# COMP SCI 5401 FS2017 Assignment 1c

Stuart Miller
sm67c@mst.edu

October 7, 2017

## 1 Overview

Assignment 1c presented a great many parameters to test. As more techniques are introduced, it becomes more and more difficult to distinguish a genuinely "'good"' parameter value as one can never be entirely sure if it was solely one value that resulted in a change in fitness landscape, a combination of multiple values, or perhaps the order of calls to the random number generator came out more favorably. Therefore, in this assignment, it has been attempted to isolate parameters being tested and pick favorable and reasonably values to the parameters not immediately relevant.

As a precursor, the entirety of this assignment will make use of n-point crossovers and random reset mutations. Although other operators are implemented in the codebase, these two have been set to be self-adaptive in Bonus 2. For consistency with the runs in the Bonus section, even when self-adaptability is turned off, the analysis will still use only these two operators. For initial values and non adaptive states, the operators will be fixed at three crossover points and a 12% mutation chance. These are values that were determined to be beneficial in Assignment 1b. Furthermore, survival strategy will always be set to comma $(\mu, \lambda)$ and selection will be k-tournaments with moderate selective pressure.

Furthermore, box plots were generated for each test run graphed below. As this totals to a rather large number of boxplots, they have not been pictured in this report. Instead, they can be viewed in ./doc/images/assn1c_boxplot_*.png.

## 2 Penalty Coefficient

To test the penalty coefficient, two control sets were run where invalid placements were solved by random replacements and then a repair. The penalty function was then implemented, with static weights of 1.0, 2.0, and 5.0, respectively.

As the graph in Figures 1, 2, and 3 all show, the penalty weight functions shows significantly lower fitness values than other methods. This is to be expected for intermediate solutions, but the best solution shouldn't have any penalty. Additionally, the fact the the different weights returned the same max fitness shows that this may not be an optimal strategy for this problem. It is worth noting that numerous evolutionary strategy were tried and this combination yielded the best and most consistent results.

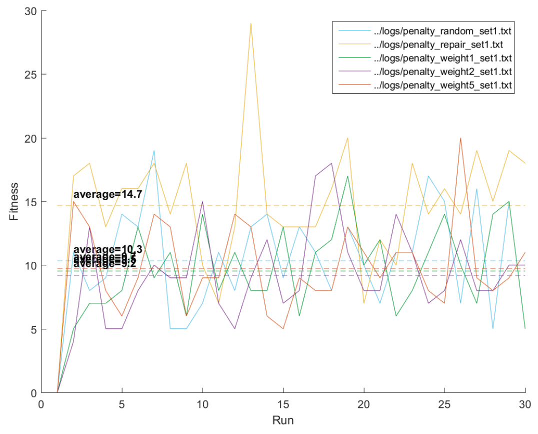

Figure 1: Penalty Weights, 50Shapes.txt

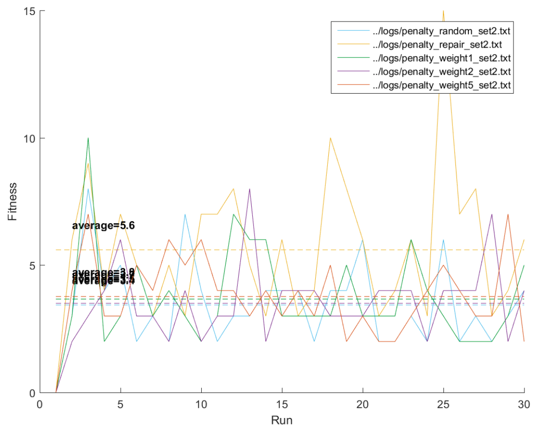

Figure 2: Penalty Weights, 100Shapes.txt

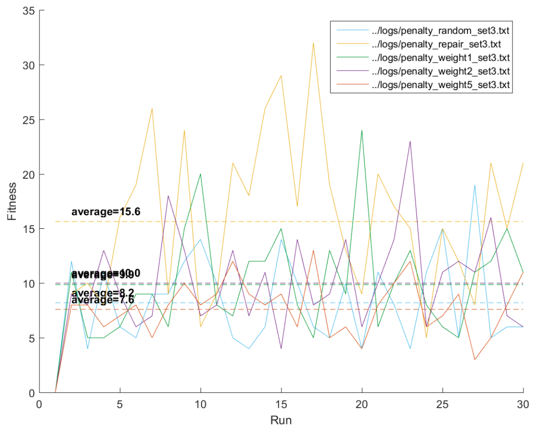

Figure 3: Penalty Weights, 100ShapesComplex.txt

## 3 Self-Adaptive Parameter

To test self-adaptability, the number of crossover points $n$ was set to be adaptable. For offspring created, there was an equivalent random chance of inheriting each parent's $n$. Additionally, $n$ had a small chance to mutate (equivalent to the genetic mutation rate) by $\pm 1$. For analysis, the mutation rate was averaged across all member of the population for each generation.

As it was determined in Section 2, the penalty function yields no great benefit for this problem, so the repair function method was used here instead. The other parameters were kept the same as in section 2.

As figures 4, 5, and 6 all show, allowing crossovers to self-adapt has a significant improvement on the average best fitness. It is also interesting to worth noting that while each instance started at three crossover points, some runs took this number higher, and some drove it lower. Examining the log (located in the ./logs/ directory) shows no consistent trend. It can only be assumed that each run's random initialization took it to a different area of the fitness landscape where more or less crossovers were desirable.

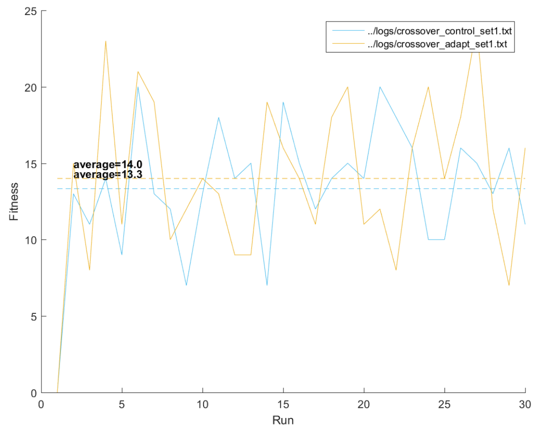

Figure 4: Adaptive Crossovers, 50Shapes.txt

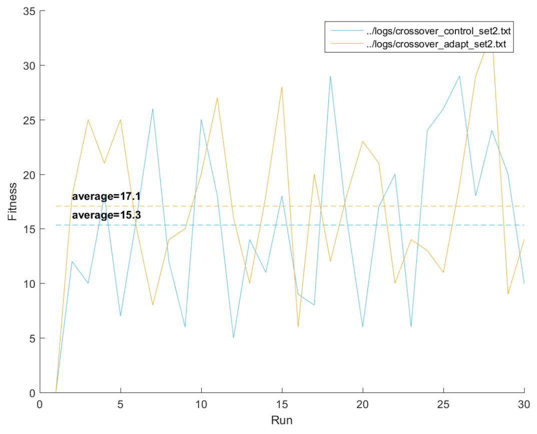

Figure 5: Adaptive Crossovers, 100Shapes.txt

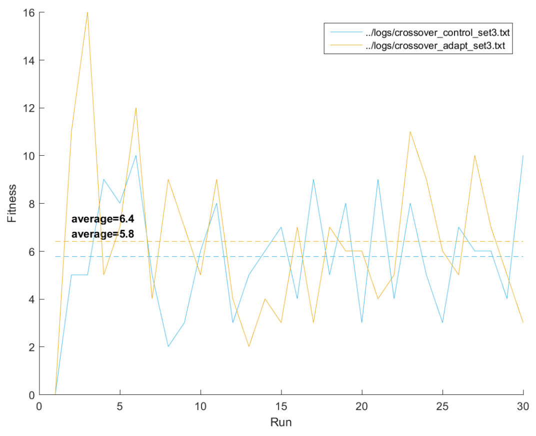

Figure 6: Adaptive Crossovers, 100ShapesComplex.txt

## 4 Bonus 1

## 5 Bonus 2