CPSC 371 Vincent Tennant
Artificial Intelligence 230099844
Phase 3

The Program

There are two main components to this phase, the Genetic Algorithm and the Genome container object. Starting with the Genetic Algorithm class, this forms the bulk of phase 3. Intended to be an implementation of a genetic algorithm, it starts by creating a random population of random genomes. From this initial population the desired number of elite genomes are chosen from this beginning population. Elite genomes are chosen based on their raw fitness and diversity scores. Raw fitness being how well that individual performs on the given problem, and diversity score being how different the genome is to all other elite genomes. Once chosen the remainder of next generations population is mad up of mutated elite genomes, and crossover genomes. For mutated genomes a random elite genome is chosen. Once chosen the values within the elite genome are randomly perturbed to anywhere between +25% and -25% of the original value. This mutation process is repeated until the desired number of mutated genomes per population is reached. Once all of the mutated genomes have been selected the crossover genome selection process begins. Crossover genomes are a combination of two different elite genomes. A mapping array is created to determine which genome parameter from which elite parent genomes goes in each position of the child genome. Once the elite genomes have been chosen, the mutated and crossover genomes created, then all three sets are combined to form the new population for the next generation. The process is repeated, selecting new elite genomes, creating new mutated genomes and crossover genomes, until the desired number of generations has been reached.

The Genome class if a very simple, and straight forward container object. Their genome class hold the information required to solve a given problem. That information could be a string, integers, or floating points and double values. An important characteristic to note is that the solution within any given genome may

or may not be an optimal solution. In fact it may not solve the problem at all. This is where the genetic algorithm come into play. The genetic algorithm will use the genome class to create, and manipulate populations. Evolving them to the point where a genome containing an optimal solution can easily be found.

There are also a number of smaller classes in this phase. Three comparators used to order the population of genomes based on either raw fitness, diversity score, or combined rank. All of which are used throughout the genetic algorithm.

XOR Results

For the XOR problem the results proved promising. With a population size of 200 and 500 generations I was able to achieve to a very high degree of accuracy a solution to the XOR problem using the genetic algorithm. The top five parameter combinations in descending order for the genetic algorithm are as follows:

(Number of Generations) (Population Size) (Elite %) (Mutated %) (Crossover %)

(1) 500 200 10 90 0

(2) 350 450 20 60 20

(3) 500 440 20 50 30

(4) 300 440 20 60 20

(5) 500 400 20 60 20

Problems Encountered

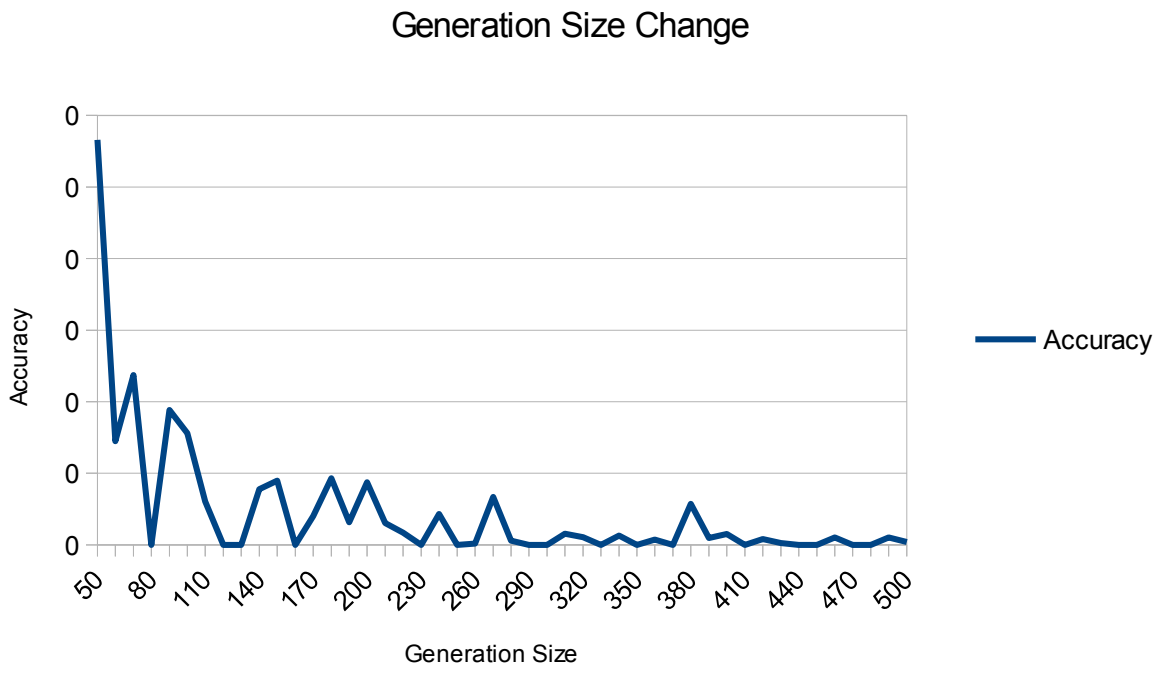
Initially I was creating the first population incorrectly. Instead of creating a population of genomes with random value between 1 and -1, I was creating genomes with only random values between 1 and 0. This led to some strange results. Though after some careful debugging was solved. During the experiments and creating of the filed used to graph the top five genomes I ran into an issue where given certain values the genetic algorithm didn't evolve at all. I was unable to locate the problem before hand-in. As a quick band-aid fix in order to make the graphs clean and readable the accuracy values of the un-evolved genomes were changed to 0.

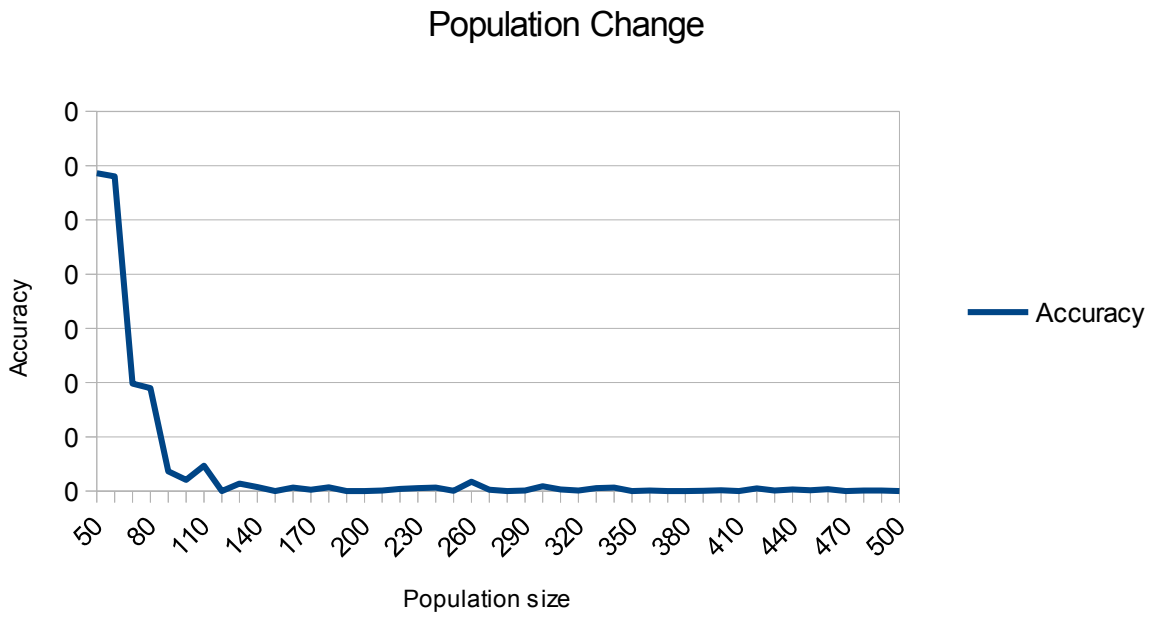
Graphs

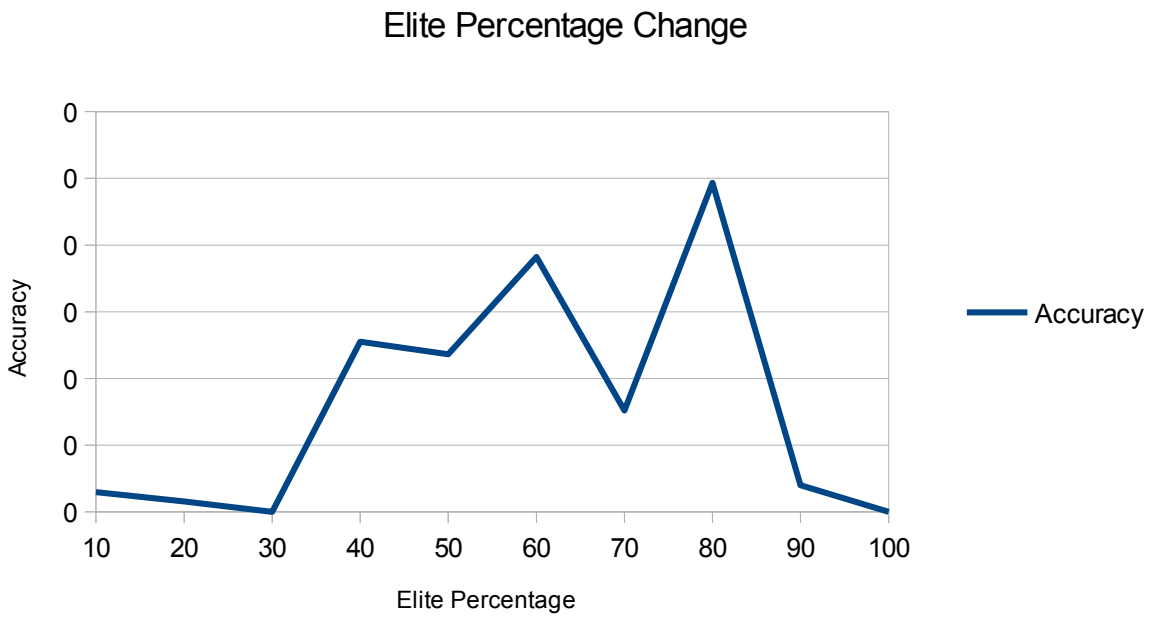
Lower the accuracy the closer to solving the problem.

Note that the y-axis only showing zeroes is a formatting issue in Libre Office Writer. They are in fact very small values.

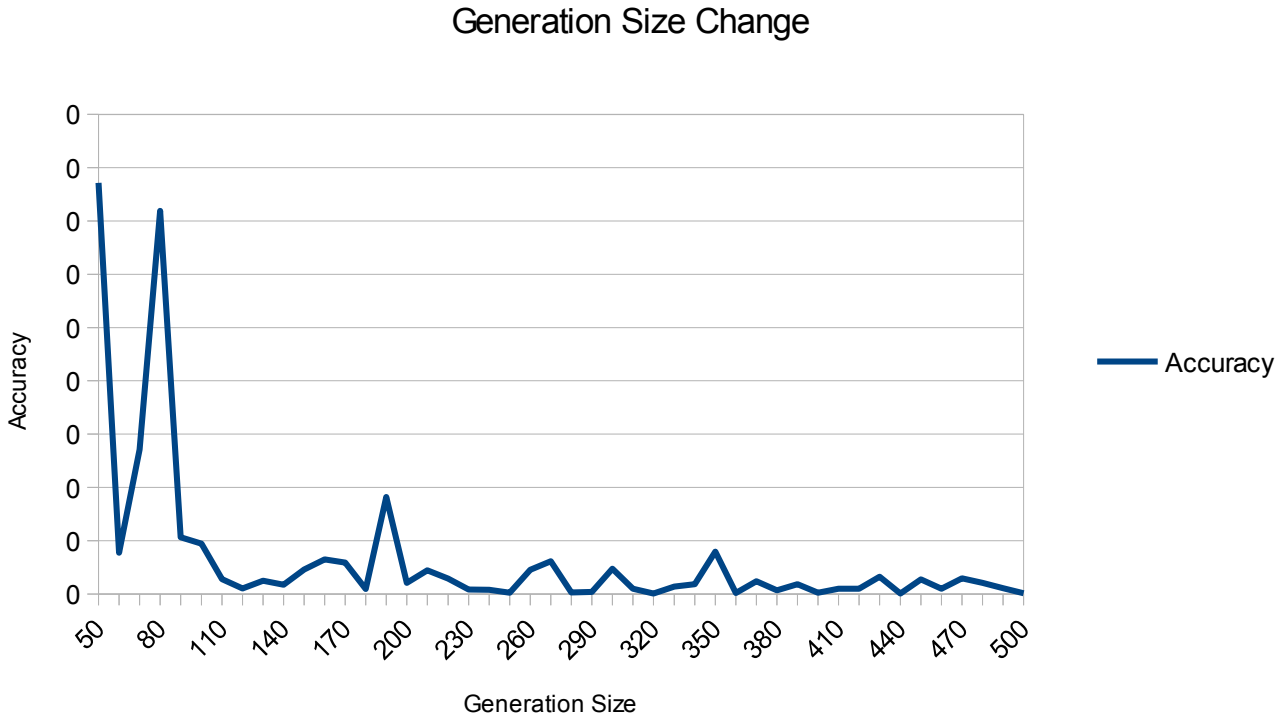
(1)

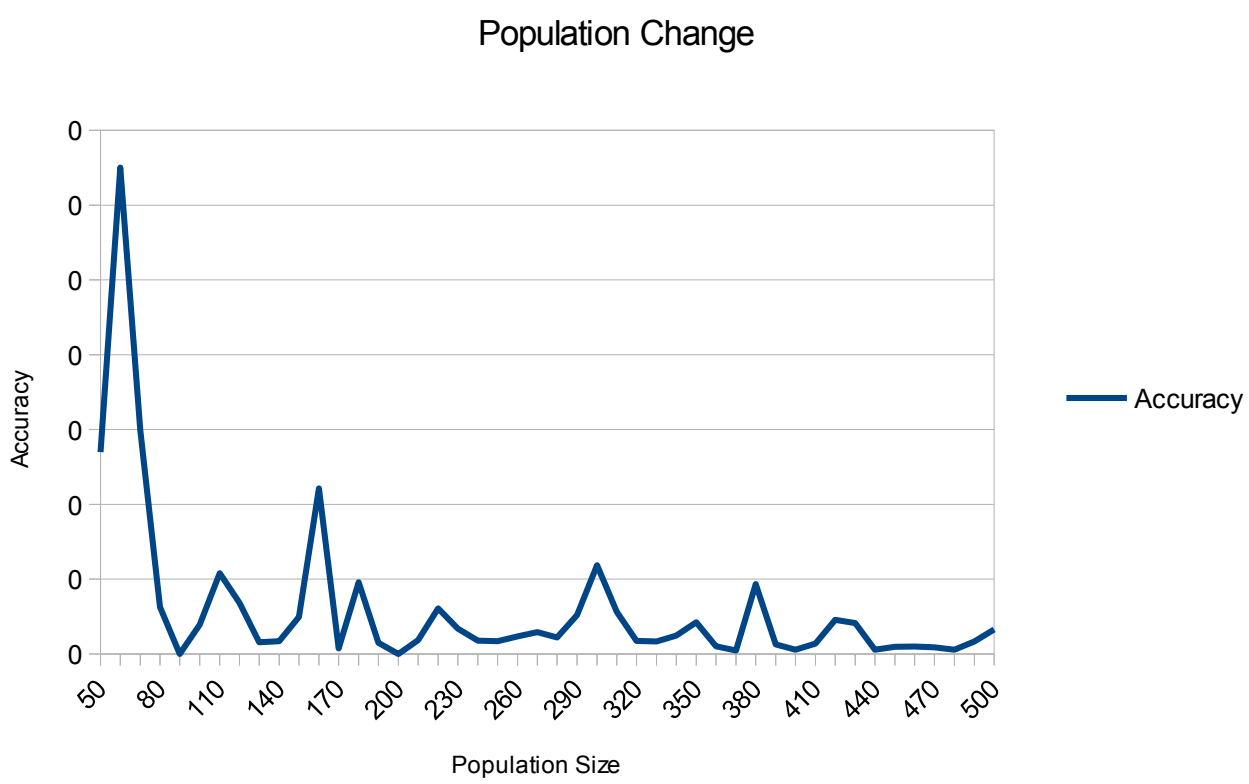


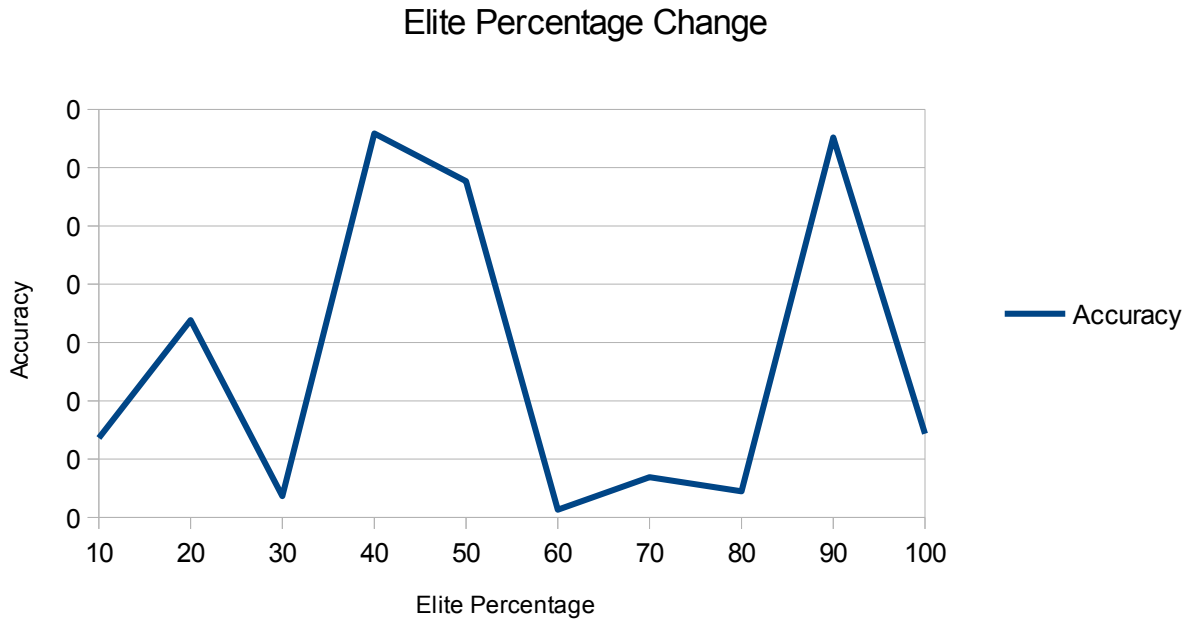



Elite Percentage Change
Accuracy
0
0
0
0
0
0
0
10
20
30
40
50
60
70
80
90
100
Elite Percentage
Accuracy

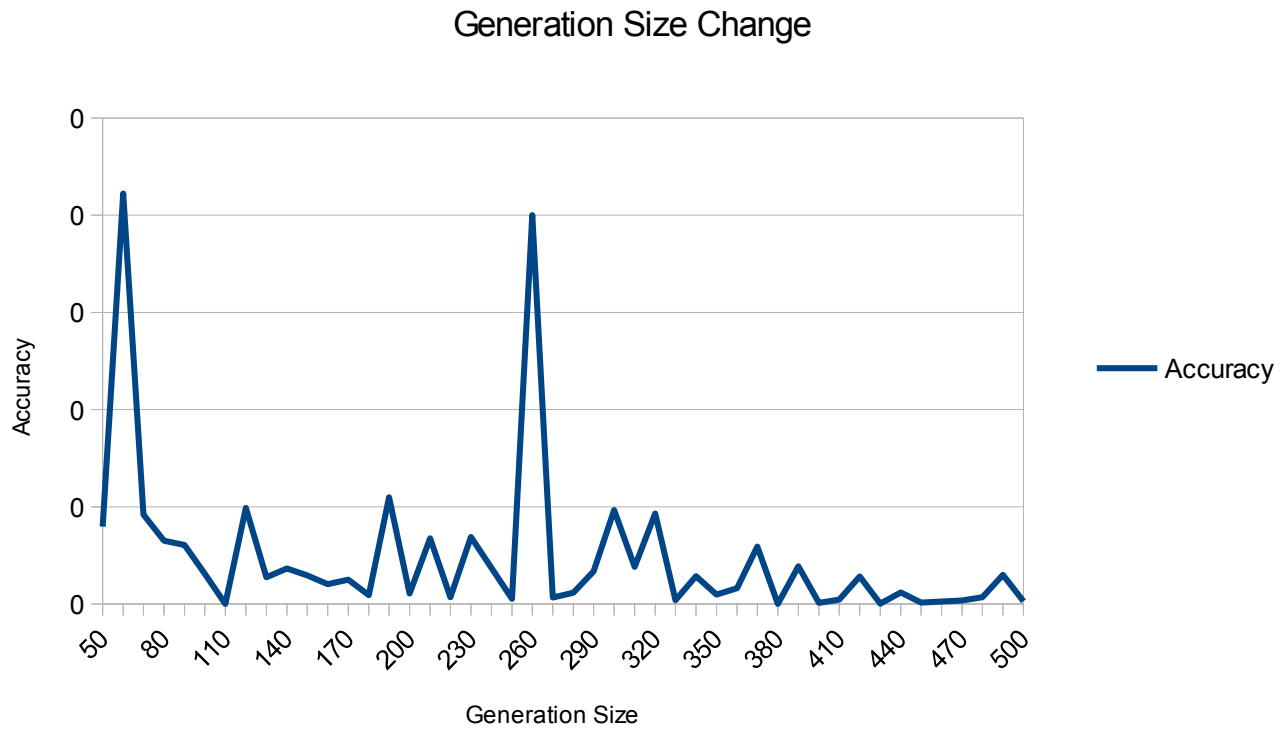
(2)

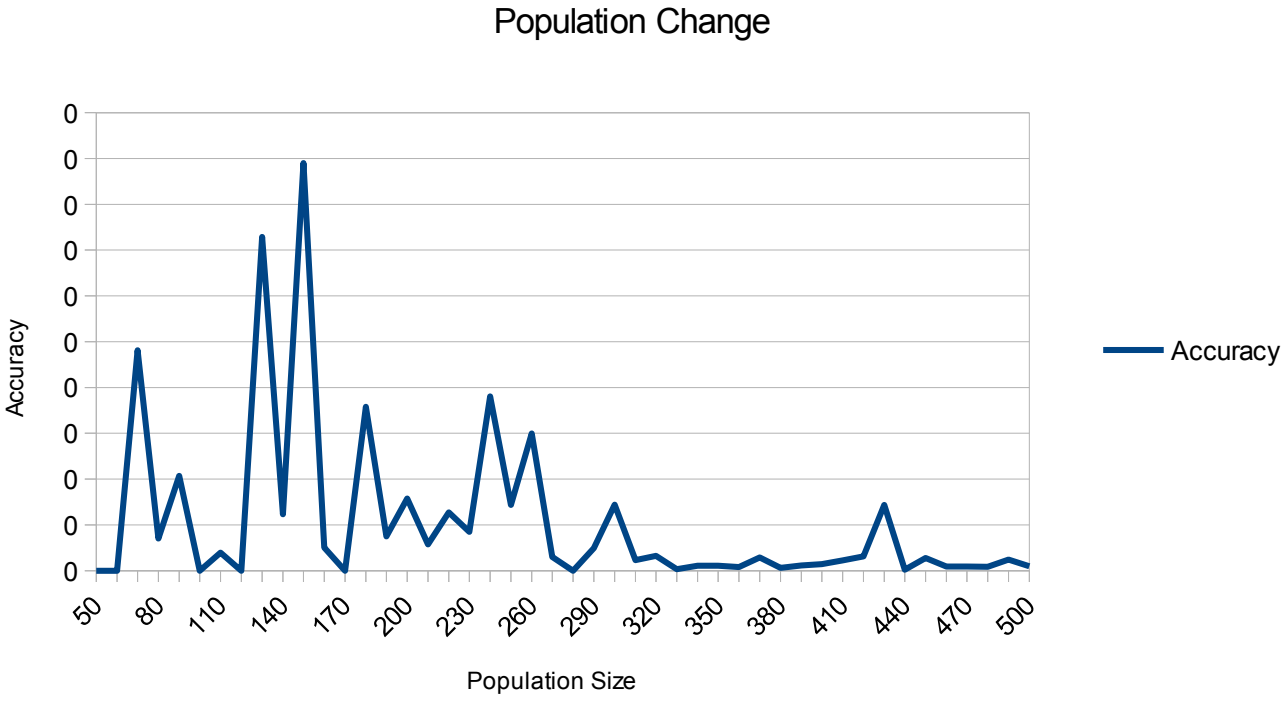

Generation Size Change
Accuracy
0
0
0
0
0
0
0
0
0
50
80
110
140
170
200
230
260
290
320
350
380
410
440
470
500
Generation Size
Accuracy


Population Change
Accuracy
0
0
0
0
0
0
0
0
50
80
110
140
170
200
230
260
290
320
350
380
410
440
470
500
Population Size
Accuracy

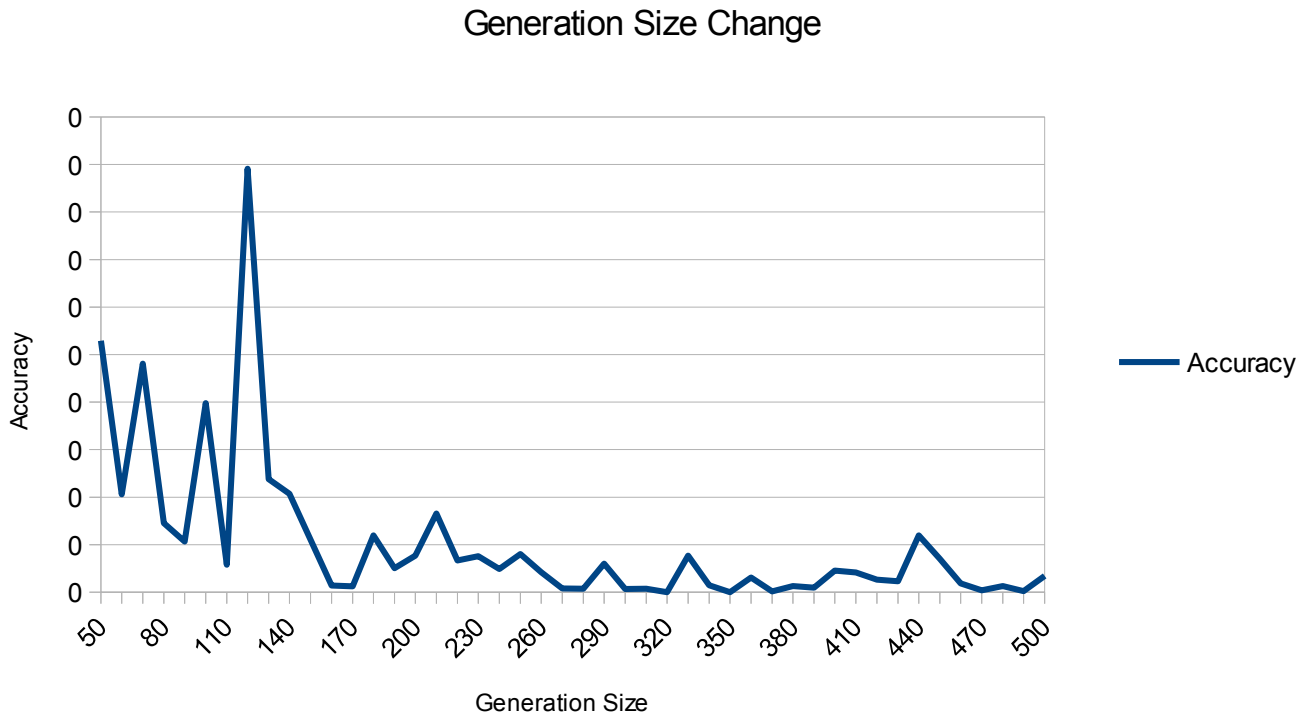

Elite Percentage Change
Accuracy
0
0
0
0
0
0
0
10
20
30
40
50
60
70
80
90
100
Elite Percentage
Accuracy

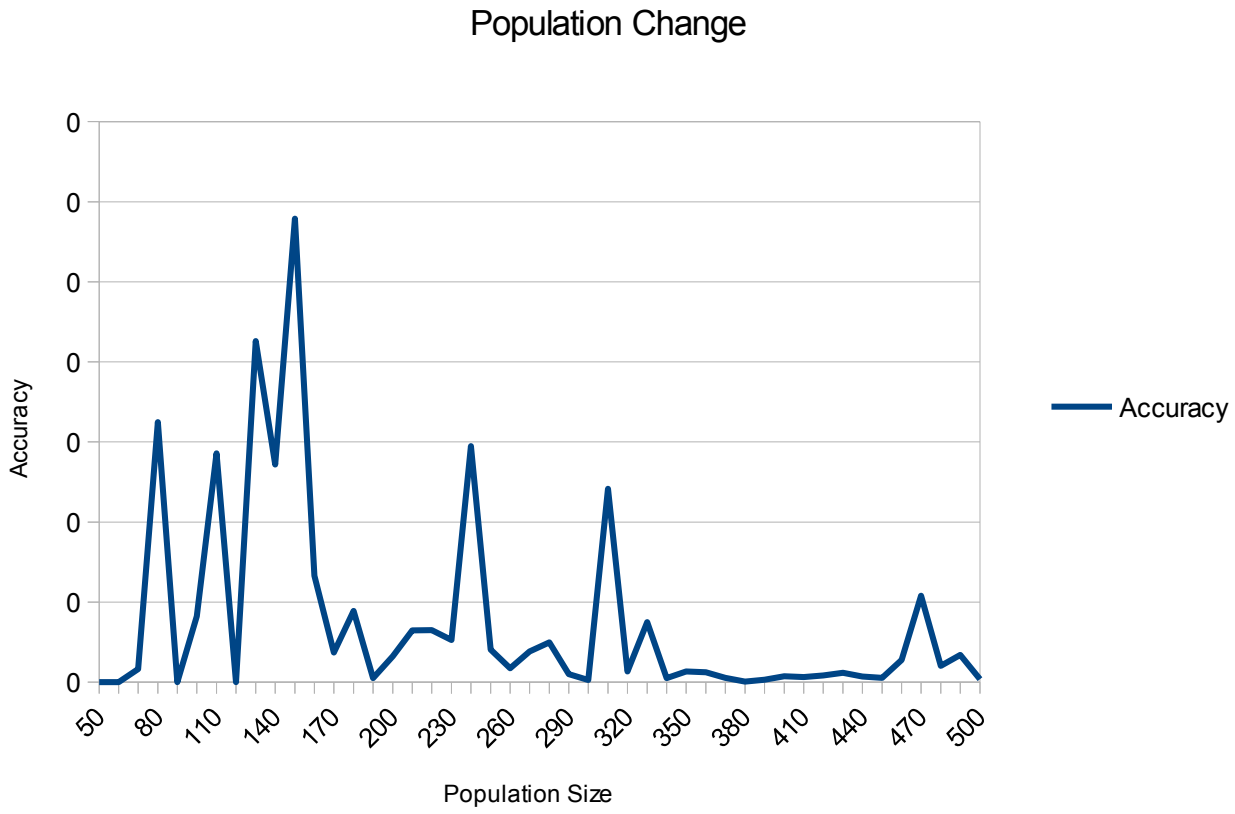
(3)

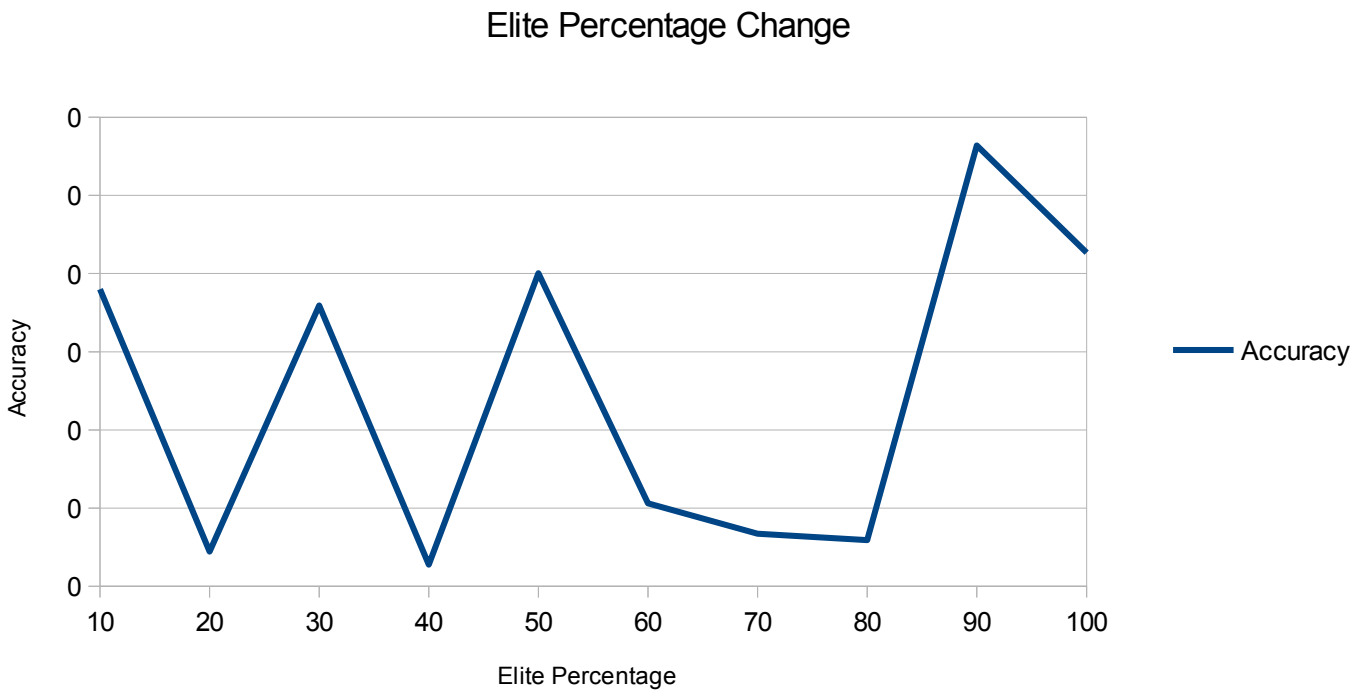

Generation Size Change
Accuracy
0
0
0
0
0
0
50
80
110
140
170
200
230
260
290
320
350
380
410
440
470
500
Generation Size
Accuracy


Population Change
Accuracy
0
0
0
0
0
0
0
0
0
0
50
80
110
140
170
200
230
260
290
320
350
380
410
440
470
500
Population Size
Accuracy

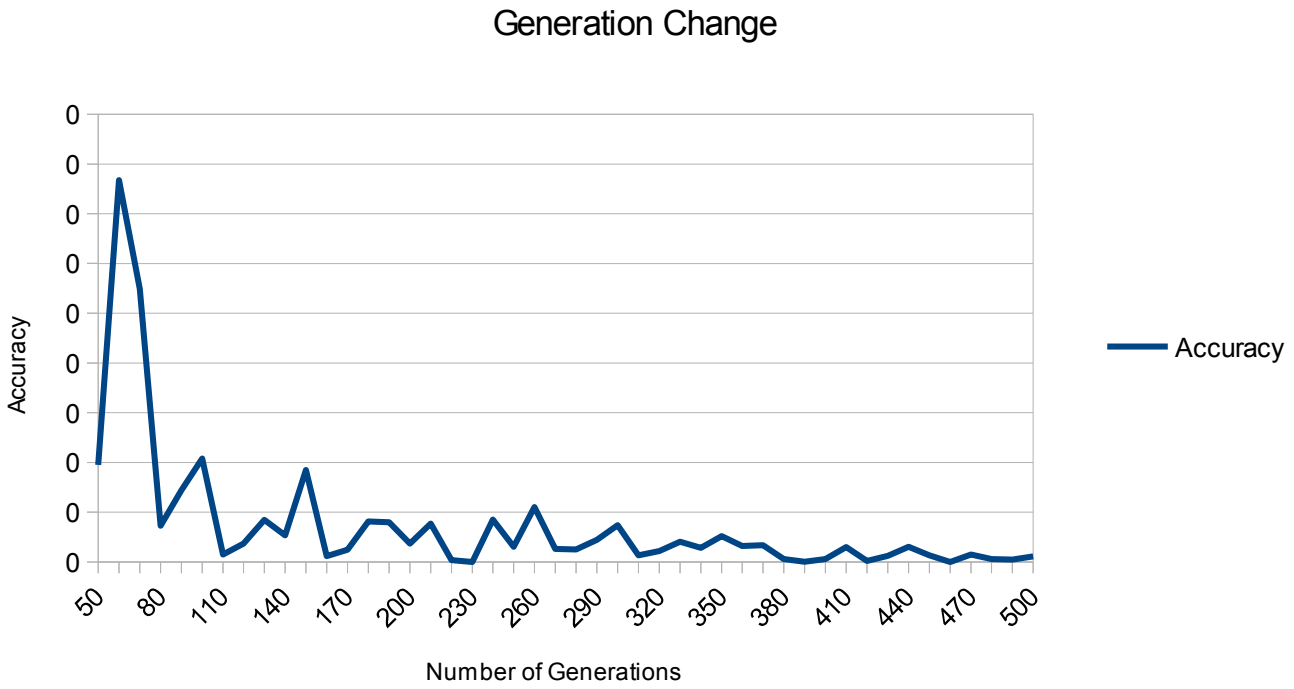
(4)

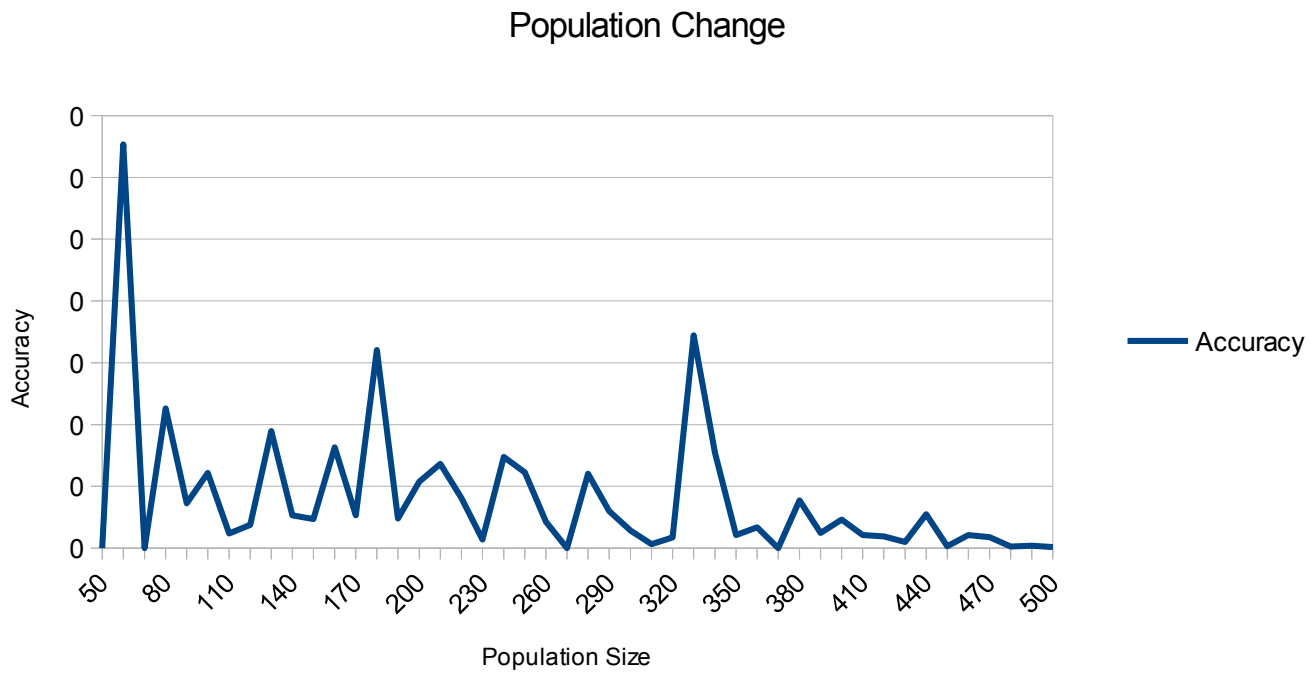

Generation Size Change
Accuracy
0
0
0
0
0
0
0
0
0
50
80
110
140
170
200
230
260
290
320
350
380
410
440
470
500
Generation Size
Accuracy

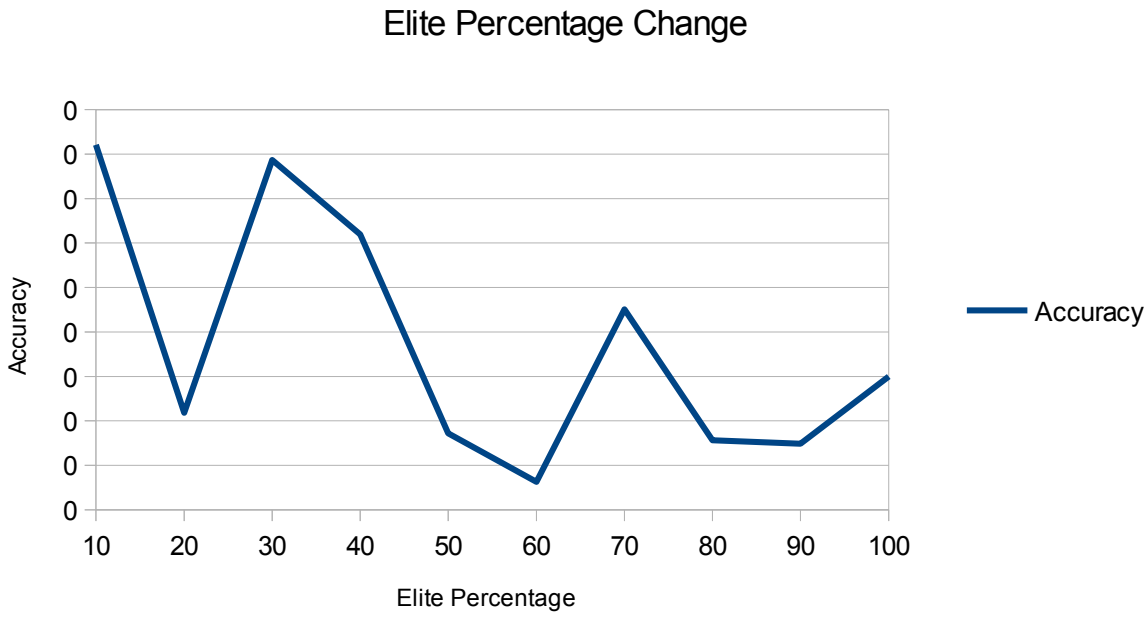

Population Change
Accuracy
0
0
0
0
0
0
0
50
80
110
140
170
200
230
260
290
320
350
380
410
440
470
500
Population Size
Accuracy


Elite Percentage Change
Accuracy
0
0
0
0
0
0
10
20
30
40
50
60
70
80
90
100
Elite Percentage
Accuracy

(5)


Generation Change
Accuracy
0
0
0
0
0
0
0
0
50
80
110
140
170
200
230
260
290
320
350
380
410
440
470
500
Number of Generations
Accuracy


Population Change
Accuracy
0
0
0
0
0
0
0
50
80
110
140
170
200
230
260
290
320
350
380
410
440
470
500
Population Size
Accuracy


Elite Percentage Change
Accuracy
0
0
0
0
0
0
0
0
10
20
30
40
50
60
70
80
90
100
Elite Percentage
Accuracy