MECH 579
Multidisciplinary Design Optimization
Siva Nadarajah

Project 4: Derivative-Free Optimization

20/20

Dritan Harizaj
260426327

November 12, 2013

Introduction

The following Gaussian distribution function was to be minimized without the use of derivatives for $-5 \leq x, y \leq 5$:

$$f(x,y) = -10e^{-[(x-1)^2+0.25(y-1)^2]} - 5e^{-[(x+1)^2+0.25(y+1)^2]} - 5e^{-[5(x-3)^2+0.25(y-3)^2]}$$

The function has 3 minima in this region of the x-y plane:

$$(-1,-1) \rightarrow -5.0674\ (local)$$

$$(1,1) \rightarrow -10.067\ (global)$$

$$(3,3) \rightarrow -5.0674\ (local)$$

A genetic algorithm with binary chromosomes was implemented to meet this goal. The population size is 12 and the number of bits 8 (i.e. chromosome length 16 sinceunless otherwise specified. The maximum number of generations was set to 100. Regardless of the number of bits in the chromosomes, after they were converted to decimal they were subsequently normalized to be between -5 and 5.

Two methods were used to determine convergence:

1) Lowest value of the population
2) Average value of the population

Single point-crossover with a randomized point was used for mating. The fitness was determined simply by sorting the population after evaluating the objective function, with the lowest values being placed near the beginning of the array, and the highest values relegated to the end.

Note that elitism was used for the fittest individuals to proceed to the next generation immediately, while the other half was discarded and replaced by the offspring of the elite. The roulette method was used to determine which of the fittest would mate.

The mutation was set to 0.05 unless otherwise specified, meaning that 5% of the bits of the entire population had a chance of flipping after the mating process. One very important modification to the algorithm, which sped up convergence, was for the fittest individual *not* to mutate.

The population size, number of bits, and mutation rate were then modified to test the effects on convergence.

1. Results

Convergence

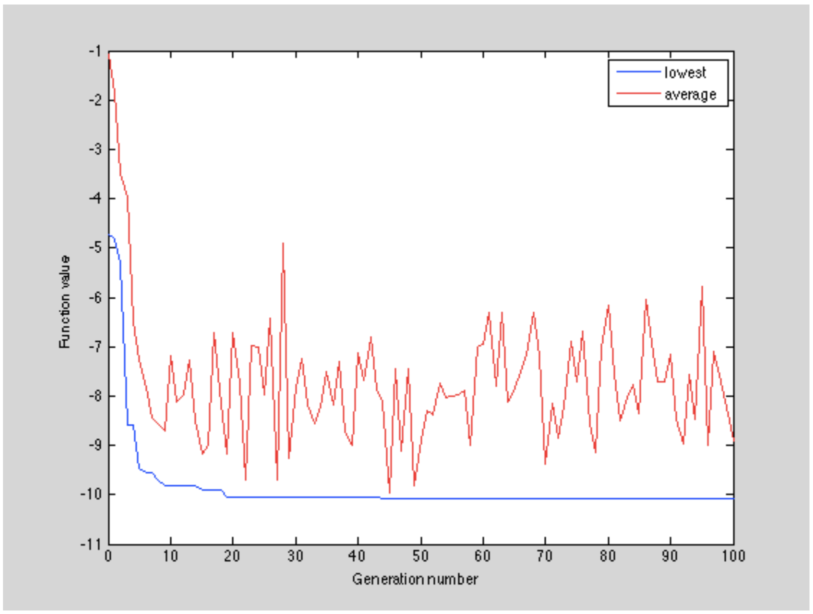

Figure 1: Convergence (1 , 1) -> -10.0674

✓

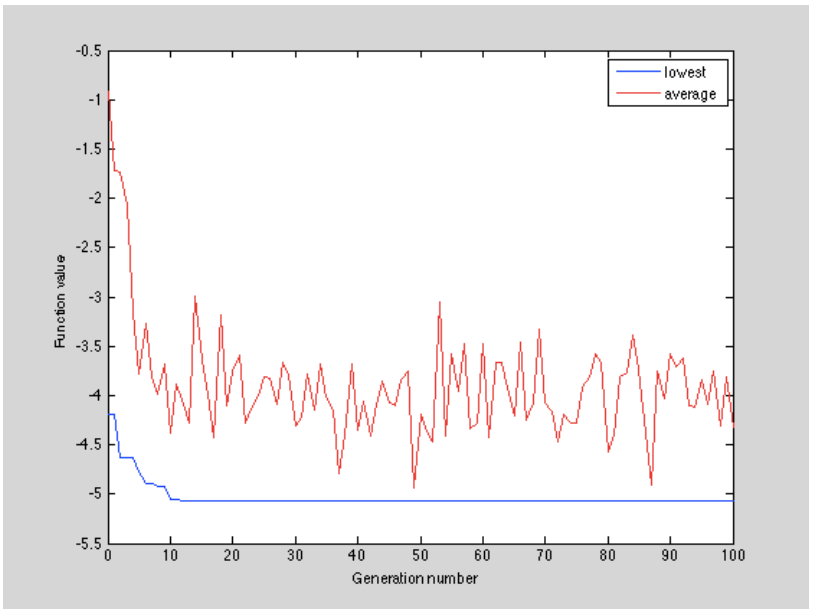

Figure 2: Convergence (-0.96078 , -0.96078) -> -5.0722

✓

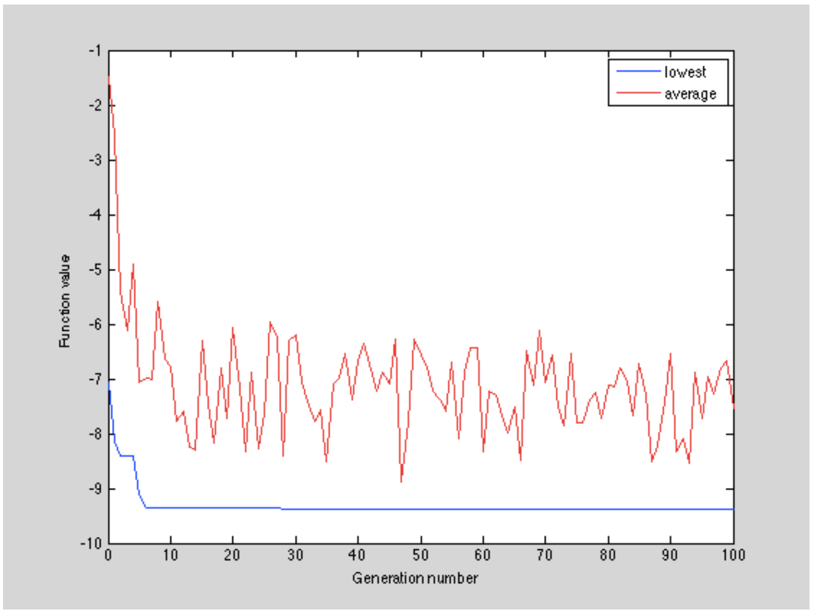

Figure 3: Convergence (1.2745 , 1) -> -9.3782

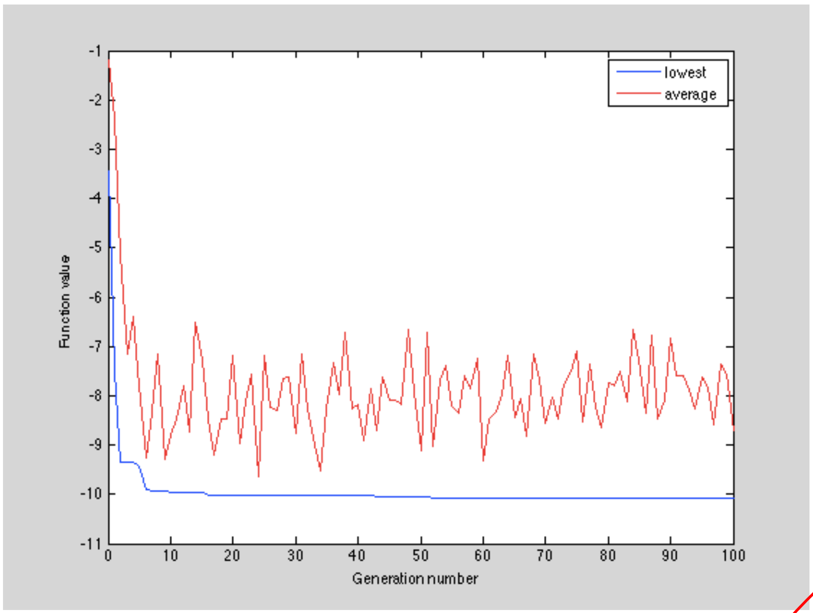

Figure 4: Convergence (1 , 1) -> -10.0674

Four different trials are shown above. In the vast majority of cases, the solution tended to converge to the global minimum of (1,1)-> -10.0674. In some cases the minimum was not reached within the 100 generation limit (Fig.3), but given enough iterations it would have most certainly converged to (1,1). On rare occasions the solution converged to the local minimum (-1,-1) -> -5.0674 (Fig. 2). The (3,3) local minimum was reached even more seldom, likely due to the fact that its basin of convergence occupies a smaller region in the x-y plane from -5 to 5 (refer to contour plots in a few pages).

Notice that the lowest value (blue) takes a while to converge in most cases, whereas the average value (red) generally does not, due to mutation. The mutation rate ensures that even if the fittest individuals mate, some of them will branch off to search in alternate regions of the x-y plane. The bit flipping generally throws the results of the mating into disarray, preventing the average value from converging.

Clusters

Shown below are several plots of the clusters of points in the x-y plane after 100 generations. The redder the color of the point, the earlier the generation; the darker the point, the later the generation. Notice that there are 3 major cases that occur: in order of decreasing likeliness:

1) Cluster only around the global minimum

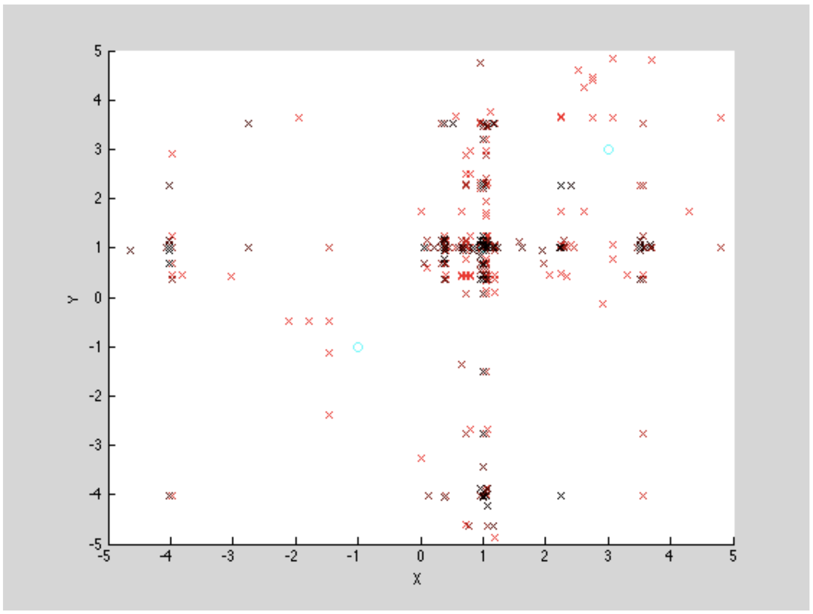

Figure 5: Cluster (1 , 1) -> -10.0674

2) Cluster around both global and local, but some early convergence toward local

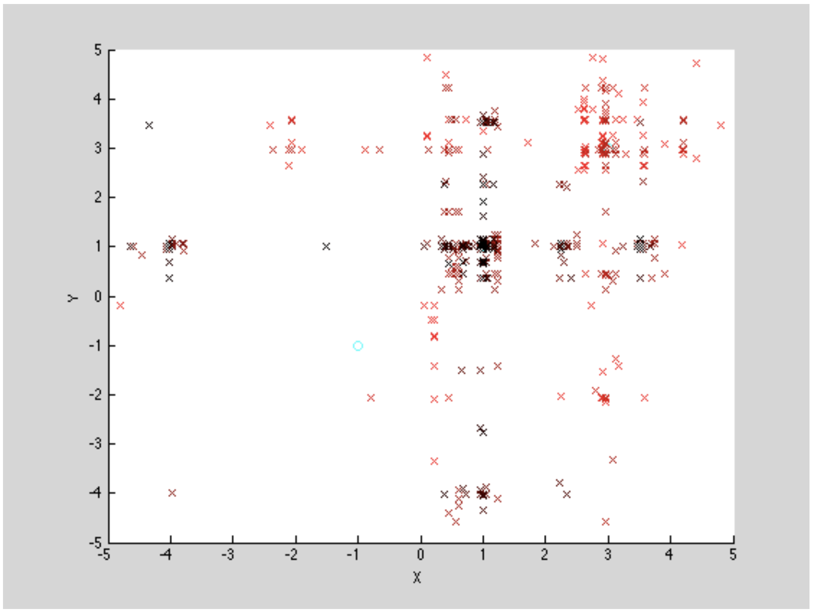

Figure 6: Cluster (1 , 1) -> -10.0674

3) Cluster only toward local (rare)

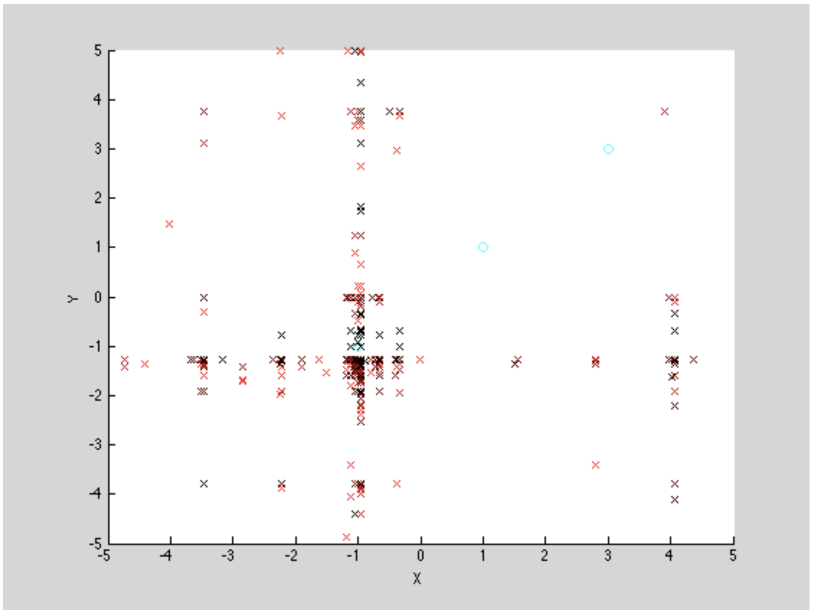

Figure 7: Cluster (-0.96078 , -1) -> -5.071

Path

To see how the lowest value fluctuates, several tests were done by tracking the path of the lowest value of the population around the contour plot. Once again, we can identify several patterns in the solutions:

1) Cluster only around the global minimum

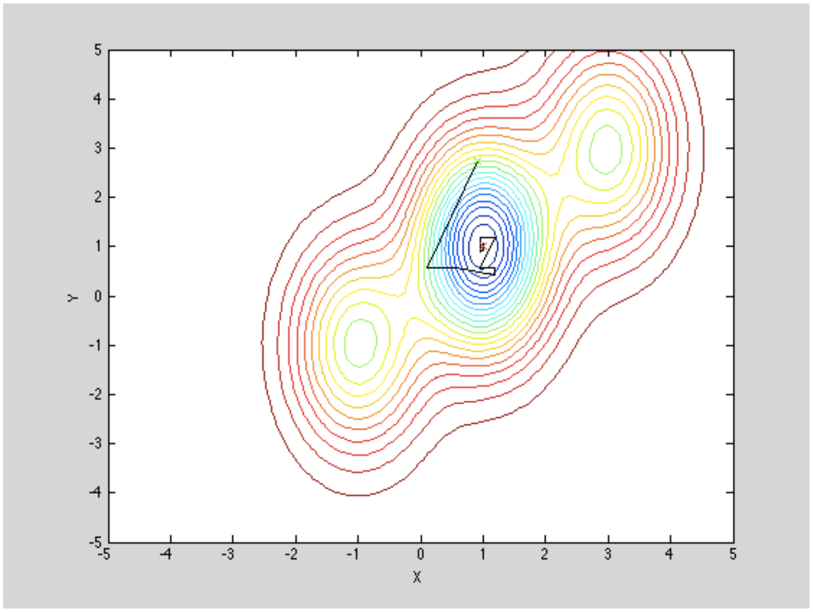

Figure 8: Contour: (1,1) -> -10.0674

✓

2) Cluster around both global and local, but some early convergence toward local

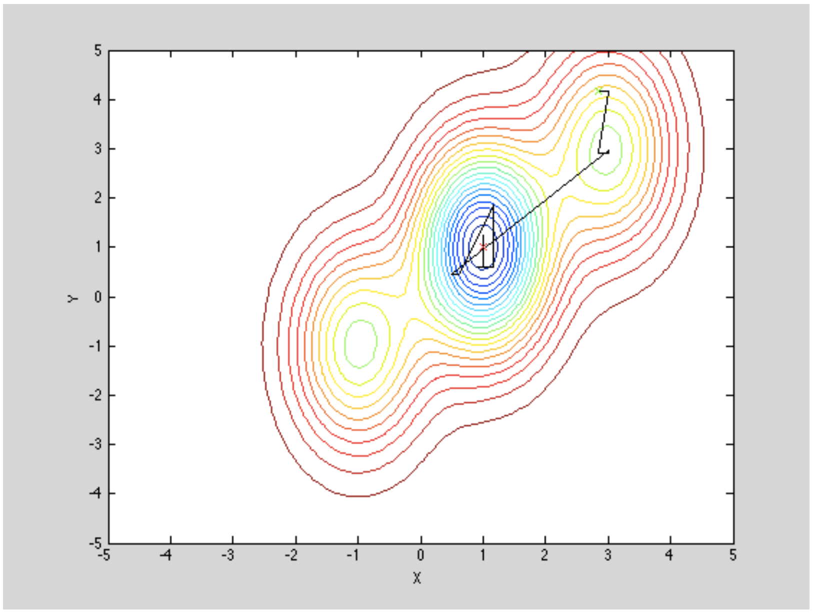

Figure 9: Contour: (1,1) -> -10.0674m,

✓

3) Cluster only toward local (rare)

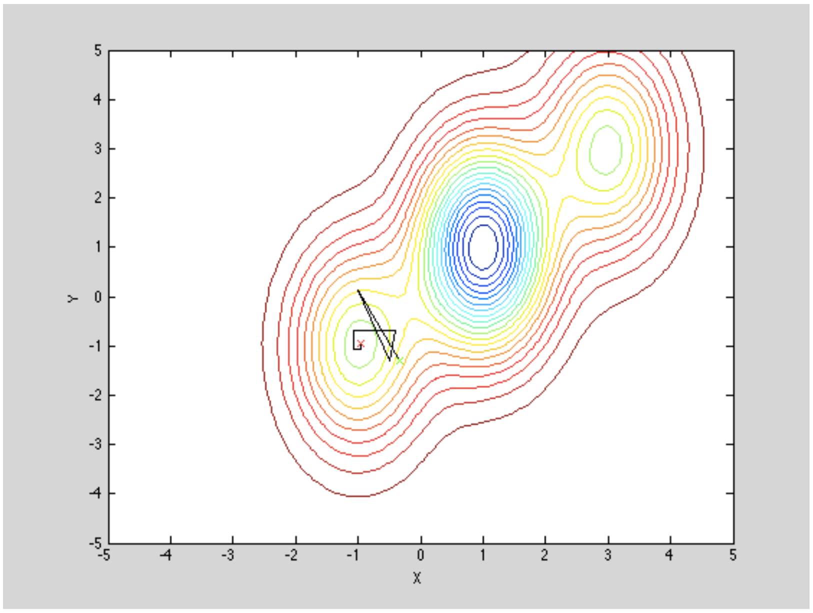

Figure 10: Contour: (-0.96078 , -0.96078) -> -5.0722

✓

2. Varying population size and chromosome bits

Population

We'll be looking at the extreme cases for both population size and chromosome bits. Setting population size to 2:

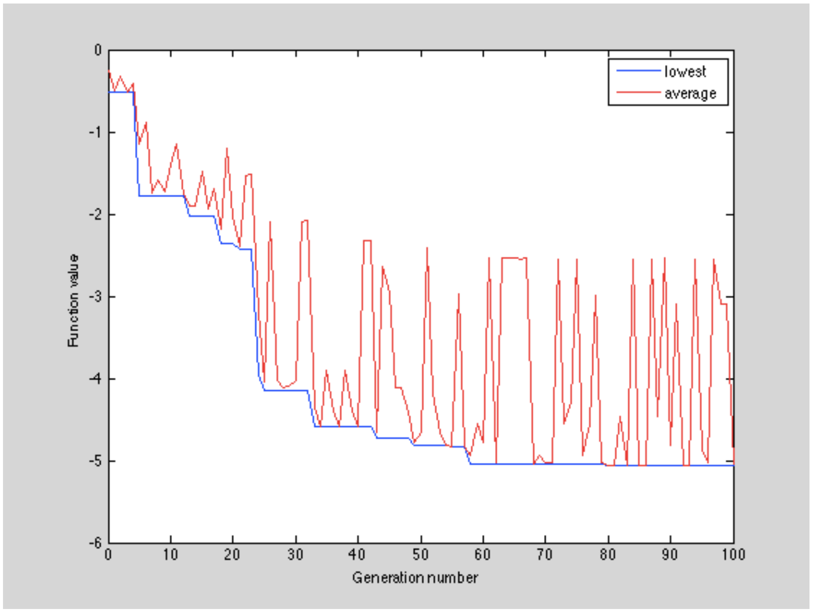

Figure 11: Convergence (-0.96078 , -0.92157) -> -5.0696

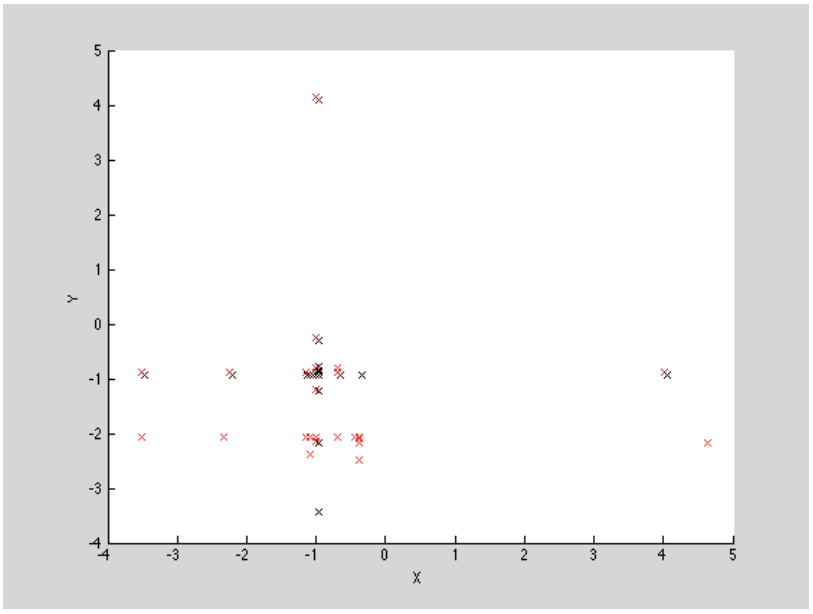

Figure 12: Cluster (-9.6078 , -0.92157) -> -5.0696

Setting population size to 100:

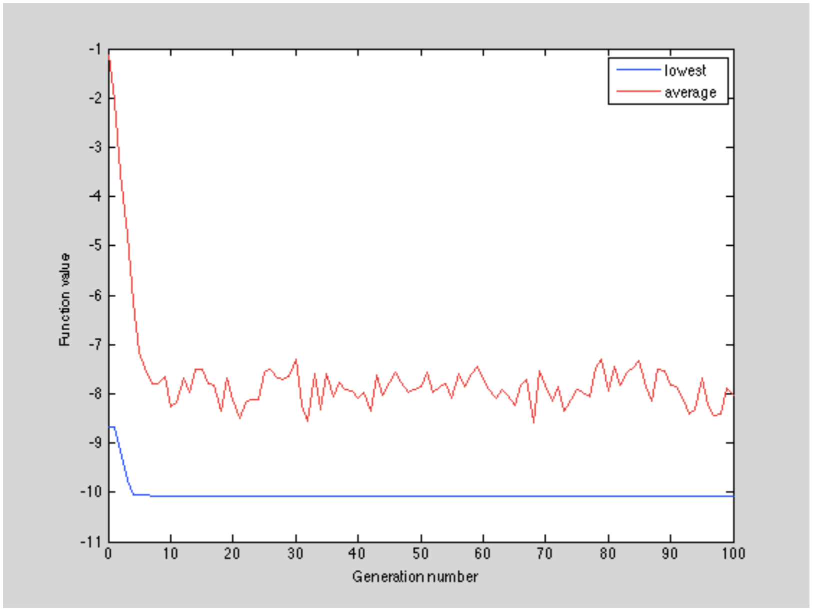

Figure 13: Convergence (1,1) -> -10.0674

✓

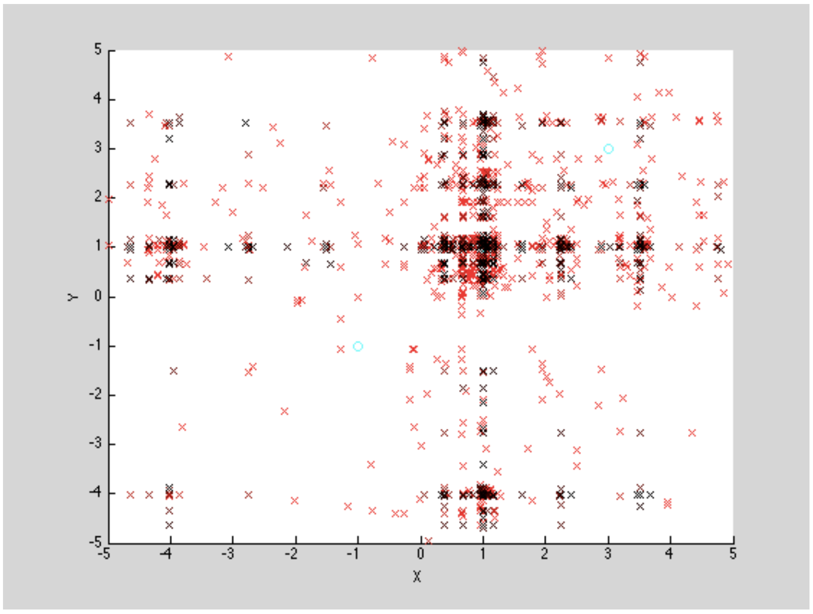

Figure 14: Cluster (1,1) -> -10.0674

A small population greatly increases the chances of finding a local minimum, but also requires many more generations to converge. Fewer points means that the initialization of the population plays a much larger role in the convergence of the algorithm. Even mutation plays less of a role, since the total number of bits is lower and hence a mutation is much less likely to occur.

On the other hand, a bigger population size almost guarantees convergence to the global maximum, whereas local minima are almost impossible to be found. Having a large population allows the algorithm to scan a wider area of the x-y plane, greatly increasing the chances of finding the lowest overall value. If even one point falls within the basin of convergence of the global minimum, the subsequent generations will be descended from it, greatly reducing the chances of finding a local minimum since it cannot compete with the fitter, lower values close to the global minimum. The average value of the population does not fluctuate as much due to mutation, likely due to the high number of very fit chromosomes that pass through to the next generation via elitism. The runtime of the program also significantly increases with the population size.

Chromosomes

Setting number of bits to 4:

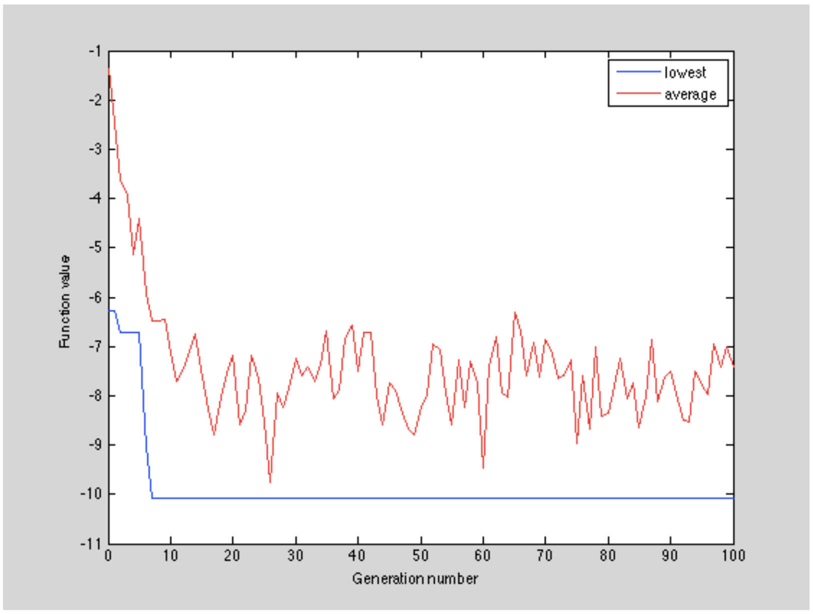

Figure 15: Convergence (1,1) -> -10.0674

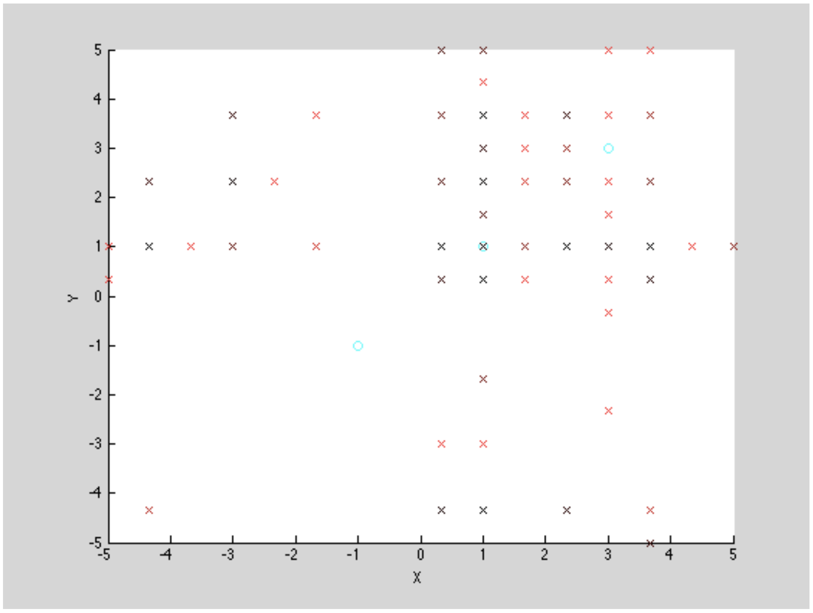

Figure 16: Cluster (1,1) -> -10.0674

Setting number of bits to 30:

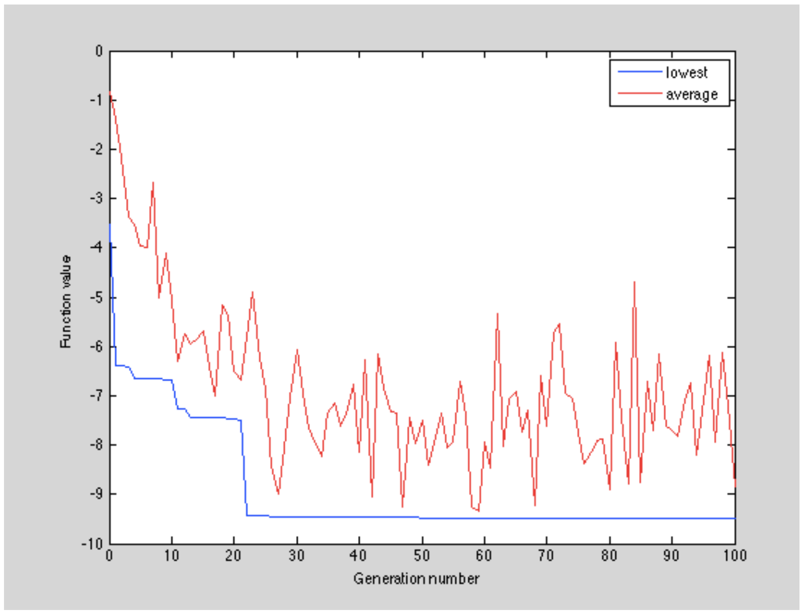

Figure 17: Convergence (1.25 , 0.93746) -> -9.4781

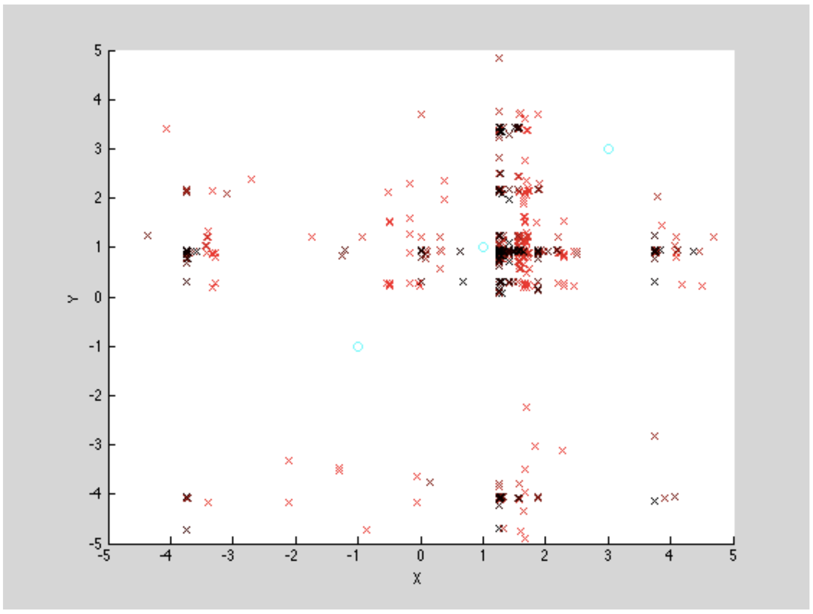

Figure 18: Cluster (1.25 , 0.93746) -> -9.4781

Note that although more chromosomes means more bits, the range is still -5 to 5 regardless. After the binary chromosomes are converted to decimal, they are then normalized to be between -5 and 5. This is why the cluster map for the 4-bit case is very coarse, while the 30-bit case is very fine.

The lower the number of bits, the "coarser" the region of possible values in the x-y plane is. A greater amount of bits is generally preferable since it allows us to better "comb" the region and provides a better resolution for scanning intermediate values. Increasing the number of bits therefore helps in finding the local minima, since their basin of convergence is smaller than that of the global minimum.

Generally speaking the coarseness associated with the 4-bit case is a bad thing, but in this case, since all 3 minima are integers, it is actually beneficial. Indeed, the accuracy of the 4-bit case is much better than that of the 30-bit case. Even after 100 generations the population has only converged to -9.4781, quite far from the global minimum of -10.0674. Meanwhile, the 4-bit case had no problem converging to the exact value.

This great reduction in accuracy is likely due to mutation. Having a large number of bits means that the chromosomes are very large in length, providing more opportunities for bits to flip. Even a small 5% mutation rate can be enough to destabilize the convergence when approaching the global minimum. On the other hand, if the number of bits is low and the mutation is 5%, it is very likely that many chromosomes will remain completely unaffected.

3. Varying Mutation

Setting mutation rate to 0:

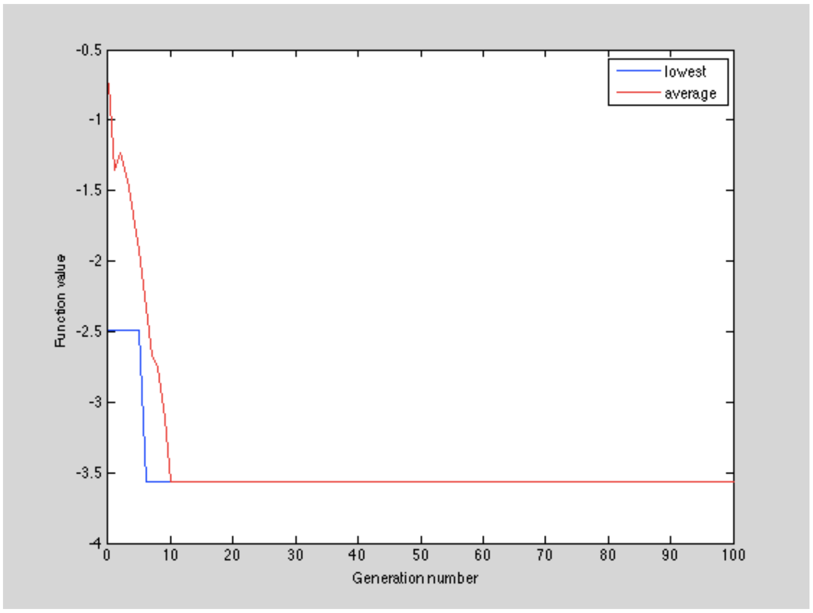

Figure 19: Convergence (3.5490 , 3.3922) -> -3.5629

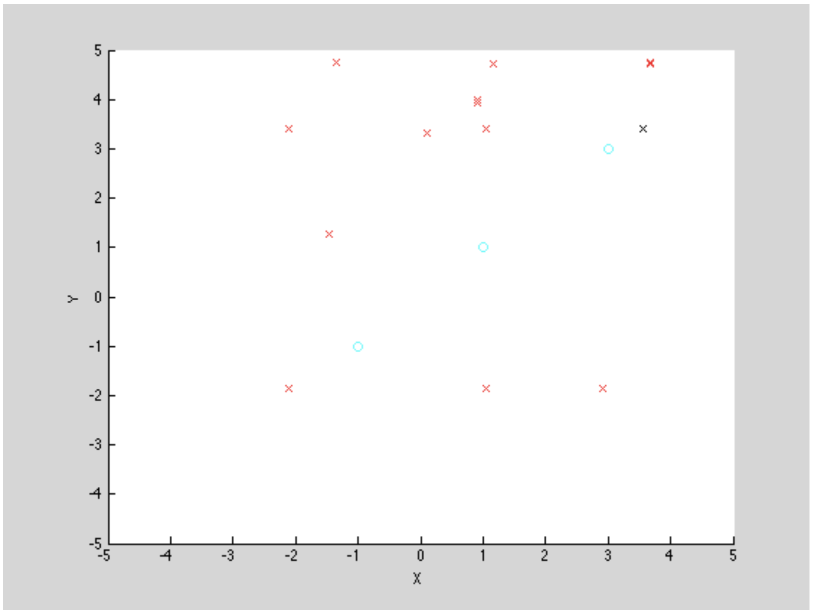

Figure 20: Cluster (3.5490 , 3.3922) ->-3.5629

Setting mutation to 0.8:

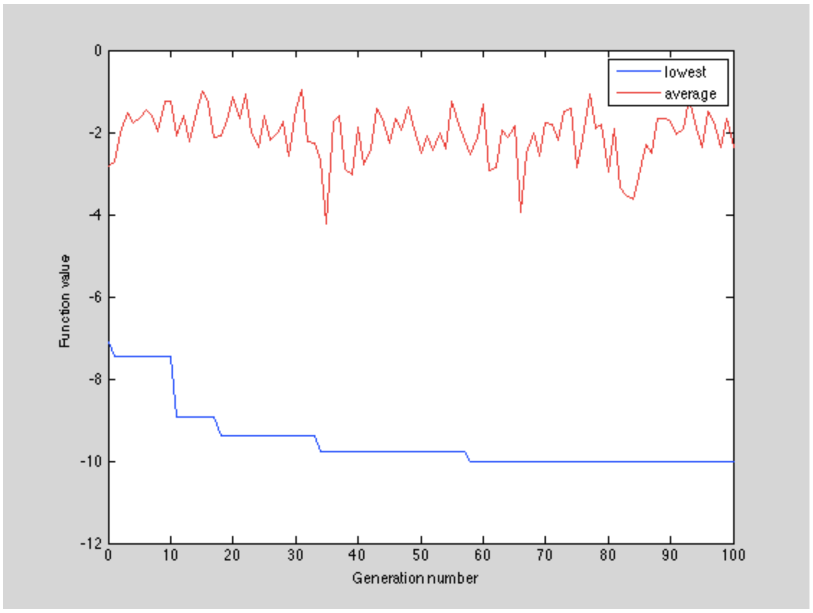

Figure 21: Convergence (1.0784 , 1.0392) -> -10.006

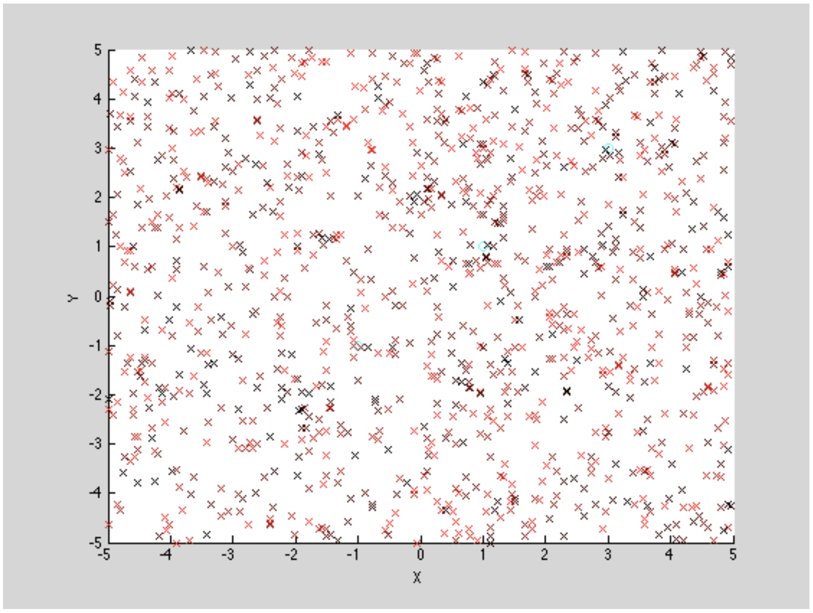

Figure 22: Cluster (1.0784 , 1.0392) -> -10.006

If the mutation is set to 0, convergence becomes near impossible, both for local and global minima. The problem becomes extremely dependent on initial conditions to provide variation. With sufficiently large population size and bit size, it might be possible to still converge provided there is sufficient initial randomness.

If the mutation rate is too high, there is much more fluctuation in the values of the population, which is reflected in the average values of the population (red curve). The higher the mutation rate, the higher the average value of the population. A high mutation essentially undermines the effect of reproduction and evolution. There is little point in reproducing if most of the bits will simply be flipped over. The algorithm becomes not so much a genetic one, so much as a random walk. The problem is then solved by stumbling upon the lowest value due to mutation rather than due to evolution.

As can be seen from the lowest value of the population (blue curve), a high mutation rate slows down convergence because it adversely affects the offspring of the fittest individuals by tinkering with their chromosomes.

Conclusion

The specific parameters for genetic algorithms are very problem-dependent. For this particular case, a bit size 4-12 is ideal, with a mutation rate of 3-8%. The population size should be around 10 if desiring to find local minima, or as high as possible if desiring to find the global minimum. The number of generations to convergence is usually less than 100. The average value of the population is generally not a good criterion for determining convergence due to the effect of mutation. Using the best function value is typically a better option.