Evolving Boid Flocking Behavior

Ryan McArdle, Gianni Orlando, Olusade Calhoun,
Alexander Costa, Zachary Sipper
17 December 2020
Institute for Artificial Intelligence
University of Georgia
Athens, Georgia, USA
rmcardle@uga.edu, gianni.orlando25@uga.edu, olusade.calhoun25@uga.edu,
alexander.costa@uga.edu, zachary.sipper@uga.edu

I. Abstract

Given their relationship with the field of artifical life, it is no surprise that Reynolds' boids have been explored using evolutionary approaches. Often, researchers seek to evolve boid like behavior either using Reynolds' definitions as a motivation or as a goal point. In this research, we use Reynolds' definition as our basis and seek to evolve realistic boid flight behavior by evolutionarily optimizing the parameters which define the behaviors of each boid. We utilize a previously trained set of classifiers which are intended to recognize 3 behaviors associated with swarm flocking to develop our fitness function and compare the best species of boids found by generational and steady state genetic algorithm approaches, as well as (μ, λ) and $(\mu+\lambda)$ evolutionary strategy approaches. We also explore the effects of various ratios between μ and λ for the evolutionary strategies. While we are unable to find a species of boid which optimizes all behaviors, we are able to gain a better understanding of the fitness space and suggest that a $(\mu + \lambda)$ approach is the most likely strategy for obtaining success under this fitness function. We are also able to corrobrate suspicions that the classifiers used do not generalize well beyond their original training set and support the hypothesis that swarming behaviors are difficult to formalize in a computational classifier.

II. Introduction

A. Boids

The concept of boids was first introduced by Craig Reynolds in 1987. His intent was to introduce a distributed behavioral model which would allow for the efficient simulation of the flocking behavior found in a wide range of natural animals, including flocks of birds, schools of fish, and crowds of humans [1]. Each individual in the simulated flock is a single boid, and the distributed behavioral model approach means that rather than having to predefine the path of each boid to produce a convincing flocking behavior, one could simply assign a set of rules to each of the individuals that relate the individual to its surroundings, and believable flocking behavior would emerge from the collective. At their core, these rules define three goals which each boid would strive to achieve: alignment, which is matching its velocity with other nearby boids; cohesion, which is keeping itself both near to and surrounded by other boids; and separation, which is avoiding collisions or sharing the same space with other boids.

There are a number of various parameters that can go into the formulation of these rules and the boids' goals. One can manipulate how far away a boid can be while considered a 'nearby' boid, the importance of each of the different goals by weighting their influence on the boid's behavior, or how quickly the boid is able to accelerate and respond to changes in its environment. As such, the flight behaviors of boids lend themselves quite readily to being manipulated and optimized through genetic and evolutionary approaches, allowing us to explore the space of possible 'species' of boids and fine-tune those parameters to select for a certain kind of behavior.

B. Related Works: Boid Evolution

Given the common philosophies regarding the development of algorithms based upon natural exemplars, it is no surprise that the intersection of evolutionary algorithms and boids, a prime example of artificial life, have been explored before.

Neural networks have been evolved which are capable of learning flocking behavior using the sensors and activators on swarm robots by evaluating fitness using metrics motivated by Reynolds' original rules. These results found that the alignment behavior is not strictly necessary for individuals in a swarm to flock together and that the separation and cohesion forces are sufficient, but also that these rules are most effective when these metrics can be evaluated globally over the entire swarm, rather than just locally over the nearby neighbors for each individual [2].

Flocking behavior has been capable of evolving from relatively few features, such as a neural network which controls the movement of individuals in a swarm that are capable of signaling to each other with a single simple signal with an intensity ranging between 0 and 1. When the signaling behavior is enabled between individuals, their population is capable of evolving to swarm around a sustaining food source, while the silencing of the signal leads to erratic behavior and a population which dies out [3].

C. This Work

Although it has been suggested that the Reynolds implementation of flocking behavior in the boids model is somewhat simplistic and that we should define artifical life in such a way that the boid model is an explicit subset of artificial behavior [4], we approach this problem using Reynolds' original formulation of the boids model. We seek not to evolve Reynolds' rules based on neural networks and some available mechanism, but rather to explicitly implement Reynolds' rules and then optimize the parameters that specify the relationships between these rules. Our goal is to evolve a species of boid which is capable of most consistently satisfying a group a classifiers which have been previously trained to identify human perceptions of swarm behaviors labeled as 'Aligned,' 'Flocking,' and 'Grouped.' Our hope is to evolve a species of boid that is most likely to be recognized (by this classifier) as exhibiting these swarm behaviors and identify which evolutionary algorithm was most capable of or effective for achieving this goal.

In the rest of this work, we formulate our problem and the fitness function that we will use to evaluate a given species of boid in section III. We then discuss the various approaches that we will explore and compare in section IV, and present the results of these various evolutions in section V. We then compare these results and discuss which method most effectively evolved the best boid species and the obstacles that have been discovered in section VI.

III. Formulation of the Problem

In order to evolve a species of boid which will most exhibit the flocking behavior, we must first formulate the problem as an evolutionary computation problem.

To avoid confusion throughout the discussion, it is important the we establish and clarify the terms that will be used. During our evolution, the individual genotypes that we will be evolving and their associated phenotypes that will be evaluated are what we call a species of boid, i.e., a representation of a specific behavior and the type of boid which exhibits that behavior. This representation of behavior will be given to our simulation, which creates many instances of boids which are members of this species and evaluates their collective behavior as a swarm. Traditionally, evolutionary algorithms are evolving individuals, and the term population references something which exists within the evolutionary structure. For this problem, however, one could also refer to a single boid as an individual and the swarm that it is a part of as a population. It is important to avoid this confusion, so we establish here that our evolutionary algorithms seek to develop an ideal species of boids, not an ideal individual boid.

A. The Species Individual

During our evolution, each potential species of boid will be represented as a list of 6 float values. These values represent, in order:

1) Alignment Weight: This is the multiplicative weight given to the force of alignment in determining an individual boid's acceleration.
2) Separation Weight: The same as above for the force of separation.
3) Cohesion Weight: The same as above for the force of cohesion.
4) Alignment/Cohesion Radius: This is the maximum separation between two boids for the boids to consider each other in their calculation of their alignment and cohesion forces.
5) Separation Radius: The same as above for the force of separation.
6) Maximum Acceleration: This is the maximum magnitude of the acceleration vector for a single boid.

For the evolutionary strategy approaches discussed in subsection IV-B. Evolutionary Strategies, these float values will be accompanied by an additional 6 float values which represent their correlated mutation values.

B. The Simulation

Our boid simulation is modified for our purposes from a simulation made available by Nicolas Rougier [5]. While an effective simulation, each of the parameters which define the behaviors of the boids that we are concerned with are hard-coded with values. To make this simulation effective for our purposes, we modified the initialization of a flock of boids such that it would accept various parameters, including the list of parameters which specify a particular species. We further modify the simulation to record data about the flight of the boids in preparation for passing the relevant information about the simulation to our classifiers, discussed in the next section.

In addition to this ability to simulate various kinds of boids and record data, efforts were made to accelerate the simulation. Given the relatively large number of boids that are expected to be in the flock by our classifiers and the number of simulations that must be run in order to complete our evolutionary algorithm, this step was vital in order to have a fitness function that could be run reasonably quickly. To this end, we utilize the grid search algorithm GriSPy in order to rapidly find all neighbors within a fixed-radius of a given boid [6].

We also update the visualization of the boid information when plotted such that each boid appears as an arrow scaled to its current velocity. Although this has no effect on the simulation or evolution itself, it is very helpful for human visualization and developing an understanding of the behaviors that are evolved, especially when considering stills of a boid swarm which can be seen later in the work in Figs. 13-16.

C. The Classifier

The classifier that is being utilized in our fitness function is one that was trained in part by a subset of the current authors for the purpose of identifying during which time-instances a sample swarm of boids is exhibiting 'Aligned', 'Flocking',

and 'Grouped' behaviors [7]. The classifier was trained using the Swarm Behavior Data Set uploaded by researchers at the University of New South Wales to the UCI Machine Learning Repository [8], [9]. Although certain methodological concerns about the collection of the original data set are raised throughout the training of the classifier, notably the open survey method in which class labels were generated, reasonable results were obtained, and we are hopeful that the classifier should prove fruitful as a heuristic measure of the expression of boid swarm behaviors [10].

We utilize a random forest classifier for each of the three behaviors, which have been optimized using a 10-fold cross validation grid search over parameters such as maximum depth, number of estimators, etc., and we choose the minimum number of training samples required to achieve 95% or higher accuracy. We minimize the training samples since we expect generalizability of these models to other boid simulations to be our primary limitation.

D. The Fitness Function

Having established to components of our fitness function, we are now prepared to discuss the fitness evaluation of any given species of boid.

Our simulation receives a random seed and a specification of the particular boid species in question. When applicable, this random seed is uniform for a whole generation of evaluations in order to evaluate the new boids on similar footing. The simulation then initializes 200 boids in random positions with random velocities (within a maximum) and allows them to begin the flight according to their behavior parameters. The simulation will run for 25 frames, at which point it will begin to collect the statistical data about the flock at every time-instance. After another 200 frames, the simulation will halt, and the recorded information will be passed to the classifier for classification.

The classifier classifies each time-instance based on the statistical information provided and outputs either a 0 or a 1 for each of the 3 behaviors ('Aligned', 'Flocking', 'Grouped'), indicating either a negative or a positive classification for the given behavior. The total number of positively classified instances for each of the 3 behaviors are then summed together with a weighting that depends on the progress of the evolution (discussed below) and divided by the maximum possible score. The value that is returned is between 0.0 and 1.0 and represents the percentage of the ideal possible fitness, which would correspond to expressing all behaviors for every instance of recorded simulation.

Given the results of initial experimentation, we find that a straight-forward summation of scores for each behavior will not provide optimal results. It seems that the 'Flocking' and 'Grouped' classifiers and far easier to satisfy than the 'Aligned' classifier. As such, individuals quickly optimize these two behaviors, while neglecting the other, leading to a take-over of species with a fitness breakdown of [0.0, 1.0, 1.0] for the three behaviors. In order to address this limitation, we introduce a 3-phase fitness function.

The first phase is in effect during the first third of total evaluations. During this phase, only the alignment behavior, the one which seems often neglected, is selected for. Performance based upon the other behaviors is ignored, in hopes of filling the population with individuals that are able to score well for alignment.

The second phase occurs during the middle third of evaluations and begins to consider the other behaviors, but still heavily weights alignment. Two-thirds of the weight is given to alignment, with the other behaviors splitting the other one-third of the weight. We also introduce an overall weighting onto the fitness for an individual. This weighting takes into consideration the overall difference between the fitness scores of each behavior, and can be defined as such:

$$w \equiv \frac{1}{1 + 4\left(d\left(F\right)\right)} \tag{1}$$

where F is a triple $[a, f, g]$ such that a, f, and g are the fitness scores for the 'Aligned', 'Flocking', 'Grouped' classifiers, respectively, and

$$d(F) \equiv \sum_{x,y \in F} |x - y|, \tag{2}$$

which measures the total difference between the elements of a given detailed fitness measure.

This weighting is defined such that a behavior-detailed fitness score F will be given a weight of 1.0 when all three behaviors receive the exact same score and be given a weight of $1/9$ if the scores are as different as possible, either by neglecting or optimizing only one behavior, or assigning the values 0.0, 0.5, and 1.0 to the three behaviors. The intent here is to punish the optimization that the evolution tends towards in hopes of keeping the individuals which satisfy the 'Alignment' classifier from being pushed out of the population.

The final phase occurs in the last third of the evaluations. This phase weights each behavior equally, but continues to weight the overall fitness using eq. 1. Our hope is that in this phase, the population will be able to evolve towards a global optima in which all three behaviors are maximized to 1.0.

While we were able to accelerate a single simulation for a given species, the simulation still takes approximately $10 - 15$ seconds to operate on a powerful personal desktop. With the number of fitness evaluations for a single evolutionary run on the order of 10^4, the computational time for this fitness function is still potentially prohibitively expensive for exploration of models. As such, when possible, fitness evaluations are run in parallel in order to allow a greater number of fitness evaluations to be completed in a given clock time. Unfortunately, this approach does not improve the execution of the steady state algorithm discussed in subsubsection IV-A2. Steady State GA, and we perform that evolution using 10^3 evaluations instead.

IV. Genetic and Evolutionary Approaches

We provide a brief overview of each of the algorithms tested and discuss the particular features selected for our problem. The algorithms included in this work are generational genetic algorithm, steady state algorithm, (μ, λ) evolutionary strategies, and $(\mu + \lambda)$ evolutionary strategies.

Throughout the algorithms, we universally use a population size of 100, a probability for crossover of 90%, a probability for mutation of 30%, and a limit on the total number of fitness evaluations of $10,000$, excepting the steady state genetic algorithm, which is limited to $1,000$ evaluations due to limits on the parallelization of fitness evaluations.

Individuals of the population are randomly initialized from a continuous uniform distribution with the following bounds for each of the six dimensions of each individual:

Parameter	Range
Alignment Weight	[0.01, 2.00]
Separation Weight	[0.01, 2.00]
Cohesion Weight	[0.01, 2.00]
Alignment/Cohesion Radius	[100.00, 300.00]
Separation Radius	[100.00, 300.00] and $\leq$ A/C Radius
Maximum Acceleration	[0.01, 5.00]

resulting in a chromosome structure for the genetic algorithm approaches as follows:

$$\langle x_0, \ldots, x_5 \rangle . \tag{3}$$

For the evolutionary strategy approaches, the six mutation step sizes σ_i are randomly initialized from a continuous uniform distribution with bounds as a function of the corresponding parameter dimension i according to the following formula:

$$\sigma_i \in [0.5 \min(i), 0.2 \max(i)] . \tag{4}$$

This results in a chromosome structure for the evolutionary strategy approaches as follows:

$$\langle x_0, \ldots, x_5, \sigma_0, \ldots, \sigma_5 \rangle . \tag{5}$$

A. Genetic Algorithms

1) Generational GA

In a generational GA, if the population size is P, then every iteration P offspring are created and mutated from the original population to replace them for the succeeding generation. The first defining attribute of our generational GA is the inclusion of elitism. Before any reproduction happens, the two best individuals are copied over to the next generation. This is to combat the inherently explorative nature of a generational GA.

For the rest of the offspring population, a selection tournament of five random individuals from the current generation are evaluated and the top two performers are recombined with one-point crossover, and both resulting children are subject to gaussian mutation at a 30% chance and added to the offspring population. This tournament is executed until the number of produced children is equal to the population size subtracting two, in order to account for the two elitist individuals. Then the current population is discarded, and the offspring population becomes the next generation's population. The only survivors from generation-to-generation are the two elitist individuals.

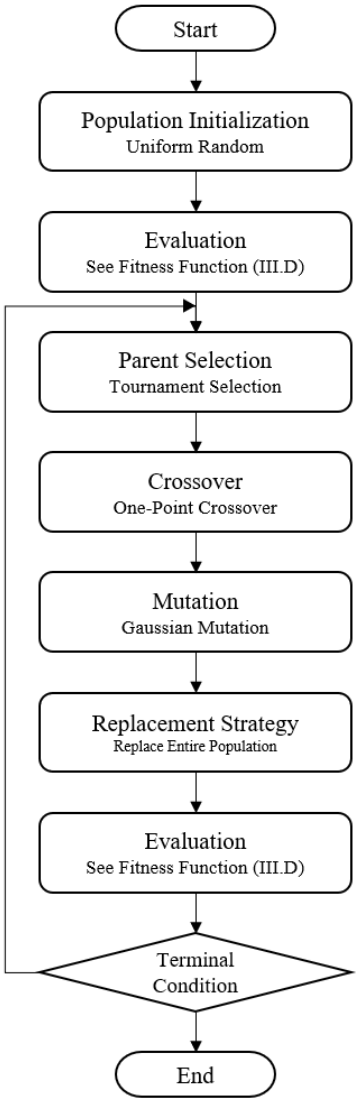

Fig. 1: A flowchart detailing the process of a generational GA.

2) Steady State GA

Steady state generational algorithms are a conservative approach to population manipulation and individual replacement. In steady state algorithms, there are no generations in the general sense, as each generation retains a majority of the exact same individuals with only a few replacement children. These children are created from the best two parents at the end of an epoch and the best of the two parents and two children are added back into the unchanged population.

In our experimentation, our algorithm instead takes the two generated children from the best individuals and replaces the worst two individuals of the population. This methodology may seem too radical for steady state algorithms, but due to limited testing time and computational power, an elitist

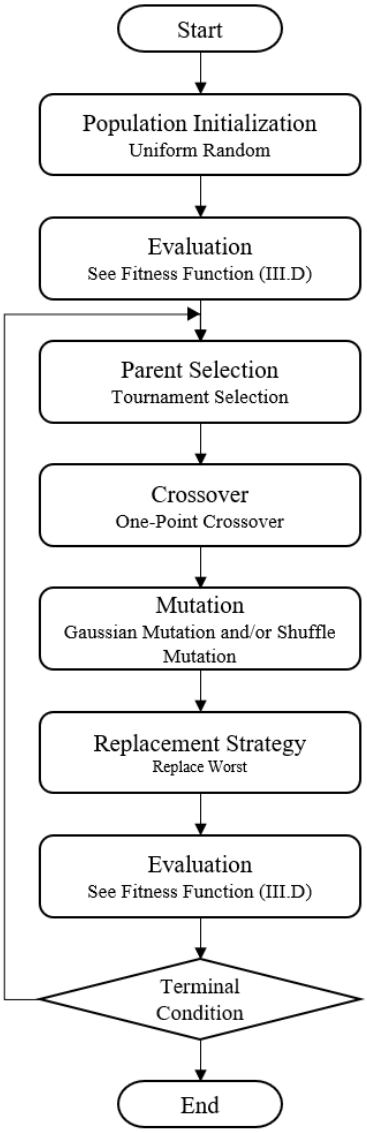

Fig. 2: A flowchart detailing the process of a steady state GA.

methodology was necessary to get results from this algorithm that are comparable to the rest. The elitest strategy also uses aggressive mutation in children to discover optimal boid parameters. Specifically, a mutation probability of 0.6 for both index shuffling and Gaussian mutation creates diverse children to speed up the process of the steady state algorithm.

B. Evolutionary Strategies

For both evolutionary strategy approaches, λ parents are uniformly randomly selected from the μ individuals of the population. A blend mutation operator is used in the crossover step. This operator proceeds by taking the weighted sum of the two parental alleles of each gene, according to the following formula:

$$\begin{aligned} child_1 &= \langle \alpha x_i + (1-\alpha)\, y_i | 0 \leq i \leq 11 \rangle \\ child_2 &= \langle \alpha y_i + (1-\alpha)\, x_i | 0 \leq i \leq 11 \rangle \end{aligned} \tag{6}$$

where $\alpha = 0.\overline{33}$ and x_i and y_i represent the ith allele of parent x and y, respectively.

An uncorrelated mutation with 6 step sizes is used. The use of 6 independent mutation step sizes allows for each dimension of the boid species to be treated separately. The motivation for this separation lies in the observation that the fitness landscape of our solution space may have a different slope in one direction (e.g. along the alignment weight axis) than in another direction (e.g. along the separation radius axis).

In this mutation strategy, first, each mutation step size, σ_i, is mutated by multiplying it by a term e^{Γ} where Γ is the sum of a common base mutation (random variable drawn from a normal distribution with mean 0 and standard deviation τ') and coordinate-specific mutation (random variable drawn from a normal distribution with mean 0 and standard deviation τ). We the mutate each allele, x_i by multiplying it by a random variable drawn from a normal distribution with mean 0 and standard deviation σ_i. The mutation mechanism is specified as follows:

$$\begin{aligned} \sigma_i' &= \sigma_i e^{\tau' N(0,1) + \tau N_i(0,1)} \\ x_i' &= x_i + \sigma_i N_i\,(0,1) \end{aligned} \tag{7}$$

where $\tau' = \frac{1}{\sqrt{2n}}$ and $\tau = \frac{1}{\sqrt{2\sqrt{n}}}$. In these formulas, $N\,(0,1)$ denotes a draw from the standard normal distribution, while $N_i\,(0,1)$ denotes a separate draw from the standard normal distribution for each variable i. The mutation operator was used with a $1/6$ probability of mutation for each attribute to be mutated, meaning that for an individual selected for mutation, that individual will, on average, have one of its six alleles (and corresponding mutation step size) mutated.

An elite selection strategy (choosing the μ highest fit individuals) is used. The (μ, λ) and $(\mu + \lambda)$ strategies simply differ in the set from which the μ highest fit individuals are chosen. In the (μ, λ) approach, the μ highest fit individuals are chosen only from the λ generated children after crossover and mutation. In the $(\mu + \lambda)$ approach, the μ highest fit individuals are chosen from the union of the individuals from the previous generation and the λ generated children after crossover and mutation.

Our evolutionary strategy algorithms are run a number of times using different ratios between μ and λ. We perform runs where $\lambda = 3$, 5, and 7μ for both approaches, and run $\lambda = 9\mu$ for $(\mu + \lambda)$ only, primarily due to time constraints.

V. Results

A. Genetic Algorithms

1) Generational GA

The generational genetic algorithm was able to achieve a score of 0.0813 after 102 generations. Of interest is the clear distinction between the maximum and average fitness values

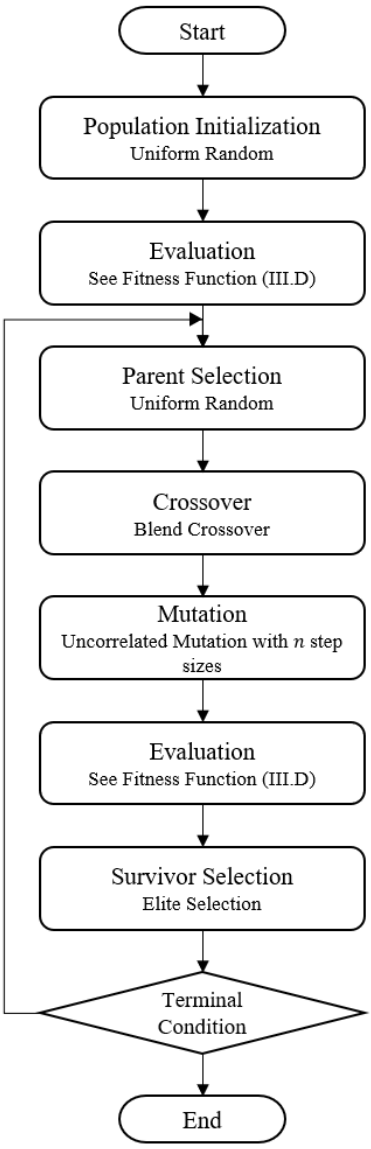

Fig. 3: A flowchart detailing the process of an evolutionary strategy approach. Both (μ, λ) and $(\mu + \lambda)$ can be described by this chart, differing only in the set from which survivor selection is performed.

throughout. It will be seen in our other methods that the average fitness often converged quickly with the maximum fitness and diversity in score rapidly decreased. This is not the case for the generational algorithm, which also produced some of the most sharp increases in fitness of our methods.

2) Steady State GA

The steady state algorithm was run for 535 epochs and achieved a best fitness of 0.0747. The algorithm was largely stagnant until reaching phase 2 of the fitness evaluations. The population quickly converged over 50 epochs to the maximum fitness of 0.037 . This behavior repeated in phase 3, with the algorithm failing to significantly improve the fitness beyond trivial solutions. At this point, the steady state algorithm

Generational GA

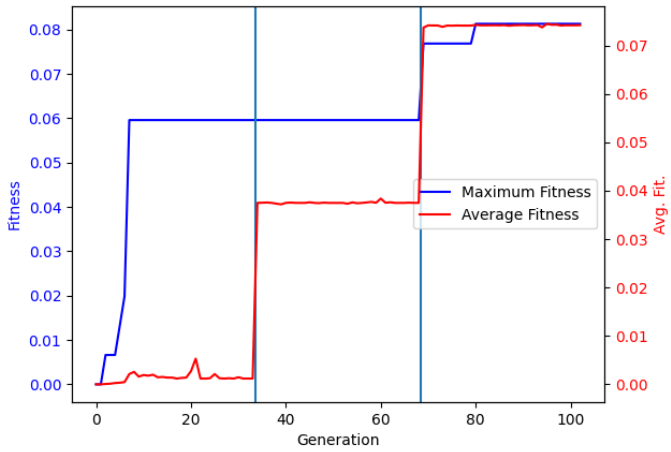

Fig. 4: A graph of the maximum and average fitnesses of the generational genetic algorithm. Notice the consistent difference between the maximum and the average fitness values.

Steady State GA

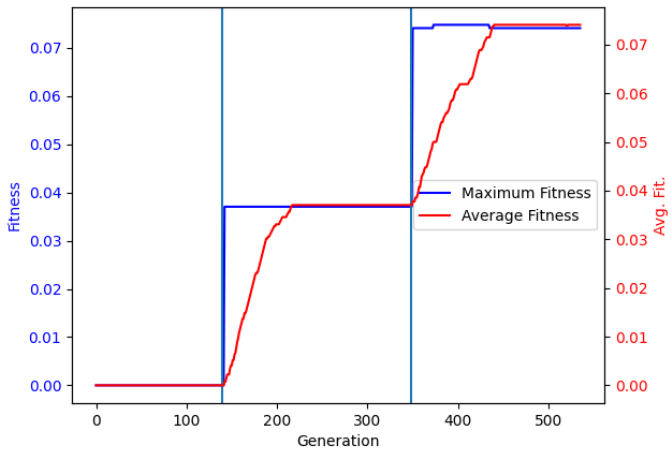

Fig. 5: A graph of the maximum and average fitnesses of the steady state genetic algorithm. This method provided the slowest convergence towards the populations maximum fitness of all the methods.

reached a perceived ceiling of 0.0747 fitness, the value for our trivial solutions. See Fig. 5 for the plot of the evolution.

B. Evolutionary Strategies

1) (μ, λ) Approach

Overall, the (μ, λ) approach has little success on this problem. See Figs. 6-8 to see plots of their evolution. While these methods do occasionally have success improving the maximum fitness of the population during a single phase of fitness evaluations, the inherent forgetfulness of this method rapidly loses track of successful individuals and falls back

(μ, λ) **where** $\lambda = 3\mu$

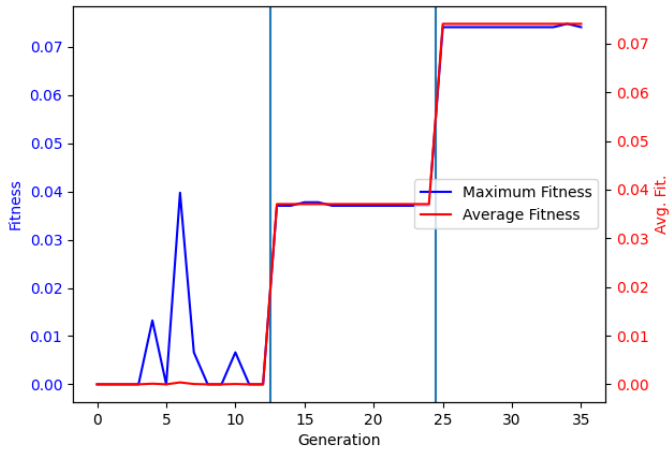

Fig. 6: A graph of the maximum and average fitnesses of the (μ, λ) evolutionary strategy for $\lambda = 3\mu$. This method was able to find early successful individuals, but due to the (μ, λ) approach, quickly lost these good individuals.

(μ, λ) **where** $\lambda = 5\mu$

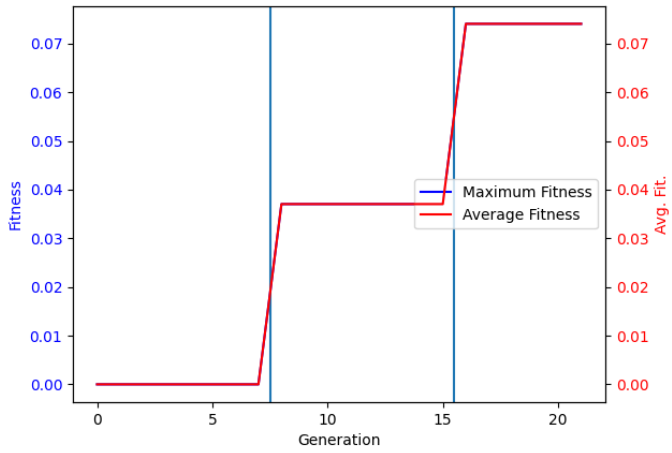

Fig. 7: A graph of the maximum and average fitnesses of the (μ, λ) evolutionary strategy for $\lambda = 5\mu$. This method had no success finding successful individuals outside of the easy optima that our fitness function was designed to avoid. Notice the nearly identical maximum and average fitness curves.

into the trivial solutions. It is interesting to note that the most success during phase 1 of evaluations was found with a low value for λ, suggesting that producing too many offspring from the population could ultimately be detrimental under this approach for optimizing alignment.

2) $(\mu + \lambda)$ Approach

The results of the $(\mu + \lambda)$ approach proved to be our most successful and are depicted in Figs. 9-12. The best fitness value found by this method was 0.0813 from $\lambda = 9\mu$, which is the

(μ, λ) **where** $\lambda = 7\mu$

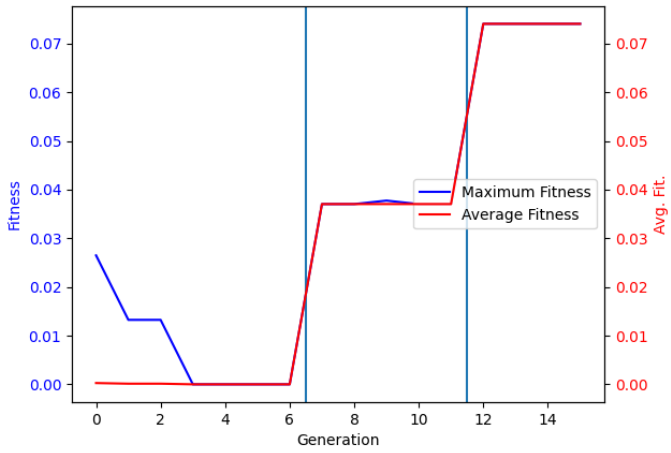

Fig. 8: A graph of the maximum and average fitnesses of the (μ, λ) evolutionary strategy for $\lambda = 7\mu$. This method had an early success similar to 6 with another brief increase in max fitness during phase 2, but also failed to maintain good individuals.

$(\mu + \lambda)$ **where** $\lambda = 3\mu$

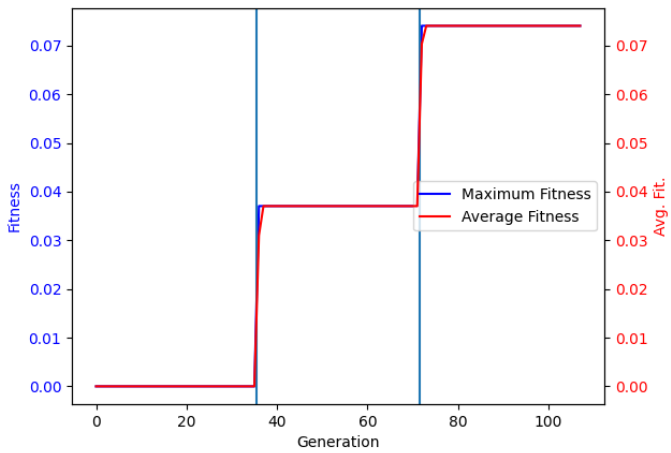

Fig. 9: A graph of the maximum and average fitnesses of the $(\mu + \lambda)$ evolutionary strategy for $\lambda = 3\mu$. This method had little success optimizing the boids. Note a slight lag in the average fitness converging to the population maximum fitness.

highest of all of our methods, and the second and third best were found by $\lambda = 5\mu$ and $\lambda = 7\mu$, respectively. The results showed that this method was in fact capable of improving the performance of individuals during both the initial and final phases of fitness evaluations and ultimately returning the individuals that most satisfied our fitness function..

$(\mu + \lambda)$ **where** $\lambda = 5\mu$

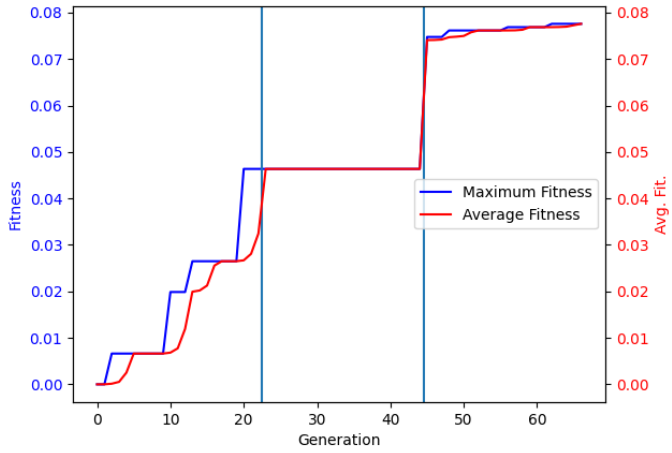

Fig. 10: A graph of the maximum and average fitnesses of the $(\mu + \lambda)$ evolutionary strategy for $\lambda = 5\mu$. This method was more capable of finding potentially fit individuals in the early phase of the evolution and acheived a higher overall fitness than $\lambda = 3\mu$.

$(\mu + \lambda)$ **where** $\lambda = 7\mu$

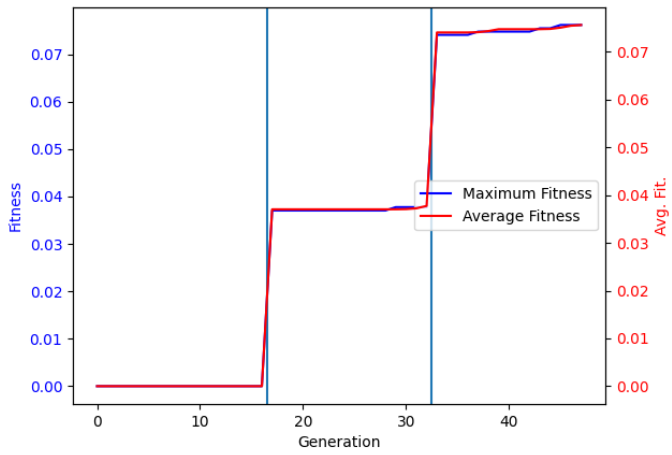

Fig. 11: A graph of the maximum and average fitnesses of the $(\mu + \lambda)$ evolutionary strategy for $\lambda = 7\mu$. This run seems to have been unsuccessful in finding less trivial fit individuals.

VI. Discussion

Overall, we find little success in optimizing boid behavior such that it expresses all three behaviors for which we have classifiers. Despite this lack of success, it is important to note that the fitness function is deceptively low, for it heavily punishes any difference in fitness for each of the three behaviors. Our algorithms often find individual species with a behavior-detailed fitness of [0.0, 1.0, 1.0], optizing $2/3$ of the behaviors, yet receiving a score of 0.0747. Our attempt to

$(\mu + \lambda)$ **where** $\lambda = 9\mu$

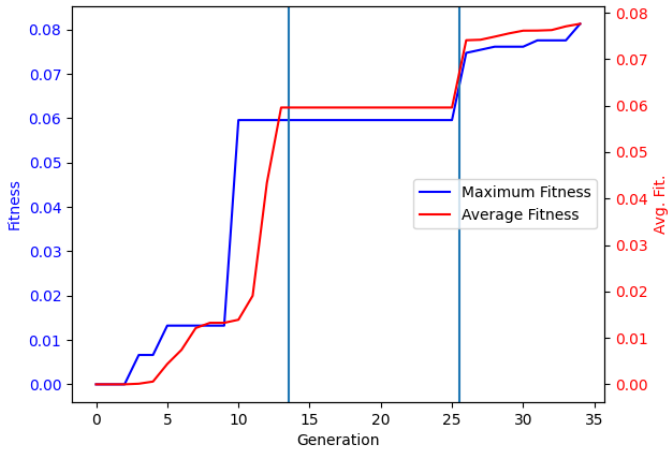

Fig. 12: A graph of the maximum and average fitnesses of the $(\mu + \lambda)$ evolutionary strategy for $\lambda = 9\mu$. This method performed the best, achieving the best fitness scores for each of the three phase of fitness evaluations.

modify the fitness function and dynamically change it based on the number of evaluations that have been completed was largely unsuccessful.

We do note that the $(\mu + \lambda)$ evolutionary strategy approach seems to have had the most success in developing promising individuals, as 3 of the top 4 individuals came from this method. One might initially expect a (μ, λ) approach to outperform given our dymanic fitness function, but the changes to the fitness function are relatively infrequent. Further, the ability to hold onto a good region of the search space when it has been found seems crucial for this problem. While the generational algorithm was able to achieve the same maximum score, the abrupt changes in fitness suggest that this score may have ultimately resulted from lucky population and mutations. The steady growth of the evolutionary strategy suggests a more reliable performance over many runs.

We note in Table I that many of the best individuals found in our evolutions have significantly different parameters for behavior. This suggests that the best regions of the search space that are being found are relatively flat and spread out. There are many possible points in the search-space with the same fitness, without much variability in the fitness values.

Score	Align.	Sep.	Coh.	A/C Rad.	Sep. Rad.	Max Accel.
0.081	1.58	0.477	2.00	107.5	107.5	5.00
0.081	1.61	0.353	2.00	154.0	145.0	5.00
0.078	1.18	0.228	1.93	211.4	211.4	4.72
0.076	0.966	0.410	1.70	300.0	218.9	3.47

TABLE I: The top four individuals found in our evolutions and their parameter values.

We also note that, while our best individuals were those that most satisfied the classifiers and consequently our fitness function, they do not accurately represent flocking behavior

in a qualitative way. See Figs. 13-15 for a sample of swarms using these behaviors in midflight. One would be hard pressed to identify these species as flocking, especially when compared to a boid species designed by hand, as depicted in Fig. 16.

1st Place Species

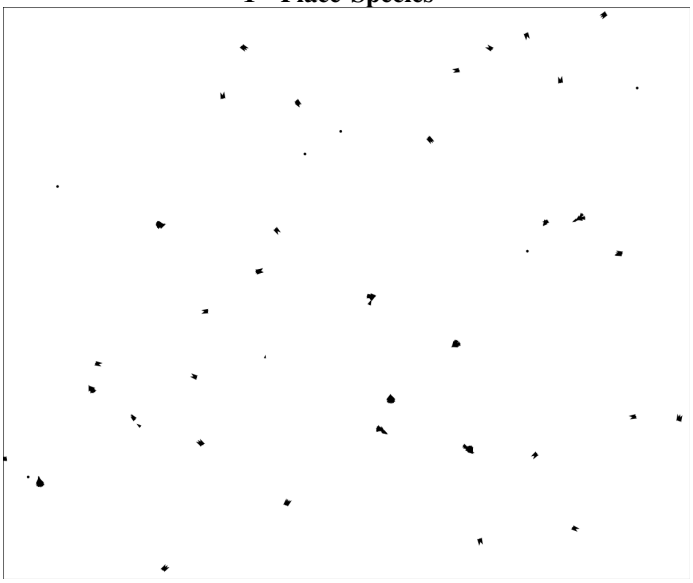

Fig. 13: A mid-flight sample of our best individual. Notice a heavy clustering of the 200 boids into very tight groups.

2nd Place Species

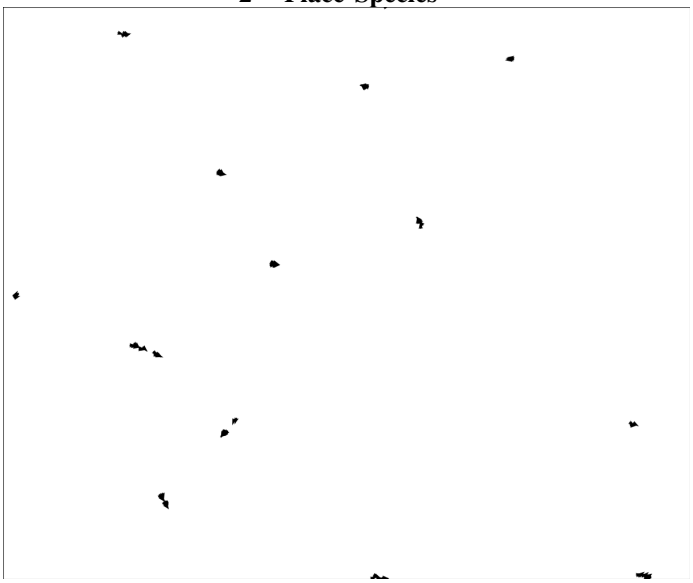

Fig. 14: A mid-flight sample of our second best individual. Notice again heavy clustering, along with a tendency for small groups to fly in tight circles.

3rd Place Species

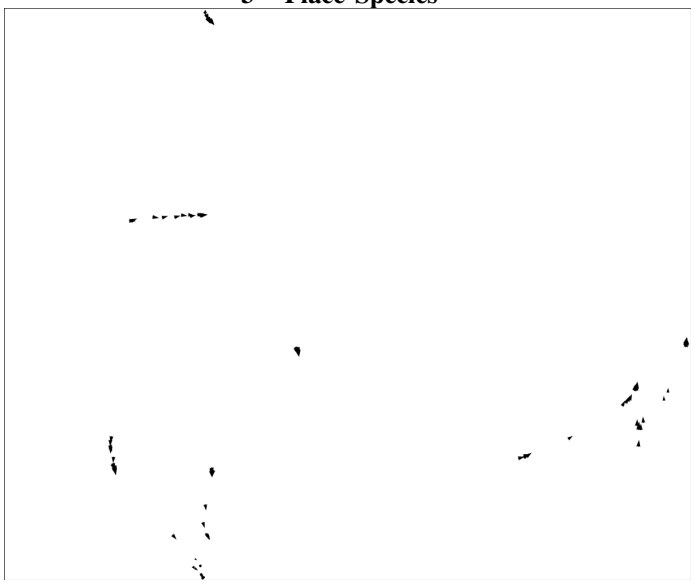

Fig. 15: A mid-flight sample of our third best individual. Similar heavy clustering with a single-file tendency.

A Custom Species

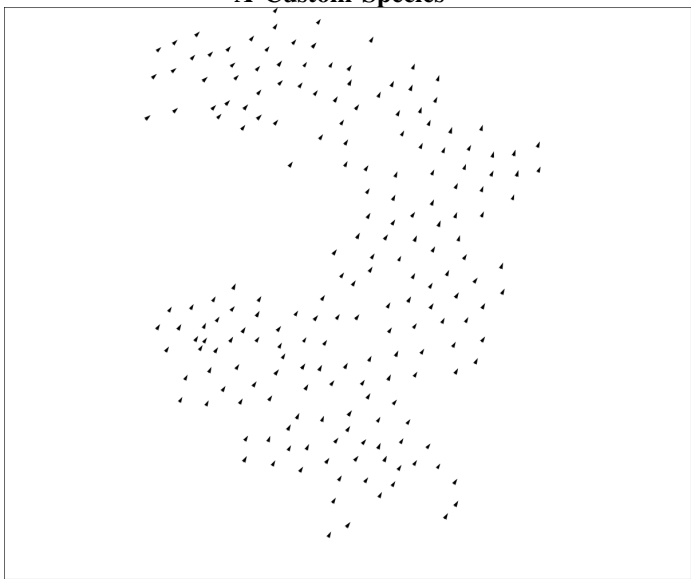

Fig. 16: A mid-flight sample of a species quickly built to flock. This significantly outperforms our evolved species from a qualitative standpoint.

The above comments combined with our lack of success with a range of methods and the overall difficulty of this problem suggests two possibilities to the authors.

The first is that the region in the space of potential boid species in which a swarm will satisfy the alignment classifier very small and perhaps deceptively structured or scattered. This would explain the incredible difficulty that the algorithms seem to have not only finding species which score well with that classifier, but also finding descendents of these species that better satisfy the classifier. The regions of boid-space which do not satisfy the flocking and grouped classifiers seem to have a similar structure, making the optimization of these behaviors relatively trivial.

The other suggestion is that there are issues with the

classifiers and their ability to accurately classify boid behavior. There are significant concerns noted by Liu et al. regarding the methodology with which the training data was generated, as well as concerns of generalizability throughout the training process. In part, our work sought to confirm the validity of their classifiers on an unseen collection of boid behaviors, and it seems that their attempts to remedy issues of generalizability failed. Many of the species explored by our algortihms do well expressing the behaviors of concern to human eyes, yet the classifier scores them poorly. Alternatively, as depicted in Figs. 13-15, some of the most successful species found behave in ways that do not reflect natural flocking behavior, such as heavily clustering, flying in circles, and jittering back and forth. This suggests that the classifiers do exploit attributes of the original data set for their models, and the evolution has learned to exploit these models in turn. Perhaps this work could be attempted again following further development of their classifiers as suggested by those authors [7].

It is also interesting to note that the species that are evolved by our algorithms most often disregard success based upon the alignment algorithm. This seems to echo lessons learned from Ramos et al.'s study which found that the alignment metric was ultimately unnecessary to produce qualitative flocking behavior [2]. Perhaps this insight suggests that success with the alignment classifier is ultimately unnecessary for the underlying goal of evolving realistic boid flocking.

VII. Conclusion

After developing a fitness function based on pre-trained classifiers in order to score species of boids based on their ability to express the behaviors 'Alignment', 'Flocking', and 'Grouped,' we find that evolutionary methods seem to simply exploit overfitting issues within the classifiers and do so in a way that does not easily reach an optimal individual. The formulation of this problem, particularly the reliance upon classifiers for which generalizability is already a concern, make this an incredibly difficult problem. That said, we note that the $(\mu+\lambda)$ evolutionary strategy approach seems to be the best method for the problem, due to its ability to remember successful individuals that have been found in previous generations.

Moving forward, this project would best be revisited following imporovements made to the classifiers as discussed in Liu et al. as discussed in their work from which the classifers were taken. It seems that the issues of overfitting on the original training data and the issues inherent to those data create very large and flat local optima or plateaus in the fitness function, which the evolutions struggle to reach past.

Given a more appropriate classifier, it would also be worth running this experiment multiple times. Given time constraints and the complexity of the fitness function, these results were produced from a single run of each method. Considering the stochastic nature of evolutionary algorithms and their success, any definite claims about performance would require a more comprehensive and statistical analysis of the methods.

It should also be noted that the goal of this problem, evolving optimal boid parameters to express flocking, inherently imposes a rather slow method onto the creation of a species of boid which successfully flocks. For practical purposes, it is much more cost and time effective to craft boid parameters by hand to satify the flocking requirements. That said, success in this problem could be enlightening to further study of swarm behaviors and the application of evolutionary algorithms to difficult problems.

References

[1] C. W. Reynolds, "Flocks, herds, and schools: A distributed behavioral model," 1987.

[2] R. P. Ramos, S. M. Oliveira, S. M. Vieira, and A. L. Christensen, "Evolving flocking in embodied agents based on local and global application of reynolds' rules.," *PLoS ONE*, vol. 14, no. 10, 2019.

[3] O. Witkowski and T. Ikegami, "Emergence of swarming behavior: Foraging agents evolve collective motion based on signaling.," *PLoS ONE*, vol. 11, no. 4, pp. 1 – 26, 2016.

[4] I. L. Bajec, N. Zimic, and M. Mraz, "The computational beauty of flocking: boids revisited," *Mathematical and Computer Modelling of Dynamical Systems*, vol. 13, no. 4, pp. 331–347, 2007.

[5] N. P. Rougier, *From Python to Numpy: Version 1.1*. Zenodo, Dec. 2016.

[6] M. Chalela, E. Sillero, L. Pereyra, M. A. García, J. B. Cabral, M. Lares, and M. Merchán, "Grispy: A python package for fixed-radius nearest neighbors search," *arXiv preprint arXiv:1912.09585*, 2019.

[7] Z. Liu, R. McArdle, and G. Orlando, "Classifying the behaviors of boid swarms," *Unpublished; available upon request*, 2020.

[8] D. Dua and C. Graff, "UCI machine learning repository," 2017.

[9] S.Abpeikar, K. Kasmarik, M. Barlow, and M. Khan, "Human perception of swarming: Swarm behaviour data set," 2020.

[10] S.Abpeikar, K. Kasmarik, M. Barlow, and M. Khan, "Human perception of swarming," 2020.