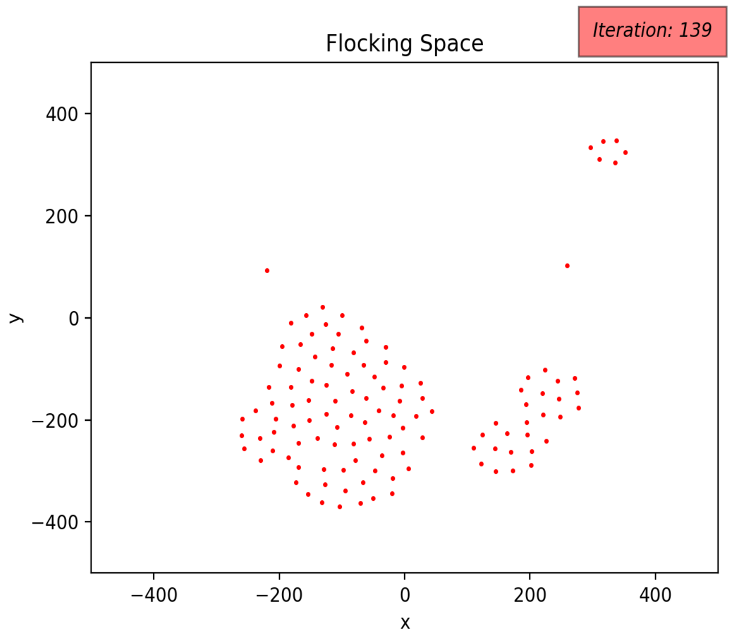
Particle Swarm Optimization with Flocking and Genetic Programming

Seth Chatterton, Class of 2019

The goal of this project was to find a more effective and efficient particle swarm optimization algorithm (PSO). PSO is a widely used optimization algorithm used to find the optimum of multidimensional functions. For example, if we wanted to find the lowest point in a landscape, PSO could help us find it. PSO works by sending out multiple particles that fly across the function space, which in our example is the landscape, and measure the function value at that point, in this case the height of the landscape. Each particle then looks at what function value its neighbors have, and adjusts its velocity to head towards its neighbors if the neighbors have found a better spot. Much of PSO research is determining what a particle's neighborhood should be. In our research, we try to create good neighborhoods for our PSO particles using flocking behavior.

Flocking in nature is determined mostly through three parameters: alignment, cohesion, and separation. Alignment means that all of the creatures in a flock tend to align their direction with one another, cohesion means they tend to move toward the average position of the group, and separation means they tend to not get too close to their neighbors. These three parameters create flocks of particles within a separate flocking space. Particles within a certain radius of one another are added to that particle's neighborhood, which is then used as the neighborhood in the PSO function space. The next question is then how we determine what values of flocking parameters we use for our flocking space. For this, we used genetic programming. Essentially, our genetic program mimics evolution in nature. Our genetic program creates a population of random programs that set flocking parameters. These programs are then evaluated on how good the neighborhoods they produce through flocking behavior are. The best programs are saved for the next generation, are mixed with other good programs to create new programs, and occasionally mutated. This hopefully produces a program that can dynamically set flocking parameters to create very good neighborhoods for PSO.

To test the algorithm, I used Bowdoin's High Performance Computing Grid, which allowed me to test dozens of settings at a time by running tests in parallel. Parameters that I tested were the size of the genetic program, the crossover rate (how often to genetic programs get mixed together), the genetic program mutation rate, and the function to optimize. The size of the population of genetic programs was typically 100 individuals, and the programs were evolved for between 100 and 300 generations. Each program was run 50 times to determine how good the program was on average. Unfortunately, over the course of this project the algorithm never evolved anything better than the standard PSO algorithm. This could be due to several factors, including not having a large enough population, not having enough generations of evolution, or bugs in the code.



Left: An example of flocking behavior exhibited by this algorithm. The red particles form flocks, and nearby particles become neighbors in the PSO algorithm.

Faculty Mentor: Stephen Majercik
Funded by the Maine Space Grant Consortium
Previous work on this project by Prof. Stephen Majercik and Sophie Ardell, '18