Matthew Daniel

CS 415

February 9, 2017

Sub Project 1a Write-Up

**Summary:**

To alter the genetic algorithm from class, I plan to use a different fitness function from the list provided. The Schwefel function that was briefly discussed in class seems promising. Additionally, I will vary mutation rates while holding the population size and number of iterations constant. I also want to increase the complexity of the crossover function. This may look something like picking random (or predetermined) sections of the DNA to exchange between the organisms passed into the crossover function.

An example graph of the data may look something like:

Matthew Daniel

CS 415

February 28, 2017

Final Project 1 Write-Up

**Abstract:**

I modified the genetic algorithm from class to measure a creature’s fitness using the Scwefel Function, to vary mutation rates, and to vary crossover. Using the same randomly generated population of size 20 and trait length of 10, I calculated the average fitness of the population after 40 iterations of the algorithm. I did this for every mutation rate/crossover combination, then compared how the combination ranked against the other mutation rates and crossover combinations. I ran 10 such trials, then took the average of the ranks for each of the variables as a measure of how well each mutation rate and crossover pattern created fit populations.

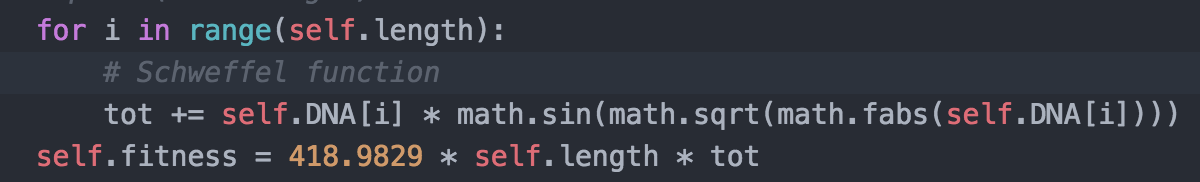
**Algorithm Descriptions:**

**Population initialization:**

I initialized the 10 trait values to a random float between (-10,10) for each of the 20 creatures in the population (same as class algorithm). I stored this initial population outside of the population class so that I could use the same randomly generated population values for each of the mutation rate/crossover combinations.

**Measure of fitness:**

I used the Scwefel Function from the list of provided fitness functions to measure the fitness of a creature. I calculated the fitness using the python math library:



**Mutation rates:**