Exploring Genetic Algorithms Through the Iterative Prisoner's Dilemma

Iterative Prisoner's Dilemma

The general Prisoner's Dilemma is a scenario in which there are two players that can each choose to either cooperate with the other or to defect. They must each make their decisions without knowledge of the other's decision. Each player then gets points based on what their decisions were. The points are given as follows:

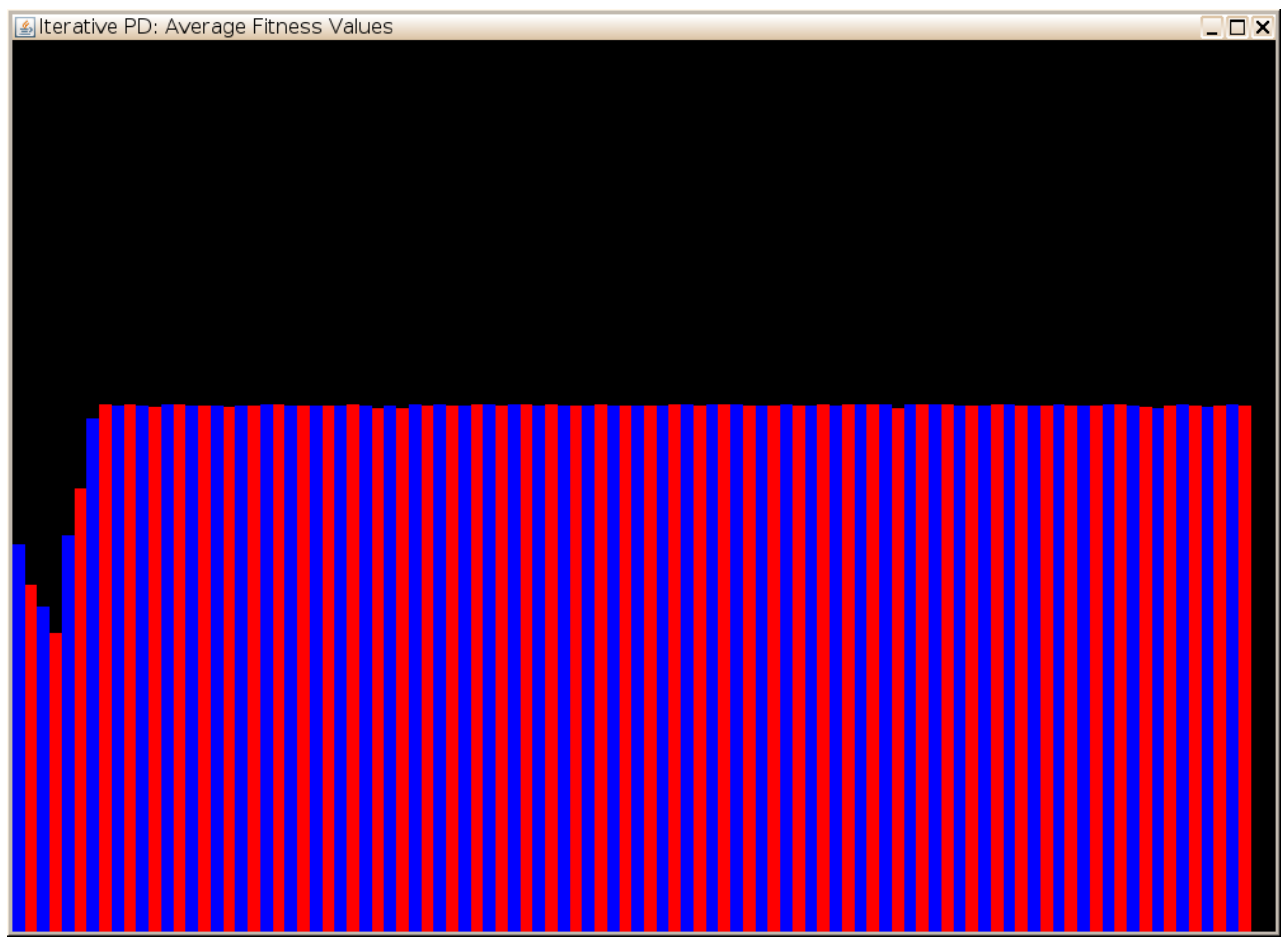
	Cooperate	Defect
Cooperate	R,R	S,T
Defect	T,S	P,P

The point values must satisfy certain inequalities. In order to be a Prisoner's Dilemma problem, the inequality $R > T > P > S$ must be satisfied.

In the Iterative Prisoner's Dilemma, the two players go through this scenario many times, and remember the past. In order to be used for the Iterative Prisoner's Dilemma, the values must satisfy the inequality $2R > S + T$. The points from each round are added to form a score for each player. I used the following table, which satisfies both inequalities and is the most commonly used set of values:

	Cooperate	Defect
Cooperate	3,3	0,5
Defect	5,0	1,1

The output of my program is a graph of the average fitness value for each generation (shown below) as well as the numbers represented by this graph in a text file, so that the graph can be recreated after the program is closed. I will use these graphs to determine how many generations the algorithm took to reach the best solution, and compare these among different algorithms.

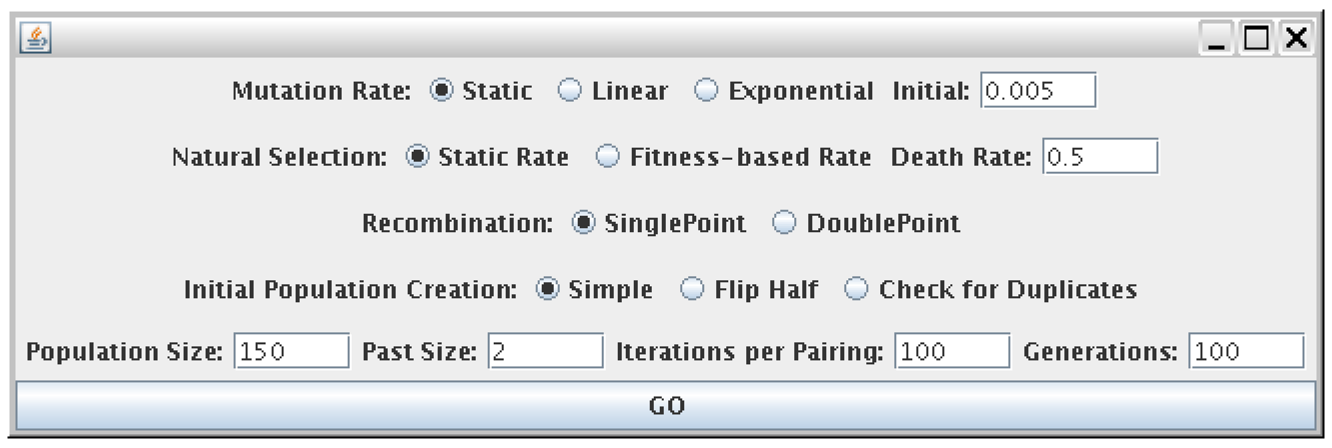


Genetic Algorithm

Genetic algorithms are used to find approximate solutions to optimization problems when the solution would be very time-consuming to compute. The general layout of a genetic algorithm is:

```
Initialization of gene pool
Loop over generations
    Natural Selection
    Selection
    Loop over empty slots in population
        Recombination
        Mutation
```

There are many ways in which to perform each of these steps. My program allows the user to select which method to use for each step. In this way, the different methods can be compared.



Mutation Rate: [✓ Static] Linear Exponential Initial: 0.005

Natural Selection: [✓ Static Rate] Fitness-based Rate Death Rate: 0.5

Recombination: [✓ SinglePoint] DoublePoint

Initial Population Creation: [✓ Simple] Flip Half Check for Duplicates

Population Size: 150 Past Size: 2 Iterations per Pairing: 100 Generations: 100