B00060572
David Kelly
Assignment 2
Computational Intelligence
Stephen Sheridan

Code Changes

I was given a partially complete GA for assessing a population of chromosomes made up of 4 integer values. These 4 integer values were input in to the formula below.

*Fitness = (a+(b*2)+(c*3)+(d*4))-30*

In order to create a successful implementation I had to initialize values for the chromosome fitness, create an array of size 4 to store the genes *a,b,c,d* and set their initial to a random integer value from 1-30.

As shown in the following section **ValueEncodedChromosome** I created code that would complete the program. I needed to create access modifiers that allowed ValueEncodedChromosome objects to be created, given a length, given values, assess their fitness and compared to other ValueEncodedChromosomes. Finally I modified the method to print the values to the screen.

ValueEncodedChromosome

getGeneAt(int pos)
- Returns the element in the array genes at index position (pos)

setGeneAt(int pos, int val)
- Sets the value (val) of the element at the index position (pos)

mutateGeneAt(int pos)
- A new random integer from 1 to upper bound is generated and set at index position (pos)

setLength(int length)
- Sets the value of the private variable length related to the length of the chromosome

getLength()
- Returns the value for length

setFitness(int fitness)
- Sets the value of the private variable fitness related to the chromosome == 0

getFitness()
- Returns the fitness value for the chromosome

equals(ValueEncodedChromosome c)
- Compares the elements of chromosome (c) to the elements of current chromosome

toString()
- Prints all the elements of current chromosome to the screen

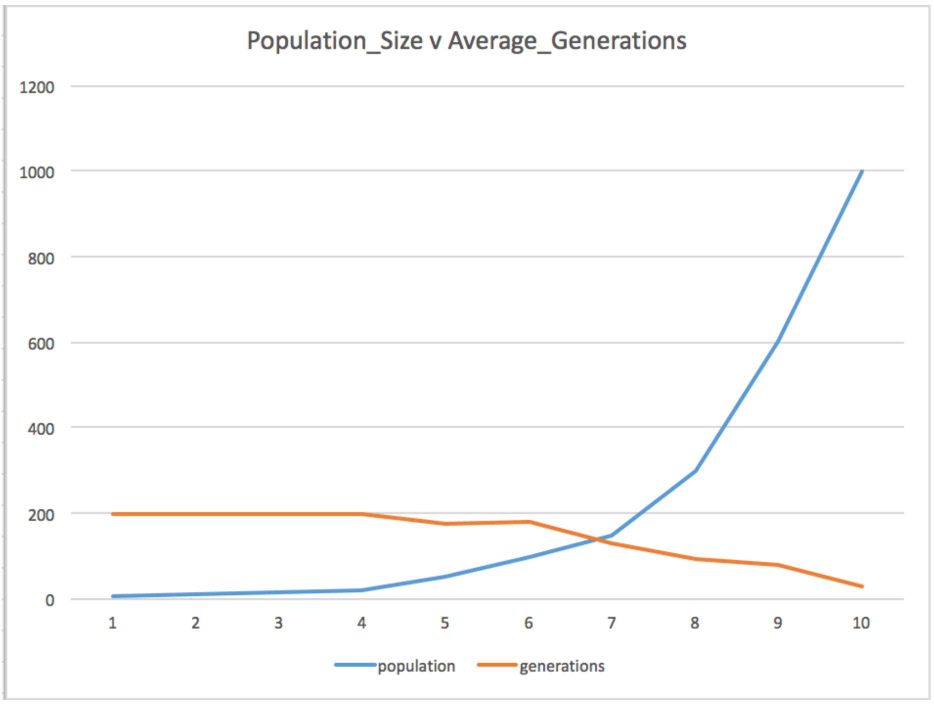
Population

evolve(boolean display)
- If the input (display) = true then the methods evaluate, sort, display, crossover, mutate, remove duplicates are called

converged()
- Loops through the population and returns true if the population[x] = 0

fitness(int a, int b, int c, int d)
- Takes 4 integer values that are inserted in to the formula *(a+(b*2)+(c*3)+(d*4))-30* and returns the answer

Population Size



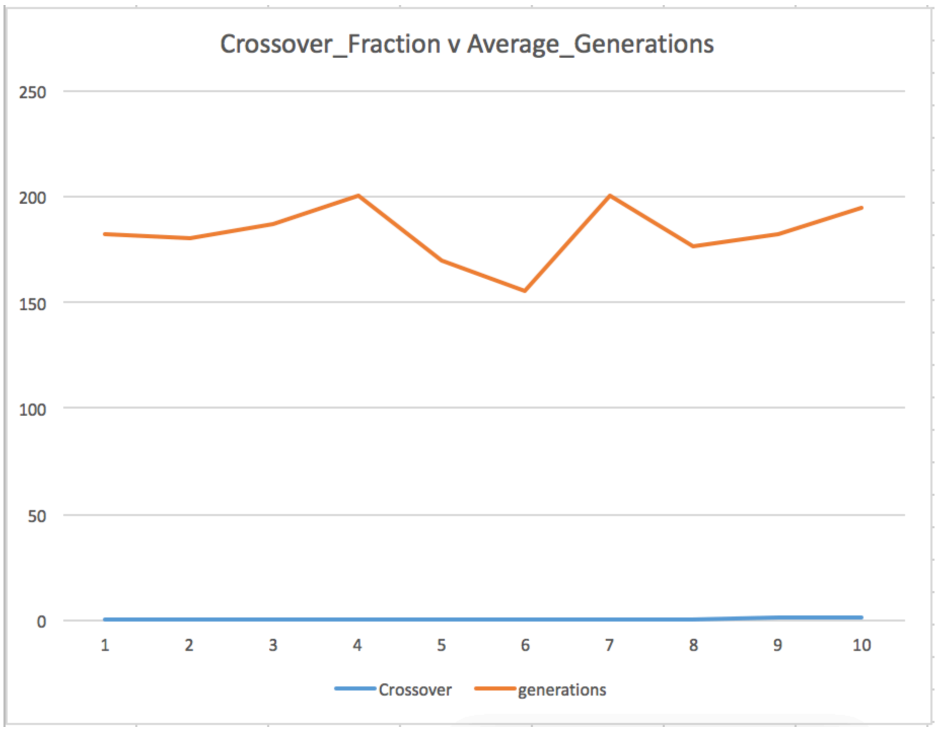
The above chart shows the decreasing average for generations to converge as the population increases. With a population size of 5, 10, 15 or 20, the average number of generations before a member of the population converged on the answer was 200 (the maximum value) this means that most of the time the population is too small to find the solution.

There is a direct correlation between the size of the population and the number of generations it takes to converge on the answer. As the population increases the ability to converge becomes greater as there are more opportunities per generation for chromosomes to find the solution.

At a population of 1000 the average number of generations to converge on the solution is 31, the lowest average of all population sizes in the trials. A full table of results is provided below.

Population	Generations
5	200
10	200
15	200
20	200
50	175
100	180
150	131
300	93
600	80
1000	31

Crossover Fraction



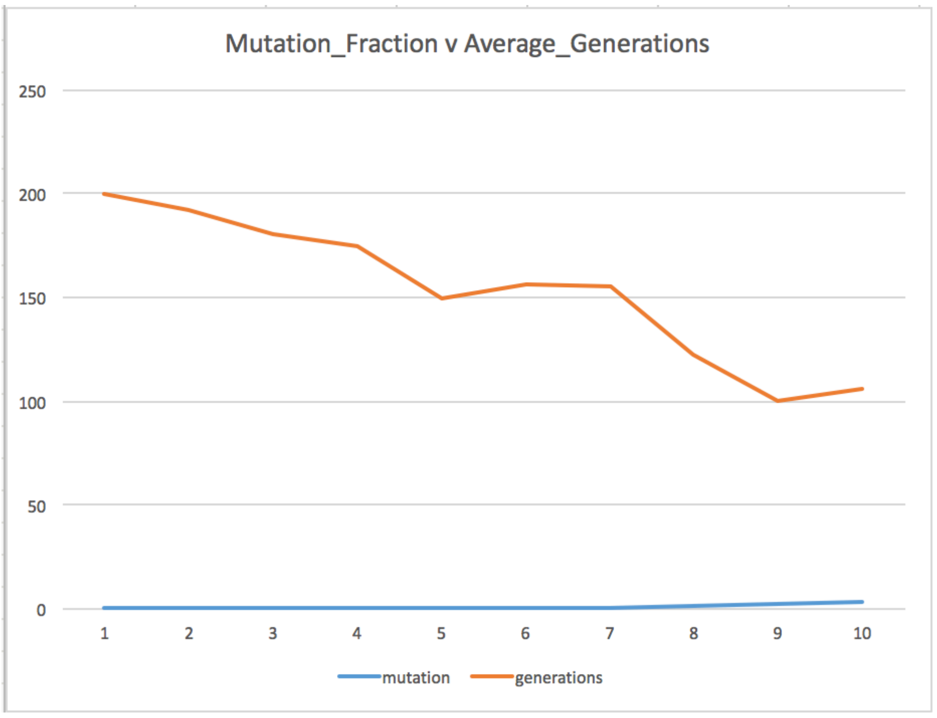
The above chart shows the average convergence value correlated to the crossover fraction. The crossover fraction determines how much of the chromosome is made from parent chromosomes. The genetic material makes up the new generations genes. Ranging from 0% to 100%. A crossover fraction of 0% means that the new chromosome will contain now genetic material from parent chromosomes. A crossover fraction of 100% will create a chromosome made up entirely from parent chromosome.

The crossover fraction can slow down or speed up the algorithms convergence rate. Having no crossover fraction creates new generations disconnected from previous solutions that can be close to a solution and likewise, having a 100% crossover rate can stagnate the convergence rate by completely copying parent chromosomes that may not be near a solution.

The chart proves that a crossover fraction between 40% and 70% is the optimal crossover fraction as it contains nearly equal amounts from two parent chromosomes. Altering the new chromosomes genetic material to make a unique combination of both chromosome1 and chromosome2. The average convergence rate at 50% was lowest with a value of 155. A full table of results is provided below.

Crossover	Generations
0	182
0.1	180
0.2	187
0.3	200
0.4	170
0.5	155
0.7	200
0.8	176
0.9	182
1	195

Mutation Fraction



The chart above shows the average generations convergence value with a fixed population size of 50 and different percentages of mutation to the chromosome. Mutation on the chromosome creates new off spring from previous generations by creating new random locus values depending on the percentage of mutation.

The maximum 300% mutation creates 3 random new chromosomes in the population. 300% mutation yielded the average fastest convergence at 106 generations. There is a correlation between the mutation fraction on a population this size. By mutating 10% of genes per generation the genealogy remains closely connected to the previous generations making the process a slow but visible development. By introducing 3 new chromosomes the process adds a little bit of randomness to the generations and can increase the convergence speed but also distance itself from past generations making it more difficult to connect the chromosomes parents. A full table of results is provided below.

Mutation	Generations
0	200
0.1	192
0.2	180
0.3	175
0.4	149
0.5	156
0.7	155
1	122
2	100
3	106