Assignment 2: Genetic Algorithm

Mikael Kindblom

October 25, 2020

Contents

1 Problem

We want to optimize the following functions:

$$f(x) = 20 + x_1^2 - 10cos(2x_1\pi) + x_2^2 - 10cos(2x_2\pi) \tag{1}$$

$$\text{subject to: } -5.12 \leq x_1, x_2 \leq 5.12$$

$$g(x) = -x_1 sin(\sqrt{|x_1|}) - x_2 sin(\sqrt{|x_2|}) \tag{2}$$

$$\text{subject to: } -500 \leq x_1, x_2 \leq 500$$

Since in the assignment description it is just written "optimize", it is not clear if we should minimize or maximize. We therefore choose to to both for both functions.

2 Initialization

2.1 Fitness function

The first step is finding its fitness function. This function should give an indication on how good the solution is. Since the function value of our functions already are a direct measurement of good/bad solution, each original function becomes the corresponding fitness function.

2.2 Population

We now create a population. This population will consists of chromosomes with binary encodings. Since we have two variables, A chromosomes and B bits, the population is a $[2\,x\,A\,x\,B]$ -matrix. The bits in this matrix are just chosen randomly. Now we need to decode and normalize the binary string.

2.3 Decoding

The binary string (genotype) must be decoded to its real value (phenotype) using the equation

$$T_i = T_{imin} + \frac{(T_{imax} - T_{imin})DV(S_i)}{2^{l_i} - 1}$$

where Timin and Timax are the lower and upper bounds of the variable T_i, $DV(s_i)$ is the decoded value of the string s_i, and l_i is the string length used to code the i^{th} parameter. For example, the binary string 10011101101010 = $(2^1 + 2^3 + 2^5 + 2^6 + 2^8 + 2^9 + 2^10 + 2^13 = 10090)$, can be decoded as

$$T_i = T_{imin} + \frac{(T_{imax} - T_{imin})DV(S_i)}{2^{l_i} - 1} = -500 + \frac{(500 - (-500))10090}{2^{14} - 1} = 115.8823$$

.

2.4 Adjustment of fitness values

If any fitness value happen to contain negative values we add the inverted amount of the most negative fitness value to the whole fitness value vector. We then normalize the fitness values of the population by giving its individual proportion divided by the sum of the population.

3 Selection

Now the selection for mating partners starts. By using either Roulette- or Tournament-selection we end up with two children for each iteration. We iterate for half of the population size such that we in the end have created an entirely new population.

3.1 Roulette

More concretely in the Roulette selection, since the sum of the normalized fitness values takes up a space of exactly 1, we can create a roulette wheel where larger fitness values correspond to higher probability of getting selected. In the minimization case we instead have that smaller fitness values correspond to higher probability of getting selected.

As we see in the table down below, we have a maximization problem with a population size of 6 where the normalized fitness values are to the left. To the right we have cumulative sum where the range 0-0.32 corresponds to the first chromosome, 0.32-0.41 corresponds to the second chromosome, and so on. If for example we generate the random number 0.44 it will then be greater than 0.41 but less than 0.64, which makes for a selection of the third cromosome.

Normalized fitness value	Cumulative sum
0.32	0.32
0.09	0.41
0.23	0.64
0.12	0.76
0.04	0.80
0.20	1

We select the same number as our population size which brings us a new set of chromosomes. After this we completely randomly select two chromosomes of the new set. These are our parents which will mate and create offspring.

3.2 Tournament

We also have the Tournament selection which is simpler than the Roulette. Here we randomly choose two contenders that will compete. We do this for as many times as we have chromosomes in our population such that we end up with 10 winners. Two of these winners will then get randomly selected and become the new parents.

4 Crossover

After selection we apply the crossover. Here the parent chromosomes get mixed which creates the children. In the solution we have defined two different types of crossover techniques, the single- and double-point crossover.

4.1 One Point

The single point basically just takes a random bit index where everything up to that index gets directly copied from the mother to the child. Everything after the index instead gets copied directly from the father.

4.2 Two Point

The two point crossover starts just as the one point crossover but then another random number is generated from which this new index the following bits instead will be inherited from the mother again.

5 Mutation

After the crossover is done the children may have mutation in their chromosome. Therefore, according to a small probability, one random bit in their genetic information could get flipped.

6 Iteration conditions

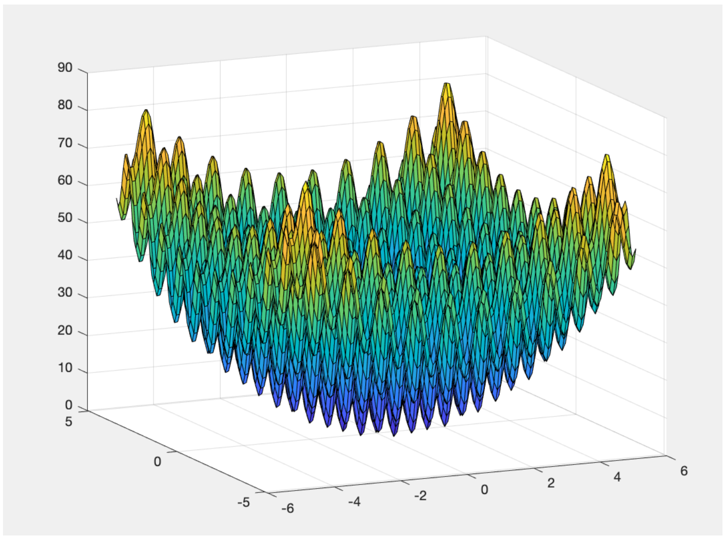
The new children make up the new population and through iterating the genetic information of the new population should become better and better (approaching the extreme point). We can use different stop conditions for the algorithm. One is to just check that there has been an improvement(within some tolerance) in best fitness value compared to the last X iterations. Here it is difficult however to choose X since it could be that for example we are inside a local minima which happen to be very deep, thus needing to increase the value of X. We can also just let the algorithm run for a while by setting a maximum iteration threshold.

7 Results

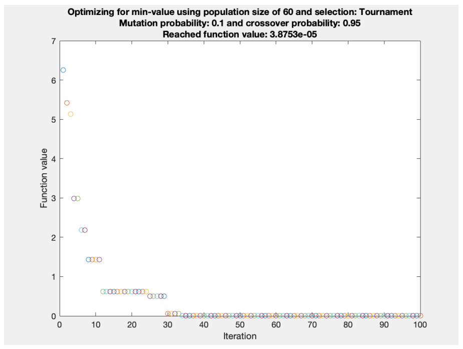
Note that these graphs shows the best value found *so far*. This is what partly gives rise to the continous horizontal lines. What this means that the algorithms current iteration might be a worse value than the best value found so far, and therefore plots the best found value instead of the *current best value*.

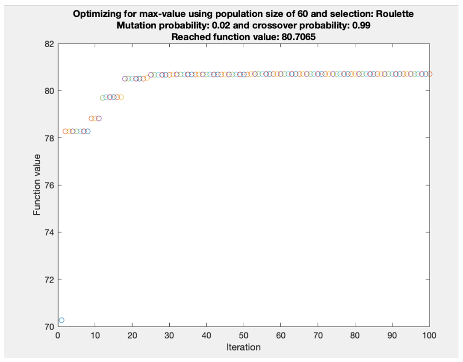
7.1 Function 1

This image below is how the surface of the first function looks on which we are moving. The global minima is around 0 and the global maxima around 80.



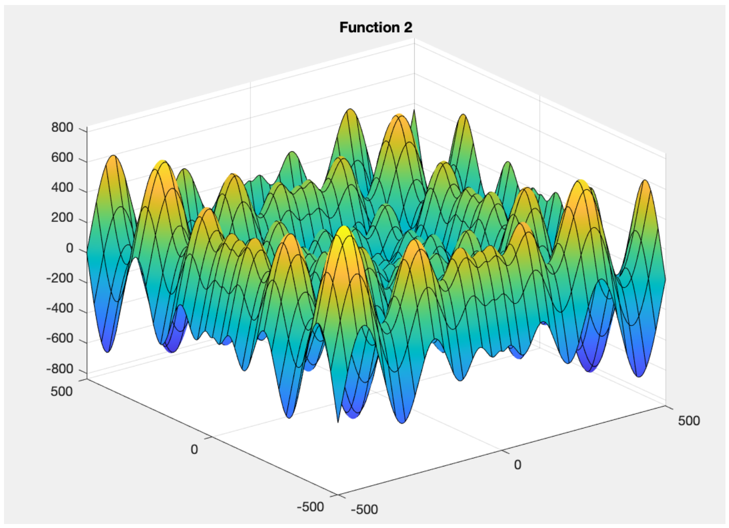
If we run the algorithm for 100 iterations we get the following results for minima and maxima (The parameters are specified in the plot):



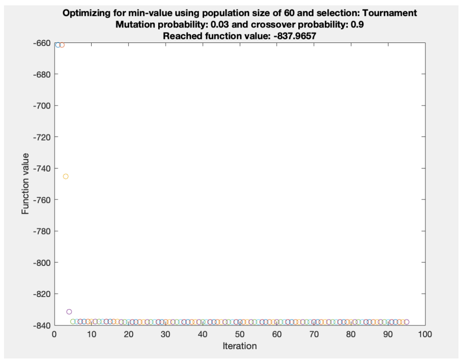


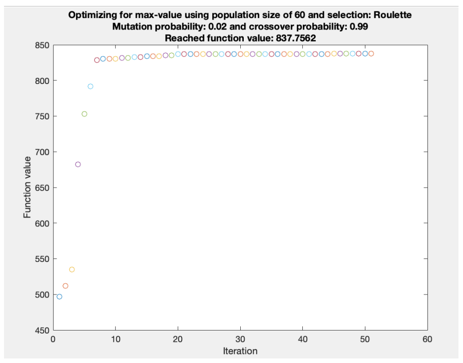
7.2 Function 2

This image below is how the surface of the second function looks. The global minima and maxima are around [-838,838].



We now run the genetic algorithm. Note that we have another stop condition in these runs. Here we instead stop the algorithm once there is no improved solution for 50 iterations.





8 Discussion

The results look pleasing, we manage to reach the optimal solutions. However since the whole algorithm is based on probability, we can happen to get non-optimal solutions. To manage this we can adjust the different parameters.

- We have bit resolution- perhaps the best found solution is not good enough since the resolution s not good enough?

- We also have the population size - by increasing this we get a better initialization and when iterating we get more possible combinations to pick from, which means we will approach the solution faster. It however takes more computations.

- Next up we have the mutation probability to tweak. If we happen to always get stuck in a non-optimal solution we can increase the mutation probability such that it could give the necessary "push" to break out of the repeating pattern.

- Tournament or Roulette? This is an interesting topic since they produce results a bit differently. In Roulette the selection if random but have some "affinity" for certain chromosomes. To illustrate this "randomness with some affinity" I tried created an array of 10 numbers, 1 to 10. I then applied the Roulette wheel selection which gave me two parents as output. I did this for 10000 iterations and then took the average. On average, the selection was [4.0 4.2] for the minimization case and [6.8 7.0] for the maximization case. Note that two *different* numbers have to be created, so for example the best possible selection for maximization is [9 10]. We can therefore see that [4.0 4.2] and [6.8 7.0] have about a value of 1.5 to the center value 5.5 and therefore shows sign of "affinity" to certain chromosomes depending on min/max-optimization.

  This means that the Roulette selection actually has a tendency to select the best ones. Tournament is however rather always sorting out the worst ones, since they will never win the competition. The worst

chromosome will *always* be sorted out.
When comparing Roulette to Tournament we see that Roulette does not necessarily kill the worst chromosome where tournament selection always will.

- Crossover technique - Single point will always copy the beginning or end from each parent when creating the chromosome. Double point instead relies on two random numbers and in turn has more possibilities of changing the genetic data, which makes it superior. To make it even better we could introduce more random variables to make it 3 point, 4 point.. and so on. The more variables we introduce, the more accurate it becomes to a "real" genetic crossover. It is not that your entire genetic code is just split in two continuous parts from your mom and dad, rather, specific chunks of code you inherit from your mother and specific chunks you inherit from your father.

- Crossover probability - To potentially improve the system we have also tried applying a textitcrossover probability. This means that the parents passed on from old generation will not necessarily always crossover, it is dependant on a probability. Introducing another probability factor which might help the solution converge faster. This is because it is beneficial to sometimes keep old generation survivor parents that were actually quite good as they were, instead of always mixing them up and perhaps generating even worse new generations.

- Stop Condition We have two different stop conditions, one dependant only on a maximum number of generations and that stops if the solution is stalling. The stalling condition is basically where we check if the solution has become any better for X number of generations. X is a parameter that we can change depending on resolution, population size and the nature of the objective function. For the example functions that we treated in this report, a value of 50 were adequate. Lower than that and we might not break out of a local optima, higher than that would in most cases not manage to break the stalling.