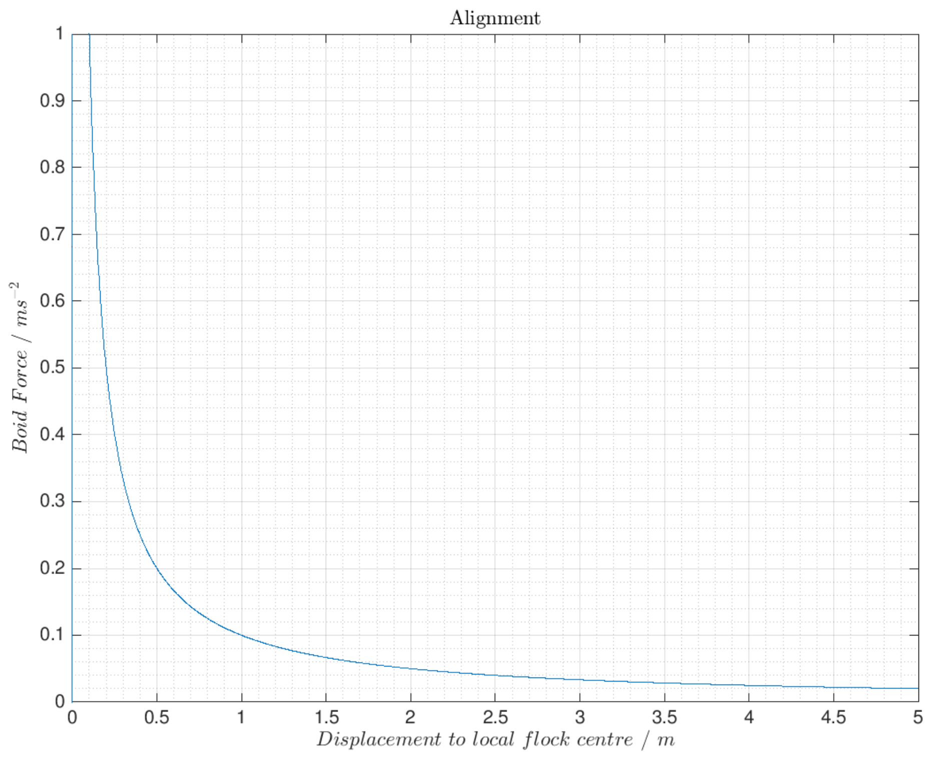

Figure 3: A Graph Displaying the Alignment Boid Force

### 3.2.3 Separation

Separation is the force that stops the boids from getting too close together. By maintaining enough space between flock members, it grants manoeuvrability where there would otherwise be collisions (which slow movement down); these could be from obstacles, other moving entities or other boids in the flock. The unit vector that represents the direction for the force to be applied was calculated by taking the negative of the vector to the nearest neighbouring boid; multiplying that by the weighted multiple of the number of neighbours divided by the distance to the closest neighbour produces the separation vector described below in Eq.4:

$$\begin{aligned}\hat{\boldsymbol{v}}_{\boldsymbol{sep}} &= -\frac{ClosestNeighbourVector}{|ClosestNeighbourVector|} \\ \boldsymbol{w}_{\boldsymbol{sep}} &= 0.025 \cdot \left(\frac{NeighbourCount}{|ClosestNeighbourVector|}\right)^2\end{aligned} \tag{4}$$

Where $ClosestNeighbourVector$ is the vector from the boid to the closest neighbour, and $NeighbourCount$ is the number of neighbouring boids.

This can be seen visually represented in Fig.4. It displays the separation force increasing as the displacement between the nearest flock member and itself decreased. This separation force increased for each neighbouring boid.

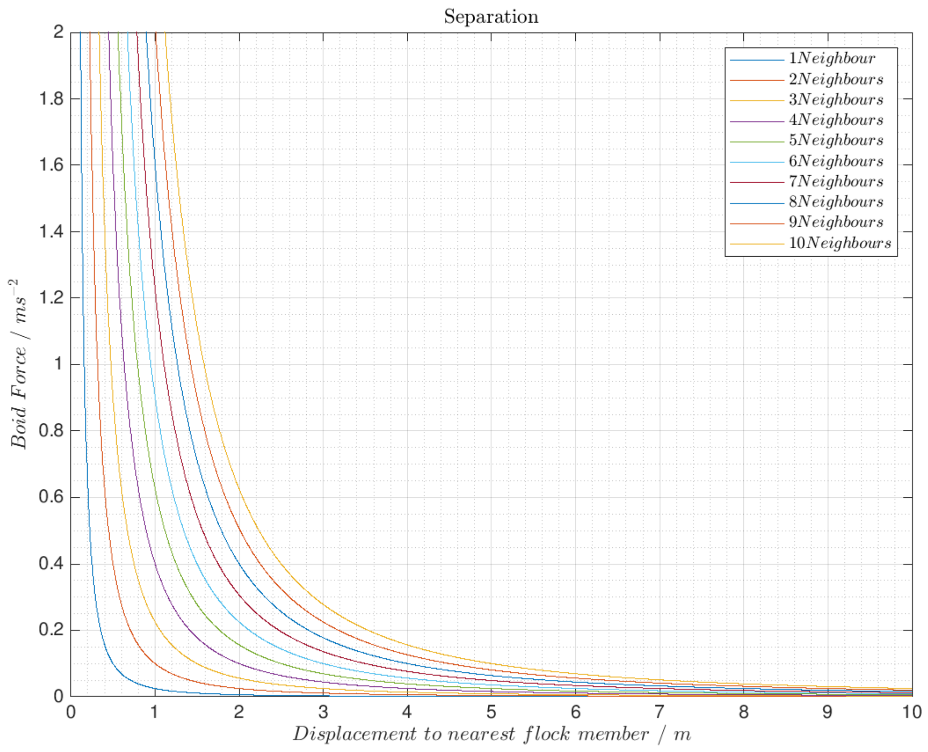

Figure 4: A Graph Displaying the Separation Boid Force

### 3.2.4 Food Attraction

This provides an accelerative force toward the nearest resource in the environment. This attraction to nearby food sources is the basis for the boids' survival in the scene. The unit vector to the closest resource gave the direction, and it is the weight which gave its relationship to food in terms of distance away. The equations used are described in Eq.5:

$$\begin{aligned} \hat{\boldsymbol{v}}_{\boldsymbol{fda}} &= \frac{ClosestResourceVector}{|ClosestResourceVector|} \\ \boldsymbol{w}_{\boldsymbol{fda}} &= 0.0025 \cdot |ClosestResource|^2 + \frac{36}{|ClosestResource|^2} \end{aligned} \tag{5}$$

Where $ClosestResourceVector$ is the vector pointing from the void to the closest resource to it, and $|ClosestResource|$ is the distance to the closest resource.

This can be seen visually represented in Fig.5. It displays the relationship between the food attraction force and the displacement squared of the food resource to the boid. It represents two increases in the food attraction force and a minima in between, where the food attraction force decreased as the boid came nearer, but then switched to an increase past a certain point.

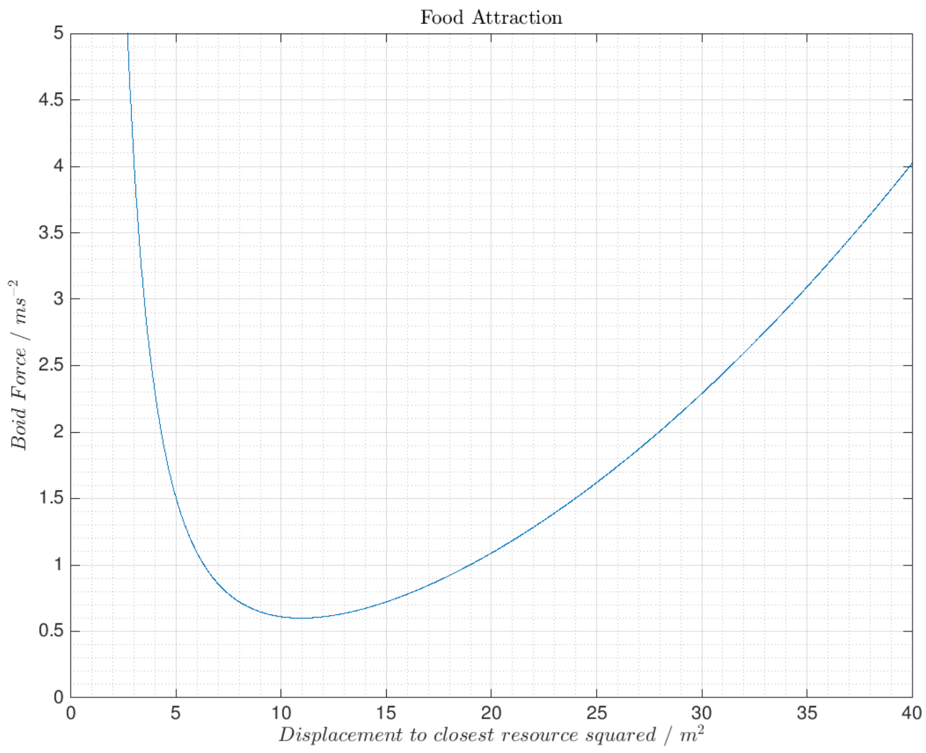

Figure 5: A Graph Displaying the Food Attraction Boid Force

### 3.2.5 Flock Avoidance

This is a vector that produces a repelling force away from another flock and increases as the boid gets closer to the other flock; this is what stoped the boids from ignoring each other and kept them in a competitive environment as they were both competing for those same resources. The mathematical relationship of the two factors produced can be seen in Eq.6:

$$\begin{aligned} \hat{\boldsymbol{v}}_{\boldsymbol{fla}} &= -\frac{OtherFlockVector}{|OtherFlockVector|} \\ \boldsymbol{w}_{\boldsymbol{fla}} &= \frac{300}{|AvgOtherFlockPos|} \end{aligned} \tag{6}$$

Where $OtherFlockVector$ is the vector pointing from the boid to the average position of another flock within interactable range, and $AvgOtherFlockPos$ is the vector to the average position of another flock within interactable range.

This can be seen visually represented in Fig.6. It displays the flock avoidance force increasing as a boid moved closer to the local flock centre.

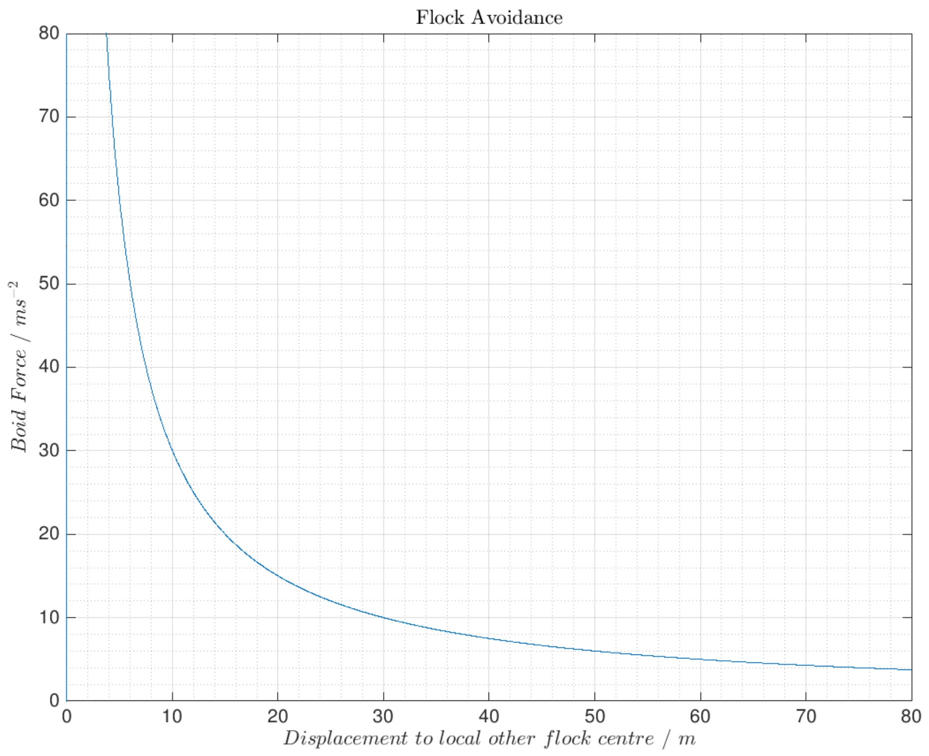

Figure 6: A Graph Displaying the Flock Avoidance Boid Force

## 3.3 Genetic Algorithm Design

The functionality for the genetic algorithm (GA) was spread over two classes - genetic_algorithm and DNA - where the genetic_algorithm class interacted with the flock and boid classes, and DNA interacted with the genetic_algorithm and boid classes. The GA had a standard approach to its design which can be seen in Fig.7.

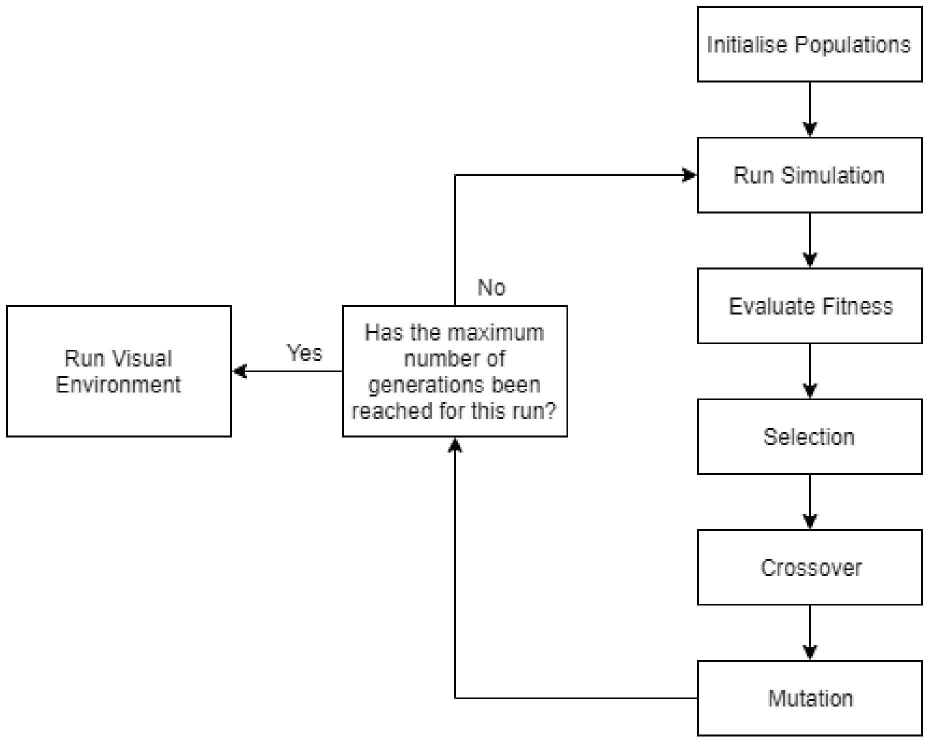

Figure 7: Flowchart displaying the process the GA takes to improve over time

Five populations were generated each time the algorithm was initialised; it could be set up to be either random or heuristic, where each population had a random set of data generated for its alleles, and the same but where one population received the heuristic used in the regular flocking algorithm respectively. These populations of boids then made up the genetically enabled flock.

Each population produced contained chromosomes unique to that population, which is represented by an array in the DNA class. Each GA boid had its own DNA object which was modified by the genetic_algorithm class. Collectively, they made up their respective genotype. Each gene in a chromosome is used in the weight calculations of its respective boid; this can be seen represented below in Eq.7:

$$
\begin{aligned}
\boldsymbol{w_{coh}} &= \frac{(chromo[0] \cdot |LFCPos - BoidPos|)^2}{chromo[1] \cdot FlockSize} \\
\boldsymbol{w_{ali}} &= \frac{chromo[2]}{chromo[3] \cdot |(LFCPos - BoidPos)|} \\
\boldsymbol{w_{sep}} &= chromo[4] \cdot \Big(\frac{chromo[5] \cdot NeighbourCount}{chromo[6] \cdot |ClosestNeighbourVector|}\Big)^2 \\
\boldsymbol{w_{fda}} &= chromo[7] \cdot |ClosestResource|^2 + \frac{chromo[8]}{chromo[9] \cdot |ClosestResource|^2} \\
\boldsymbol{w_{fla}} &= \frac{chromo[10]}{chromo[11] \cdot |AvgOtherFlockPos|}
\end{aligned} \tag{7}
$$

Where $chromo[x]$ represents each gene in the chromosome for that boid, changes of the allele of each gene affect the weights in the scene. These weight calculations made up the phenotype representation in the environment space and determined the behaviour of a boid.

### 3.3.1 GA Methods

**Evaluation** At the end of each simulation, each boid was evaluated against the fitness function, which can be seen in Eq.8. The evaluation function then sorted the boids into population types for the selection process.

$$FitnessFunction = BoidHealth + \boldsymbol{w_1} \cdot FlockHealth - \boldsymbol{w_2} \cdot OtherFlockHealth \tag{8}$$

Where $BoidHealth$ is the health of the boid, $FlockHealth$ is the combined health values of all boids in the flock, and $OtherFlockHealth$ is the combined health of all boids in the competing flock.

This fitness function in Eq.8 was designed to take into account the three important factors to a boid in a competitive flock: accounting for its own health was to represent a survival instinct; adding the total flock health to the equation meant improvements to itself over time should be in benefit the flock as a whole, and taking away the weighted total health of the other flock meant that the boid had a focus on improvement by decreasing the health of the other flock. This fitness function was intended to produce behaviour in the boids that outcompeted the regular flocking algorithm in the environment.

**Selection** The application contained two different kinds of selection algorithm: Roulette Wheel Selection (RWS) and Stochastic Universal Sampling (SUS), both used in answering the research question. In both selection algorithms, increased probabilities of being chosen were given to those populations with better overall fitness. These methods were chosen as they provided selection pressure toward fitter individuals, and also avoided prematurely coming to a solution which may have been more likely to be a local maxima.

**Crossover** Crossover of genetic data between parents was done as a single-point crossover that was randomly selected along the genotype; once selected the genotype was halved, and then swapped. This was selected due to it being one of the more efficient types of crossover involving the least calculations.

**Mutation** Mutation occurred via random increases or decreases of an allele in a randomly selected gene in the genotype. This was intended to be slow enough so as to not skip over solutions, and fast enough to come to solutions quicker.

### 3.3.2 Generations

Part of running the GA was simulating the environment in sped-up runs for the genetic algorithm to improve itself over successive generations; this was done by cutting off unnecessary calculations. That meant all generations occurred in the update function in a loop until the maximum number of generations had been reached. This cut out the requirements for rendering or indeed any calculations not related to the simulation of boid movement and running the genetic algorithm.

The calculations for the simulation physics were also cut down to speed up simulation time in each generational loop. By increasing the time per physics calculation, the time passed in each simulation was effectively reduced. For example: If one second of simulation time happens in one frame, then if the application runs at sixty frames per second, sixty second of simulation time per real time second can be achieved; this is displayed in Eq.9.

$$\boldsymbol{SimTime}(s^{-1}) = SimTimeStep \cdot FramesPerSecond \tag{9}$$

Where $SimTime$ is the amount of time passed in simulation per actual second, and $SimTimeStep$ is the difference in simulation time for each calculation.

Based on this sped-up time, each generation had a simulation time limit before moving onto the next one. This had to be balanced, as it required enough time for real data to come through to get an accurate estimation of fitness and little enough so that the genetic algorithm's improvement did not take too long. The amount of simulation time given per generation will be discussed in the Results chapter as a factor that is controlled for each test due to this.

## 3.4 Testing Criteria and Metrics

The testing criteria has been designed with answering the research question in mind.

### 3.4.1 Testing the Effect of Selection Method

As there had been two kinds of selection method developed, it made sense to test them both in a comparative way, in order to identify which method was more suitable for application in the program. The tests then were designed to single out the individual factor of selection method. With all other changeable variables and methods constant in each test, the variables kept constant and their values are identified in the Results chapter. They were each run for twenty-thousand generations, which was predicted to be an appropriate length to observe short and longer term trends in the data. This analysis was useful as it meant potential swaps in effectiveness over time were unlikely to be missed.

### 3.4.2 Testing the Effect of Mutation Probability

Measuring the effect of mutation probability on the change in fitness over generations was another test. This test was carried out in order to identify the best potential mutation rate, and to see how much of an effect it has on the fitness values of the boids.

### 3.4.3 Controlled Factors

In a simulation with so many variables, many required consideration as control variables, to identify trends which might otherwise be obscured. In this section, factors will be identified as necessary to control for when running the simulation and justifications given. Factors that need to be controlled have been mentioned alongside the results produced where relevant.

Initialising populations was controlled due to two methods being developed in the application; having this controlled meant assurance that there was no effect from changing the population initialisation method. As there were also two different selection methods developed, controlling for selection was important as it was necessary to know in order to make sure runs of the simulation were not accidentally mixed and matched when relevant to producing results, as this could affect the quality of analysis around them and may have resulted in inaccurate conclusions. The selection was always the same during a run of the simulation and so was a factor to consider when producing the data sets. However, this did not need to be controlled for when looking for potential points of convergence in values for each of the boids alleles, as this pointed toward potential maximas of the solution and was not specific to a selection method.

The probability of mutation was a control factor; maintaining a fixed value meant the tests could be compared in a temporal manner across generations, especially considering the potential for convergence of data over time. The amount of boids placed in the scene each generation being kept constant meant that the amount of competition for resources was constant across generations and simulation runs. This was similar to the number of resources in the scene, this amount of resources meant more competing with the other flock over food resources; increasing it or decreasing it in different runs could have had an impact on the strategies the flock developed.

### 3.4.4 Data Collection

Data was collected at the end of each generation into CSV files and then collated and used in their relevant contexts.

# 4 Results

## 4.1 Testing the Effect of Selection Method

The aim of this test was to identify the effects, if any, which the sampling method had on the genetic algorithm. Two implementations of sampling method were used in the program: Roulette Wheel (RW), and Stochastic Universal Sampling (SUS). Two tests were conducted, using each sampling method respectively, the below factors were standardised between the tests:

- Number of Generations: 20000
- Population Generation: Randomised Method
- Mutation Probability: 1%
- Time Step: 1.5 Simulation Seconds per frame
- Generation Length: 120 Simulation Seconds

The data displayed in Fig.8 and Fig.9 was from all 50 boids over 20,000 generations laid on top of each other. The reason for this is that the data is very similar and should be seen to represent broader trends rather than specific values.

### 4.1.1 Roulette Wheel Data

Fig.8 shows the fitness trends over all the generations run; this helps to identify the effect RW selection had on the fitness of the boids in the flock.

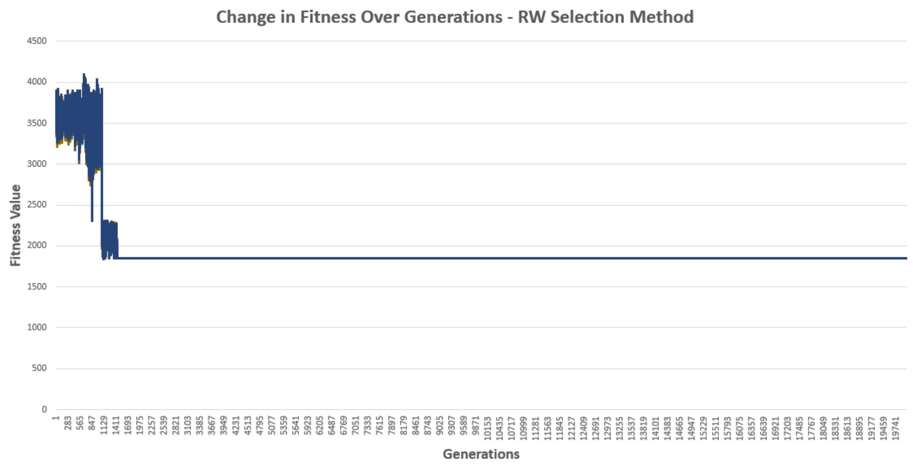

Figure 8: A graph displaying the fitness each generation using RW selection

### 4.1.2 Stochastic Universal Sampling Data

Fig.9 shows the fitness trends over all the generations run; this helps to identify the effect SUS selection had on the fitness of the boids in the flock.

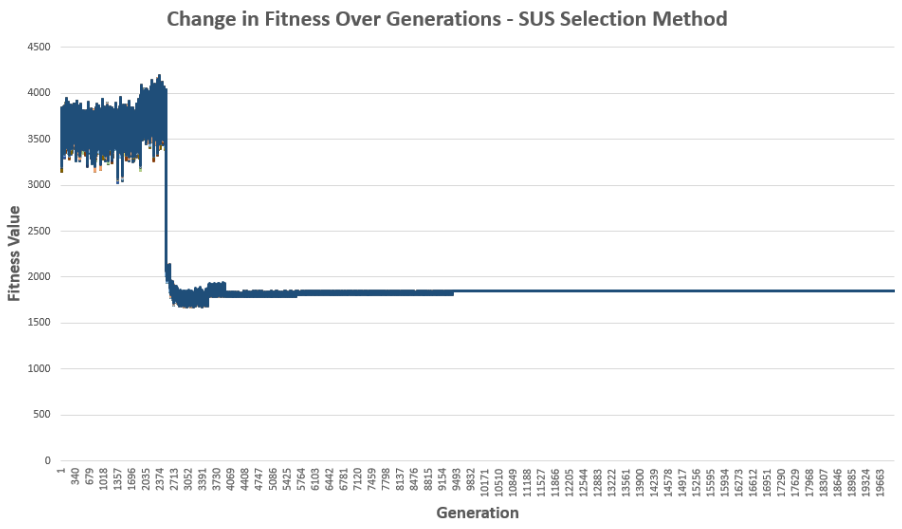

Figure 9: A graph displaying the fitness each generation using SUS selection

in fig.9 the fitness value can be seen to have stayed roughly the same for 2000 generations (Fitness approx. 3600), increased briefly for roughly 500 generations after that (Fitness approx. 3800), and then decreases abruptly to a much lower fitness value (Fitness approx. 1800) from approximately 2500 generations onwards for the remainder of the generations.

## 4.2 Testing the Effect of Mutation Probability

This was a test to identify the effect mutation probability had on the genetic makeup of the flock and to identify the impact it may have had on the effectiveness of the genetic algorithm as a whole. This required standardising the tests in each run of the GA, these were:

- Number of Generations: 5000
- Population Generation: Randomised Method
- Selection Type: Stochastic Universal Sampling
- Time Step: 1.5 Simulation Seconds per frame
- Generation Length: 120 Simulation Seconds

Keeping these factors controlled across all tests, 10 tests were then produced with different mutation probabilities: 0.5%, 1%, 2%, 3%, 4%, 5%, 10%, 15%, 20%, and 25%.

### 4.2.1 The Effect on Fitness

Below is the data collected regarding the fitness value of the boids over each generational improvement. This data will shine a light onto the effect the mutation rate has on the boids' fitness. This data belonging to each mutation rate can be seen in figs. 10 to 19 respectively, in ascending order.

The fitness data in fig.10 trends downward very slightly, until approximately the 1000 mark, where it takes a steep drop to a fitness value of approximately 2250 and fifty from approximately 3400. It then stays that way for approximately another thousand generations before dropping again to an approximate fitness value of 1600. Fitness then stays at approximately that value for the remainder of the run.

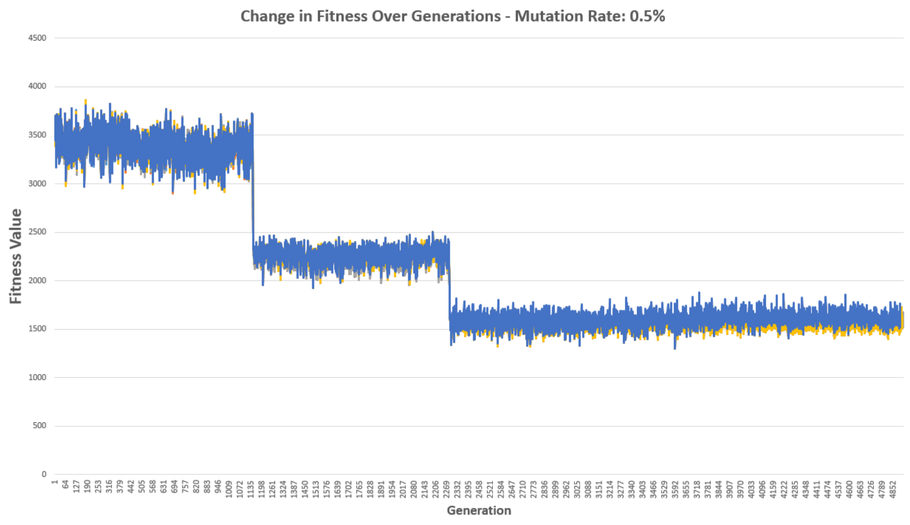

Figure 10: A graph displaying the fitness each generation with a mutation rate of 0.5%

The fitness data in fig.11 maintains an average value of approximately 3600 for all 5000 generations.

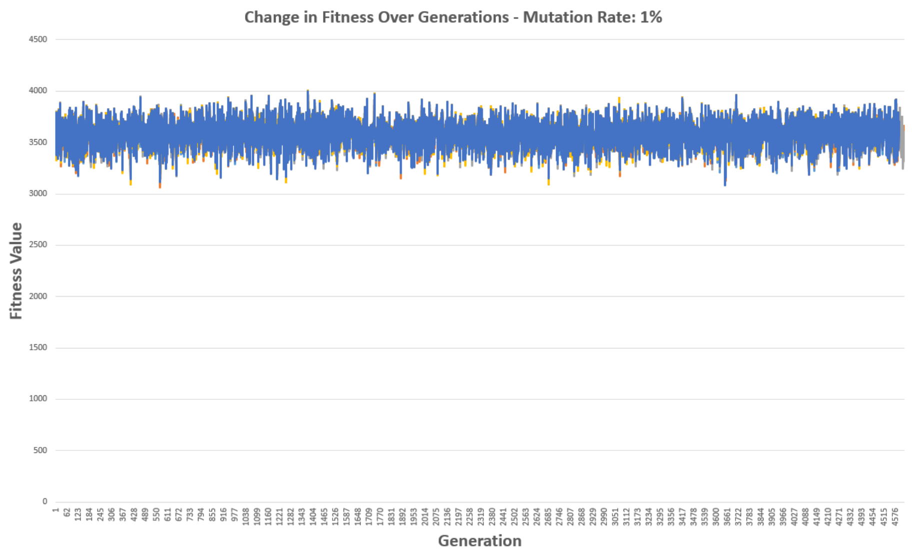

Figure 11: A graph displaying the fitness each generation with a mutation rate of 1%

The fitness data in fig.12 maintains an average value of approximately 3600 for almost all 5000 generations, where in the last couple of hundred generations it drops down to an average fitness value of approximately 1700.

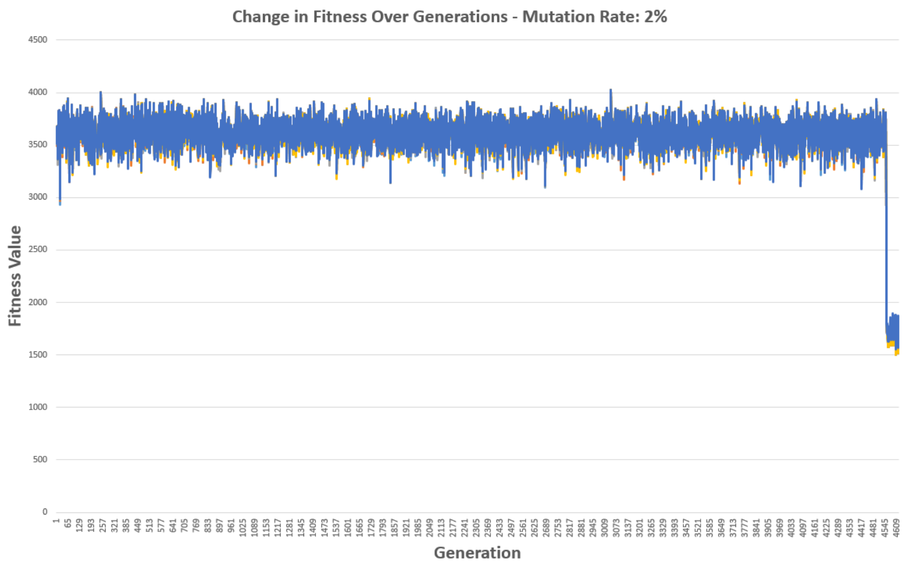

Figure 12: A graph displaying the fitness each generation with a mutation rate of 2%

The fitness data in fig.13 maintains an average value of approximately 3600 for the first 2000 generations and then drops down to an average fitness of approximately 1600 for the remainder of the run.

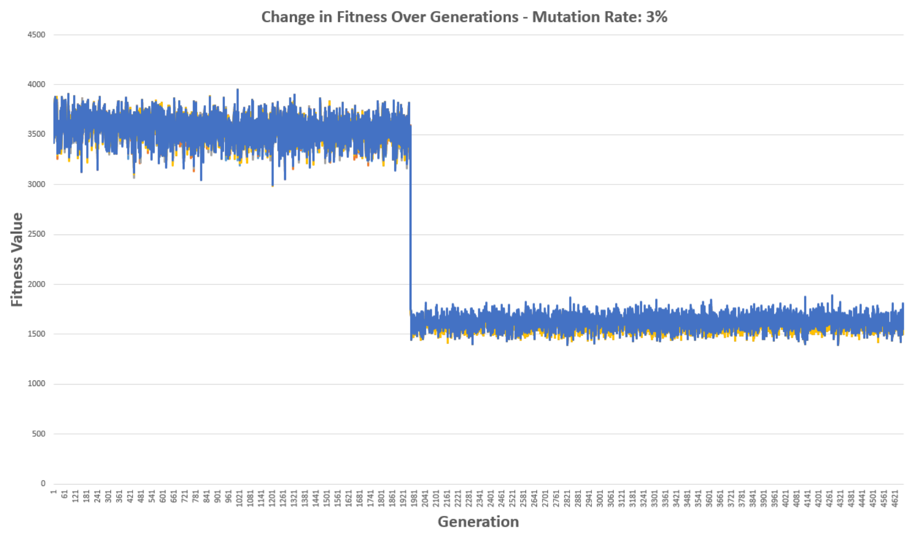

Figure 13: A graph displaying the fitness each generation with a mutation rate of 3%

The fitness data in fig.14 maintains an average value of approximately 3600 for the first 3500 generations and then drops down to an average fitness of approximately 2250 for the remainder of the run.

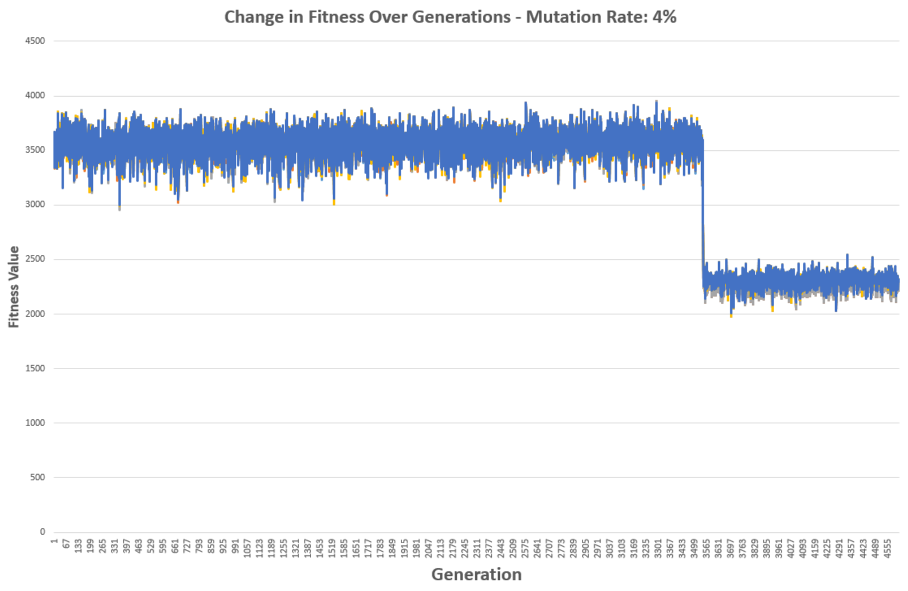

Figure 14: A graph displaying the fitness each generation with a mutation rate of 4%

The fitness data in fig.15 maintains an average value of approximately 3500 for all 5000 generations.

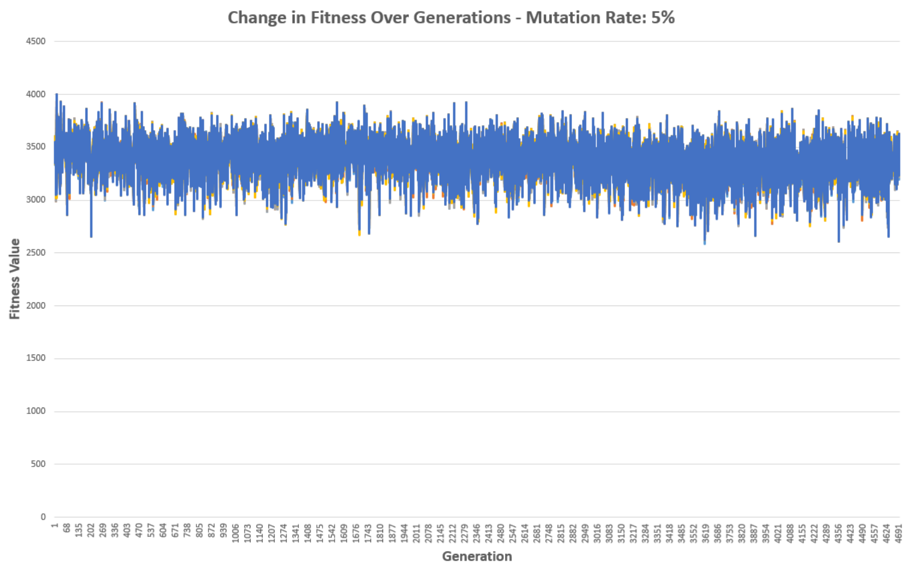

Figure 15: A graph displaying the fitness each generation with a mutation rate of 5%

The fitness data in fig.16 maintains an average value of approximately 3500 for all 5000 generations.

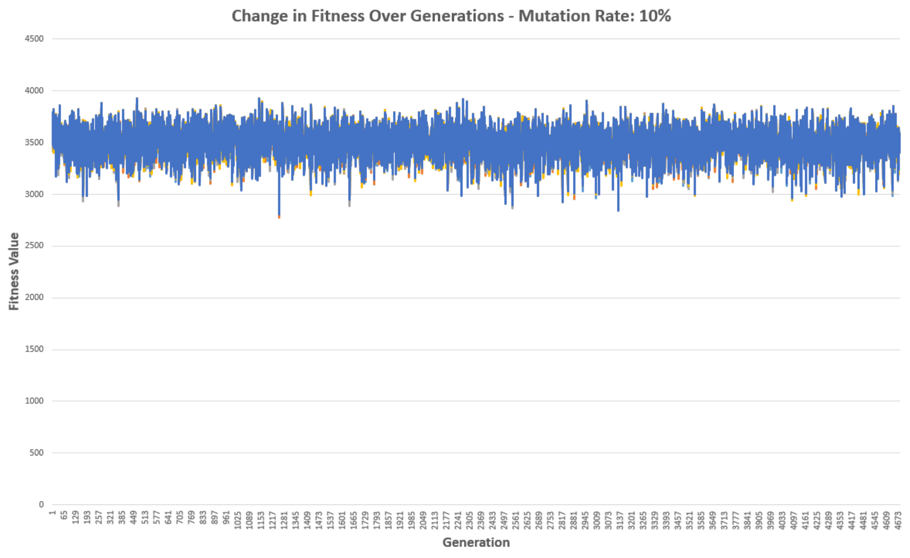

Figure 16: A graph displaying the fitness each generation with a mutation rate of 10%

The fitness data in fig.17 maintains an average value of approximately 3200 for all 5000 generations, with some variation along the way.

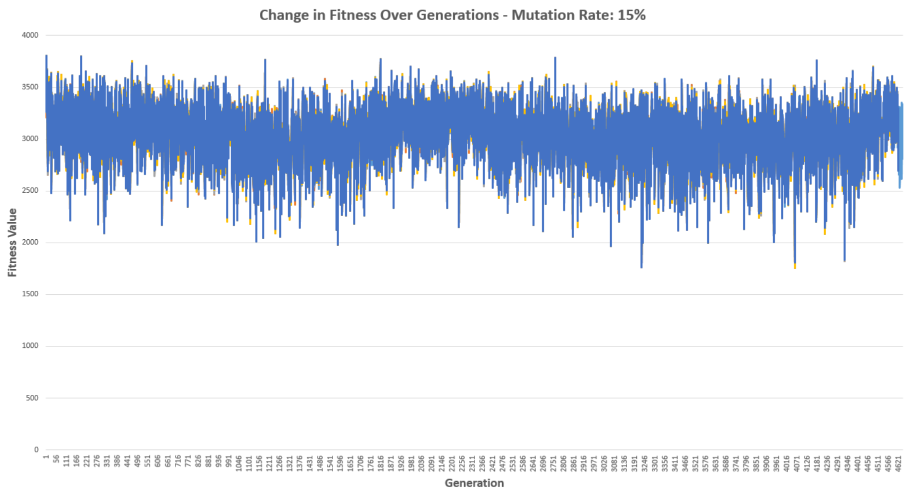

Figure 17: A graph displaying the fitness each generation with a mutation rate of 15%

The fitness data in fig.18 maintains an average value of approximately 3600 for all 5000 generations.

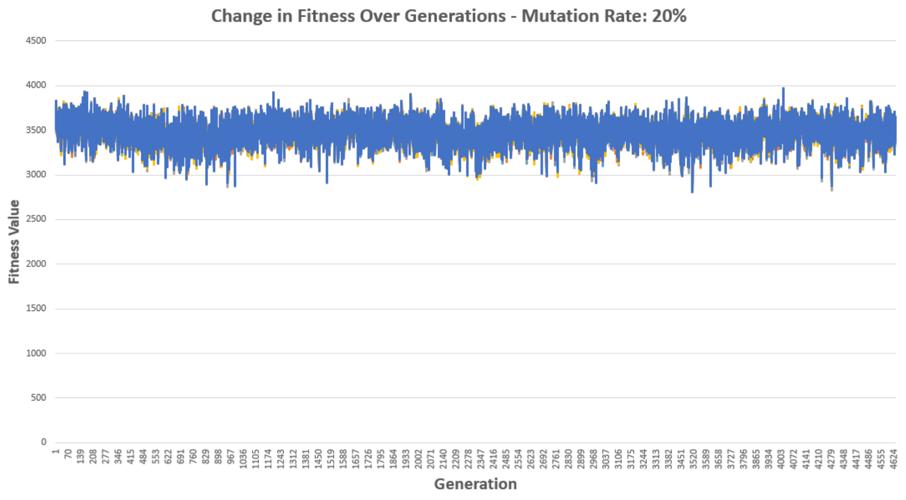

Figure 18: A graph displaying the fitness each generation with a mutation rate of 20%

The fitness data in fig.19 maintains an average value of approiximately 3500 for the first 2800 generations and then drops down to an average fitness of approximately 2250 for the remainder of the run.

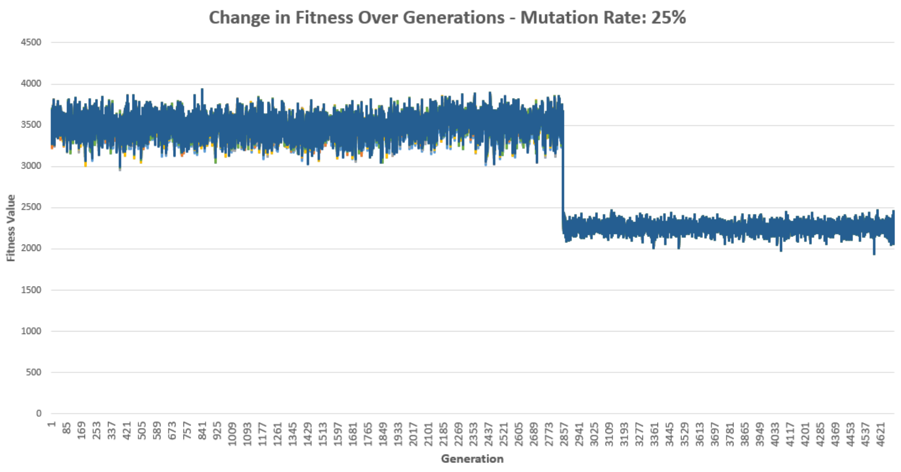

Figure 19: A graph displaying the fitness each generation with a mutation rate of 25%

## 4.3 Allele Convergence

Using the data collected from the previous tests, the aim was to analyse the frequency of different alleles to identify the most frequently produced values in the simulation, as this may point toward optimal solutions for the boid algorithm.

## 4.4 Strategies

By visually analysing the flocks in action, the aim is to identify the ways in which the genetic algorithm flock can outcompete the other flock. Below in fig.20, 22 and 21 screenshots of potential strategies the genetic algorithm may have developed. Qualitative observations can be made to give furhter insight into the impact of the genetic algorithm on the boids' end behaviours, and on the patterns and causations which emerge in terms of interaction between the flocks.

Here in fig.20, we can see that the boid (blue) going for the resource (green) is causing the other flock (red) to react by moving away from the food source.

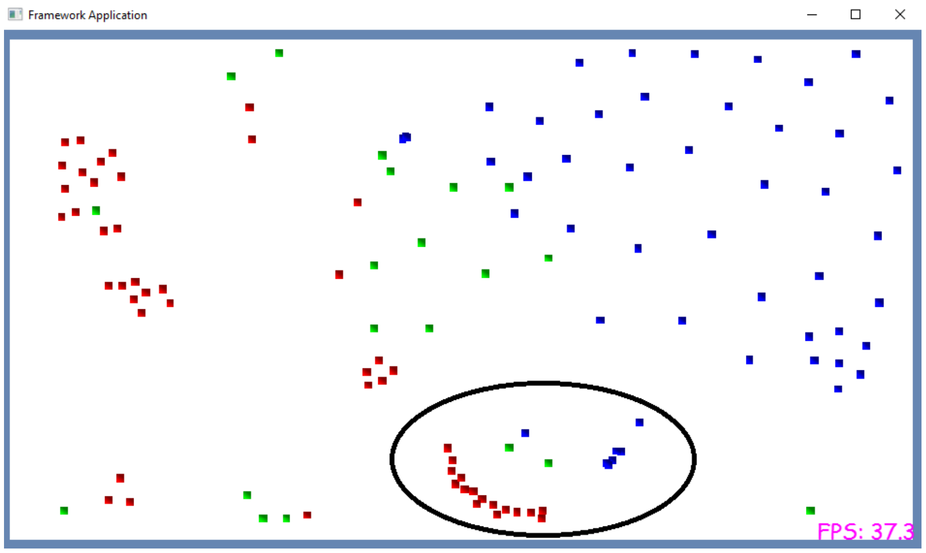

Figure 20: Screenshot: In the circle a boid moves forward to get the resource, causing the other flock to move away

This screenshot in fig.21 displays the advantage of spreading out, a resource (green) has spawned inside of the flock; this causes the boids in that position to move in.

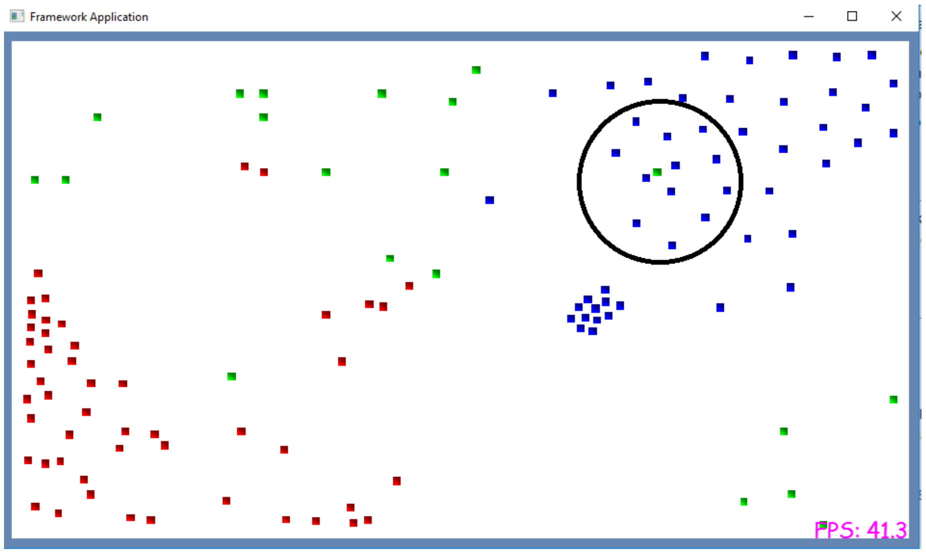

Figure 21: Screenshot: In the oval we can see boids moving in on a resource, taking advantage of the spacing

In fig.22 the advantage and pitfall of a high food attraction force can be seen, a resource (green) has spawned inside of the flock, this causes the boids in that position to move in.

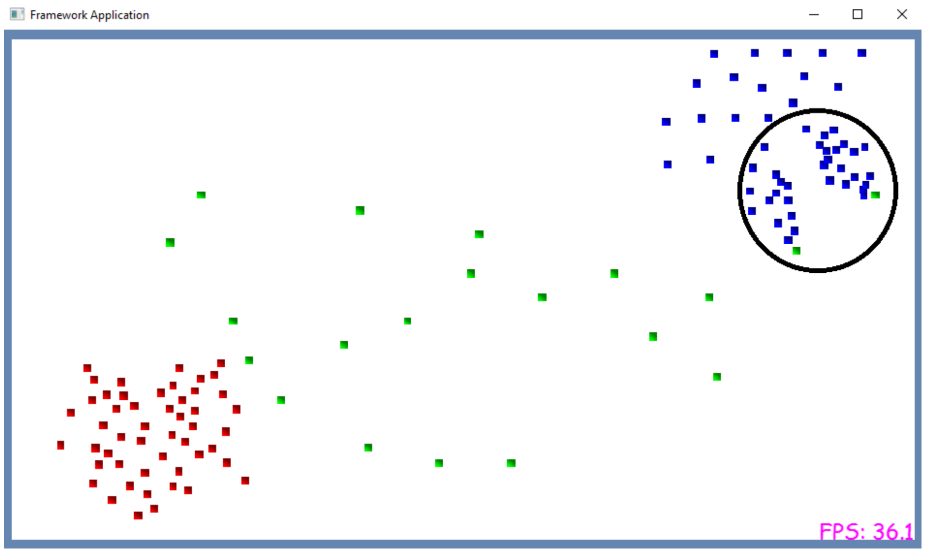

Figure 22: Screenshot: In the circle, the advantage and disadvantage of a high food attraction force

# 5 Discussion

This chapter will evaluate the potential significance of results from the tests carried out, by highlighting key trends and producing observations in line with the research aims.

## 5.1 Sampling Method

By identifying the better sampling method, the aim was to improve the ability of the genetic algorithm to improve itself over time. From the data collected from each it can be concluded that Stochastic Universal Sampling (SUS) selection was a more effective method than Roulette Wheel (RW) selection at producing a viable flock.

This is significant because it means further tests should be run using the SUS selection method to produce results which are further improved on those presented here. By picking two parents randomly instead of one, there is increased genetic variation and therefore a higher chance of survival should issues crop up. For example, the RW selection method encountered the boid health issue that deteriorates the genetic algorithm's fitness far earlier than the SUS selection method; this could potentially be because there was less genetic diversity to stave away bugs this way, or it could be sheer random chance that continues to occur because of the law of large numbers.

The SUS run was the only test to display a slight sustained significant increase in the fitness function: something all other tests have failed to do (including those that use SUS selection while testing the response to different mutation rates).

Other sampling methods that could have been analysed would have been Tournament Selection and Rank Selection. These sampling methods would be worth testing for suitability for the genetic algorithm, to identify in which contexts they would be useful or if one is more efficient than the two already discussed. The advantage of both selection methods would be that they could work properly with negative values too, lessening the need for the weights added to the fitness function.

## 5.2 Mutation Probability

Through testing a variety of mutation probabilities, the aim was to identify what effect (if any) changing the mutation probability had on the effectiveness of the genetic algorithm coming to a better solution over time and why.

The main conclusion from the data seen in figs. 10 to 19 is that the mutation rate generally didn't affect the average fitness of the boids over successive generations. At first, it seemed like it actually decreased over time – however, this was found to be a bug in the program as they are giant leaps and all linked to the same issue which is discussed in the fitness section (quantitative results) of the Results chapter.

What can be seen however, is that changing the mutation probability does not have a significant effect, positive or otherwise, on the fitness of the boid in the way the mutation is implemented in the application. This does not mean it had no effect anywhere in the application, but does mean that changing the mutation rate will not yield the increase in fitness that would be ideal for the project to have made a firm conclusion.

## 5.3 Fitness Results

With only one brief exception, the fitness value in each test - on average - only ever stayed approximately the same or suddenly decreased dramatically, where it then resumed its stability at the lower value. This shows that the genetic algorithm as it is implemented in the environment will most likely not improve against its fitness function on average. If the fitness increases over generations then it is a good indicator that the flock improved by the genetic algorithm is out-competing the regular flocking algorithm, and vice versa.

### 5.3.1 Addressing the Significant Drops in Fitness

In most runs of the genetic algorithm after a few thousand generations, the fitness would drastically drop (in rare instances, this would happen twice). These were also consistent drops and averaged either approximately 2300 or the lower 1700. The fact that these drops had such consistent magnitude means it is an issue that can be attributed to a select range of causes, if not one.

When looking at the data, one consistent value sticks out: the boid health is consistently around a value of 48, and only in the flock improved by the genetic algorithm. This can be identified by analysing and comparing the boid health, and other flock health from each generation in tests which had this problem. In fig.23, the boid health can clearly be seen to drop to a value of 48 unexpectedly, and no longer rise (health should be reset to one-hundred each generation so this should not happen so consistently). Below is fig.23, which is included to demonstrate the moment all boid health values turn to 48, which lines up with that of the SUS fitness data.

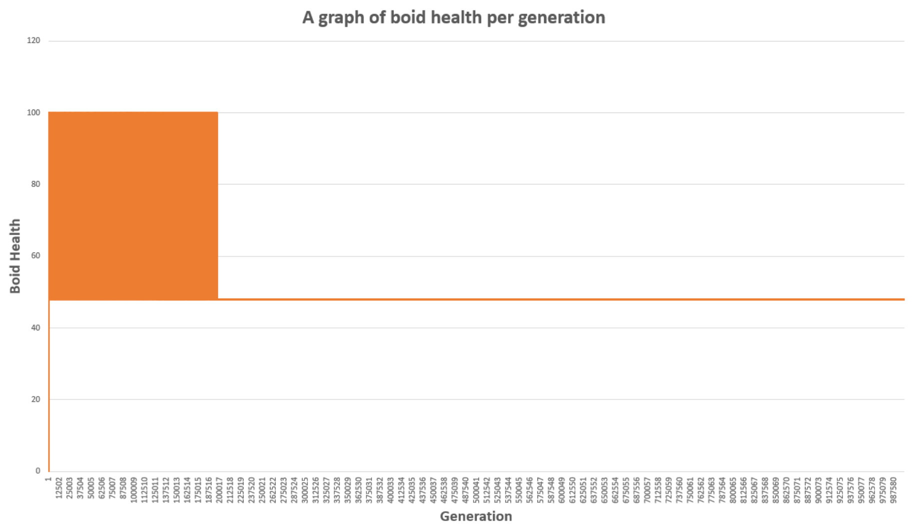

Figure 23: This graph highlights the moment the all boid health results equal 48

Every time this bug effected the simulation, it prevented any further meaningful results from being taken; this was an issue to the project when collecting data and limits the value of those results as beyond that point there are no changes to the data and so therefore no meaningful observations can be made on those following results.

### 5.3.2 The Exception

The only exception to the general conclusion of the fitness results discovered, was with the Stochastic Universal Sampling (SUS) selection test, in which the boid fitness takes a jump up to an average of a few hundred higher than the general average of 3500. Identifying which change in the genetic algorithm was the cause behind this increase in fitness may be key to designing a genetic algorithm that better improves a flock in a competitive environment.

## 5.4 Competitive Strategies

The aim here is to identify and explain behavioural patterns which may have been advantageous to the flock improved by the genetic algorithm. The first and most obvious advantage produced is the genetic algorithm having a weaker flock avoidance force than the regular flock; this was found in the behaviour seen in fig.**??**. However, it doesn't seem to be an adaptation: more something that consistently comes out of the random population initialisation at the start of each run. Regardless, it is an advantage for the flock, as it means when a boid from each flock comes racing for a resource to eat, the boid with the lowest ‘other flock avoidance’ - particularly in an environment such as this one, where there is no risk of harm from getting too close to a boid from another flock – will benefit from a higher chance of survival.

Secondly, an advantage in this simulation lies in the boids’ expanding over a broad area while keeping in contact proximity of other flock members to be alert of food resources outside of the. This means there is a larger surface area held by the improved flock, in which to catch resources as they spawn in the scene. This has the additional, territorial benefit of keeping the competing flock out of the area.

Another advantage observed was in the improved flock having a much larger attraction to food sources than the product of the regular boid algorithm. This meant that, when resources were detectable, the boids went immediately toward the food resource in question - great in a competitive environment. However, food attraction was high enough that it overwhelmed any separation force and caused the boids to overlap when moving toward a resource until it was consumed, which is not a realistic behaviour.

These strategies aren’t necessarily on-purpose or developed per the genetic algorithm. This lack of purpose is generally confirmed, as random values tend to produce these results regardless and so these are less strategies and more the end result of random large values. So more strategies need to be investigated alongside a fitness increase to see if the GA has in fact adapted itself a strategy or not. What it does show is that there is potential for improvement from the genetic algorithm to find new ways of increasing its fitness value, and competitiveness versus a regular flocking algorithm.

## 5.5 Evaluation of Application

The compatibility of flocking behaviour and the competitive environment in which it is placed is questionable from the results gathered, and specifically the lack of improvement in fitness over generations. Flocking is a group behaviour that has several advantages but mainly: less individual effort in avoiding predators, decreased energy cost of movement, and an increased ability to find food. The environment is currently able to produce one of these factors: that is it is much easier to find food as the boids pull on each other, and generally in the direction of food.

To improve this application and test in future, implementing a decreased energy cost of movement is something that would be interesting to factor in. Currently, hunger (or energy level) runs down at a flat rate but this could be modified to decrease at an increasing rate which correlates to the distance moved out with the alignment with nearby flock members, representing a ‘streamlining’ buff to the boids. This could be implemented via a drag factor, which could be a combination of the currently implemented drag force and a unit vector to determine direction. With a few changes to the magnitude of the drag force, this drag force could then be linked to a boid’s energy or hunger levels and make it deplete faster.

Predatory flocks or predators in general are not currently implemented but would be an interesting dynamic to introduce. The boid algorithm would need to incorporate a reaction force to a predator entity in the environment; health damage functionality would also need to be introduced. To go even further, introduce the ability for prey flocks to cause rebuttal damage to a predator build would be a fascinating mechanic to implement in a limited environment, to see how the genetic algorithm would adapt to flee or fight dependent on context. At this point it may be necessary to start adding a state machine layer on top, so that a boid could adapt within the scene.

**Efficiency** The resulting boids algorithm could have been more efficient. Instead of checking against all other boids, resources and the other flock's boids every frame, the Barnes-Hut method would have been an excellent way of partitioning space so that boids were only able to interact with relevant surroundings. This would've essentially divided the area up via a quad-tree, and so each entity only has to check against local entities in their area.

**The Genetic Algorithm** The genetic algorithm's fitness function, as it stands, is relatively simplistic, with only three weighted factors considered: boid health, flock health and 'other flock health'. This was designed so that the algorithm would try to improve on those three metrics, not ignoring the health of the competing flock if it were too high: making sure that it helped the flock overall and didn't just focus on its own health (which is a much smaller factor to accommodate for this). The design of the boids could have been improved to give the genetic algorithm more of a solid founding to work with. For example, resources could have been designed so that each would have an area bonus to health and hunger for the flock members near to the boid which consumed it. By introducing an incentive by way of some health released, this may have affected some interesting patterns of behaviour, and could have been produced through the resource class update method. Boids could also have been designed to interact with other flocks in more complex ways; introducing a state machine boids could have had emotional or physical states to switch between, introducing more variation to behaviour.

The environment was designed to be competitive so that the genetic algorithm enabled (improved) flock could work to improve itself against the regular flock, however this has not been the case in any clear consistent manner, so some redesign is in order. More options may have been available for movement if a wraparound of the boundaries was done effectively - that is, not only would it update the position vector of the boid, but also its direction and its ability to interact with boids now on the other side of the map, but also technically next to each other. Different types of flock could have also been introduced into the environment to see if it was the flock type stopping the genetic algorithm from improving on its quantitative fitness. The most appropriate thing to do would be to test each feature in order to narrow down what features of the test environment have an impact on end behaviour and how. It would also be pertinent to look at what other people have done in regard to this kind of issue in other fields using genetic algorithms.

The genetic algorithm itself clearly requires a lot of data tests to identify the particular areas in which it is flawed. The genetic algorithm already has an improvement in selection method via the two examples tested (Roulette Wheel and Stochastic Universal Sampling), and mutation rate has also been isolated as to not be the culprit - assuming there is one.

# 6 Conclusion

An application was produced to investigate the impact a genetic algorithm had on the dynamic interaction of flocks with each other, and identify if the improved flock could outcompete that produced from a regular flocking algorithm. It was also produced to fulfil the aims and objectives presented at the start of the paper. The results and end conclusions of this paper shall be summarised here.

This research has identified that the improved (genetic algorithm) flock has not been conclusively found to exhibit better or worse fitness levels over generations. This finding is based on fitness function calculations which saw little variation, and by identifying the cause of dramatic decreases in the fitness of the boids as a bug which needs to be located and fixed.

The aim of the project was to investigate if a flock improved by a genetic algorithm can outcompete a regular flock in a competitive environment, and to identify: what impact the genetic algorithm had on behaviours and strategies, the kinds of behaviours produced over time, and was to be met through the objectives outlined in the Introduction chapter.

Relatively no evidence was found to support any increased effectiveness of the genetic algorithm, in terms of improving itself in a complex environment. This was drawn from the evidence of the fitness values staying more or less the same throughout every simulation (barring errors).

An application was produced which models flocking behaviour, allowing for the observation of the genetic algorithm's flocking strategies against a regular flocking algorithm. Through qualitative analysis it was found that there were three visible strategies that the genetic algorithm could use to give itself an advantage, these were: spreading out over a wider area, a lower flock avoidance weighting and an increased food vector weighting - with all three leading to a healthier, more fit flock. Despite the fact that the designed application failed to meet all of its intended data outcomes, it was an example of good practice in trial-and-error in research. The resulting algorithm has a lot of potential for development and addition of other features, as detailed in the Discussion chapter. Tentative qualitative trends suggest that, with further modifications and development, there may be potential for this kind of application taken forward, to eventually start producing positive trends in increased boid fitness. Eventually, with practice and improvement, this kind of application might even yield some light on natural flocking behaviours which could be relevant to those in the field of natural sciences, studying animal behaviour. In this way there was some benefit to introducing the genetic algorithm, and through the lessons learned through this work, there has been a lot of potential introduced here for further research and development.

## References

Al Toufailia, H., Couvillon, M. J., Ratnieks, F. L. W. and Grüter, C. (2013), 'Honey bee waggle dance communication: signal meaning and signal noise affect dance follower behaviour', *Behavioral Ecology and Sociobiology* **67**(4), 549–556.
**URL:** *https://doi.org/10.1007/s00265-012-1474-5*

Chen, Y.-W., Kobayashi, K., Huang, X. and Nakao, Z. (2006), Genetic algorithms for optimization of boids model, Vol. 4252, Springer Verlag, pp. 55–62.

Dorigo, M. and Gambardella, L. M. (1997), 'Ant colonies for the travelling salesman problem', *Biosystems* **43**(2), 73 – 81.
**URL:** *http://www.sciencedirect.com/science/article/pii/S0303264797017085*

Flake, G. (1998), *The Computational Beauty of Nature: Computer Explorations of Fractals, Chaos, Complex Systems, and Adaptation*, A Bradford book, Cambridge, Massachusetts.
**URL:** *https://books.google.co.uk/books?id=0aUhuv7fjxMC*

Hartman, C. and Benes, B. (2006), 'Autonomous boids', *Computer Animation And Virtual Worlds* **17**(3-4), 199–206.

Husselmann, A. and Hawick, K. (2011), Simulating species interactions and complex emergence in multiple flocks of boids with gpus, *in* 'Proc. IASTED International Conference on Parallel and Distributed', IASTED, pp. 100–107.

Jaderberg, M., Czarnecki, W. M., Dunning, I., Marris, L., Lever, G., Castaneda, A. G., Beattie, C., Rabinowitz, N. C., Morcos, A. S., Ruderman, A., Sonnerat, N., Green, T., Deason, L., Leibo, J. Z., Silver, D., Hassabis, D., Kavukcuoglu, K. and Graepel, T. (2018), 'Human-level performance in first-person multiplayer games with population-based deep reinforcement learning'.

Jorge, P. E. and Marques, P. A. M. (2012), 'Decision-making in pigeon flocks: a democratic view of leadership', *Journal of Experimental Biology* **215**(14), 2414–2417.
**URL:** *http://jeb.biologists.org/content/215/14/2414*

Kawabayashi, H. and Chen, Y. (2008), Interactive system of artificial fish school based on the extended boid model, *in* '2008 International Conference on Intelligent Information Hiding and Multimedia Signal Processing', pp. 721–724.

Reynolds, C. W. (1987), 'Flocks, herds and schools: A distributed behavioral model', *SIGGRAPH Comput. Graph.* **21**(4), 25–34.
**URL:** *http://doi.acm.org/10.1145/37402.37406*

Shang, Y. and Bouffanais, R. (2014), 'Consensus reaching in swarms ruled by a hybrid metric-topological distance', *The European Physical Journal B* **87**(12), 1–7.

Shiffman, D. (n.d.), ‘Flocking’.
**URL:** *https://processing.org/examples/flocking.html*

Watts, I., Nagy, M., de Perera, T. B. and Biro, D. (2016), ‘Misinformed leaders lose influence over pigeon flocks’, *Biology Letters* **12**(9), 20160544.
**URL:** *https://royalsocietypublishing.org/doi/abs/10.1098/rsbl.2016.0544*