CS 480: Assignment 06 Report
Zachary Ferguson

Genetic Algorithms

Design and Implementation:

Implementing the functions for this assignment was a simple conversion from the algorithms provided in the lecture. Utilizing the provided helper functions, I was able to simplify the implementation in many cases. The only function pseudo-code not provided was the evolve function. This function, however, is a straight forward loop creating new generations each iteration.

First, I setup any parameters necessary for the problem, then I create the first generation with the given creator function. I then loop *generation* times, crossing-over and mutating a selected subset of the previous generation, repeating the above process with the new generation. Depending on the population size and number of generations the problem can take a significant time to solve.

Overall the implementation of the functions was straight forward and easy to implement in a C-like manor. Some thought was required when implementing in a more standard LISP style, but not more than any previous assignment.

Reflection:

This assignment helped to cement the ideas behind genetic algorithms and there usefulness. The provided evaluation functions were helpful when testing and exemplified the ability of genetic algorithms.

Analyzing the Results:

Using the provided evaluation functions, I was able to test my implementation. This testing involved varying the parameters of the functions in the form of the tournament size, mutation rate, crossover rate, and mutation variance. The population size is held constant at 50 individuals. The size of the floating point vector is 20, and the number of generations is 1000.

For the first test function, sum, the fitness is exactly equal to the sum of the genes of the individual. All parameters were left at their default values. The average best fitness of 50 trials was equal to 101.37. With a max fitness of 5.12 * 20 = 102.4, the percent error of the results is 1.01%. This is well within reason.

For the next test, step, the fitness is equal to the sum of the floor of each gene plus six times the length of the vector. Therefore the max fitness achievable is 20 * 5 + 20 * 6 = 220. As above the default parameters were used and the resulting average best fitness was 214.36. The percent error of these results is 2.56% well within reason.

The sphere test function minimizes the squared sum of the vectors values. Therefore vectors closer to zero have a better fitness. In experimentation, using a tournament size of 14 resulted in an average best fitness of -0.198. For this test fitnesses range from -524.288 to 0.0, so the resulting value is well within expectation.

The other tests followed a similar vein, resulting in low percent error and good fitnesses. The following table illustrates the results of all experiments: