Optimizing Beam – Assignment 5

Author: Jillian O'Connell

Purpose: The purpose of this assignment was to use the genetic algorithm to minimize both the Cross Section Area and the Static Deflection. When one is minimized, by the nature of the functions, the other is maximized. Because of those, both functions must be observed separately and then observed together using a weighted function.

Function 1:

In order to have the code more efficiently, R was used to obtain the results whereas google colab was used to write and save the code. As seen in Figure 1, all of the x values are towards the lower end of their allotted ranges to minimize the function. A fitness value of -85.36 was produced through the specific run of the genetic algorithm whose results are shown in Figure 1 and Figure 2. In Figure 2, the graph of the results is shown. On the graph, the best fitness value of each particle in each generation is shown by the green dotted line whereas the average fitness value of each particle in each generation is shown with the blue dotted line.

```
── Genetic Algorithm ──────────────────

GA settings:
Type                  =  real-valued
Population size       =  50
Number of generations =  100
Elitism               =  2
Crossover probability =  0.75
Mutation probability  =  0.001
Search domain =
      x1 x2  x3  x4
lower 10 10 0.9 0.9
upper 80 50 5.0 5.0

GA results:
Iterations             = 100
Fitness function value = -85.35987
Solution =
           x1       x2       x3       x4
[1,] 19.75594 18.72456 1.759355 1.491352
```

Figure 1: The results obtained when only function 1 (the cross section area) was minimized.

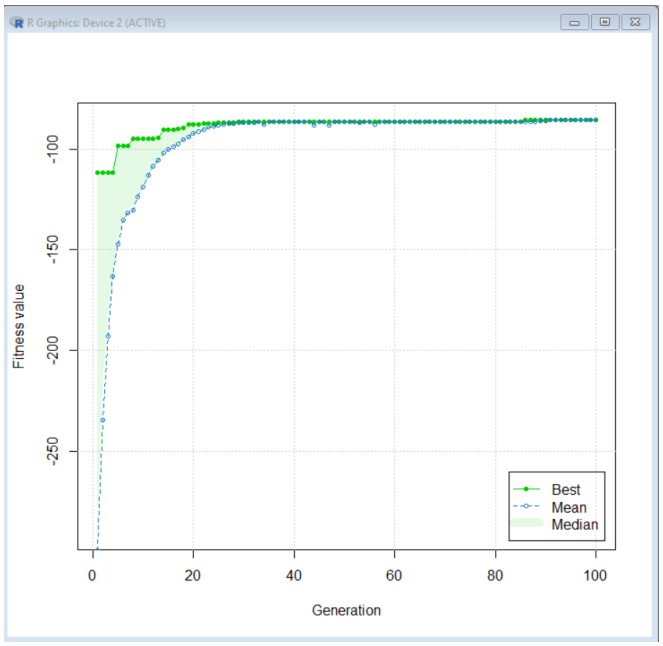


Figure 2: The graph obtained when only function 1 (cross section area) was minimized.

Function 2:

In order to have the code more efficiently, R was used to obtain the results whereas google colab was used to write and save the code. As seen in Figure 3, all of the x values are towards the higher end of the ranges allotted. With this run of the genetic algorithm, a fitness value of -0.0098 was obtained. In Figure 4, the graph obtained from this run of the genetic algorithm is shown. On the graph, the best fitness value of each particle in each generation is shown by the green dotted line whereas the average fitness value of each particle in each generation is shown with the blue dotted line.

```
── Genetic Algorithm ──────────────────

GA settings:
Type                  =  real-valued
Population size       =  50
Number of generations =  100
Elitism               =  2
Crossover probability =  0.75
Mutation probability  =  0.001
Search domain =
      x1 x2  x3  x4
lower 10 10 0.9 0.9
upper 80 50 5.0 5.0

GA results:
Iterations             = 100
Fitness function value = -0.009867034
Solution =
           x1       x2      x3      x4
[1,] 73.13178 44.83932 3.72649 3.90698
>
```

Figure 3: The results obtained when only function 2 (static deflection) was minimized.

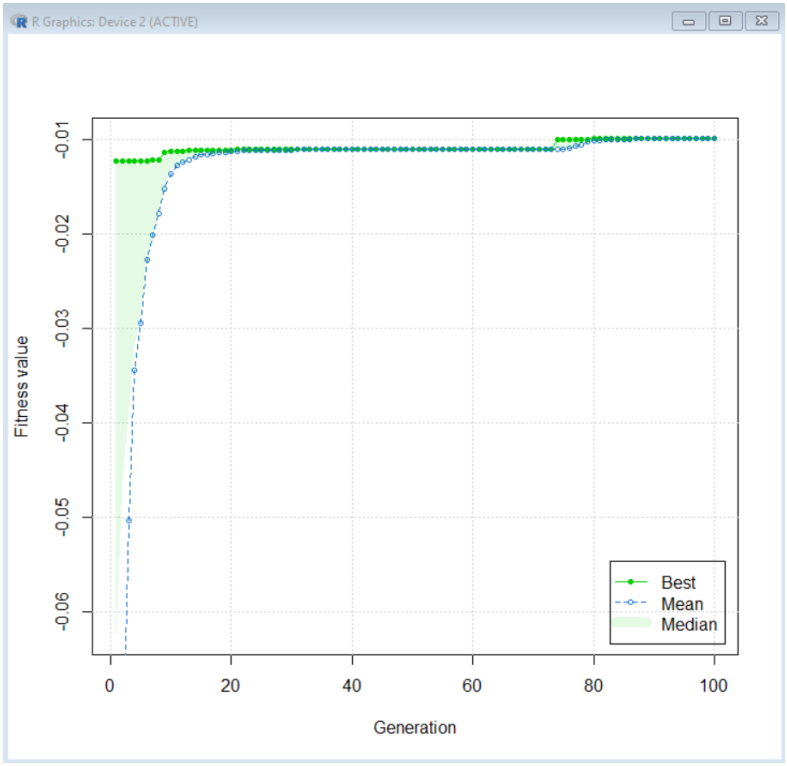


Figure 4: The graph obtained when only function 2 (static deflection) was minimized.

Weighted Function:

Because when the cross section area was minimized on its own the x values were on the low end of the allotted ranges, and, when the static deflection was minimized on its own, the x values were on the higher end of the allotted ranges, it is clear that when one function is maximized, the other is minimized. Because of this, the functions must be minimized together using a weighted function. In order to observe how the genetic algorithm of this weighted function as a whole was minimized, the genetic algorithm was run once using an extra parameter 'a'. To represent the b value in the function, '1-a' is used as the coefficient of function 2. This is because both a and b can be in the range 0 to 1, but, when summed up, a + b must not exceed 1. Using '1-a' as b makes this simpler in the code.

As seen in Figure 5, the fitness value of the weighted function was found to be -6.68. Some of the x values fell at the top of the range, some at the bottom of the range, and some in the middle of the range. X5, in this summary, represents the value of a. Because a is 0.52, it can be assumed that b is 0.48 because b is equivalent to 1-a. This means that both functions were given roughly equal weight to obtain the best fitness value when the genetic algorithm was run for the weighted function. Function 1 (cross section area) was given a weight of 52% whereas function 2 (static deflection) was given a weight of 48%. Figure 6 shows the graph obtained during this run.

```
── Genetic Algorithm ─────────────────

GA settings:
Type                  =  real-valued
Population size       =  50
Number of generations =  100
Elitism               =  2
Crossover probability =  0.75
Mutation probability  =  0.001
Search domain =
      x1 x2  x3  x4 x5
lower 10 10 0.9 0.9  0
upper 80 50 5.0 5.0  1

GA results:
Iterations             = 100
Fitness function value = -6.617758
Solution =
          x1       x2       x3       x4        x5
[1,] 13.39608 44.94214 1.227832 4.240702 0.5160001
```

Figure 5: The summary obtained when the genetic algorithm of the weighted function as run where x5 represents a.

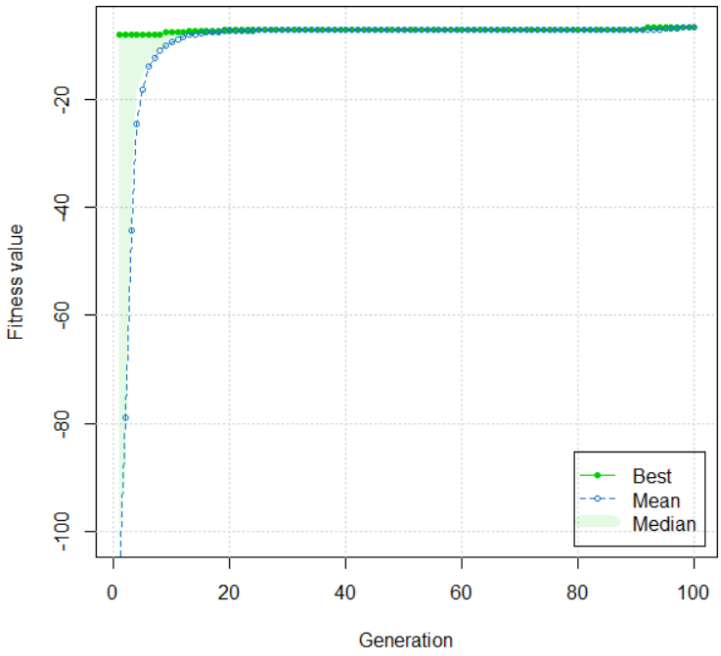


Figure 6: The graph obtained from the genetic algorithm of the weighted function.

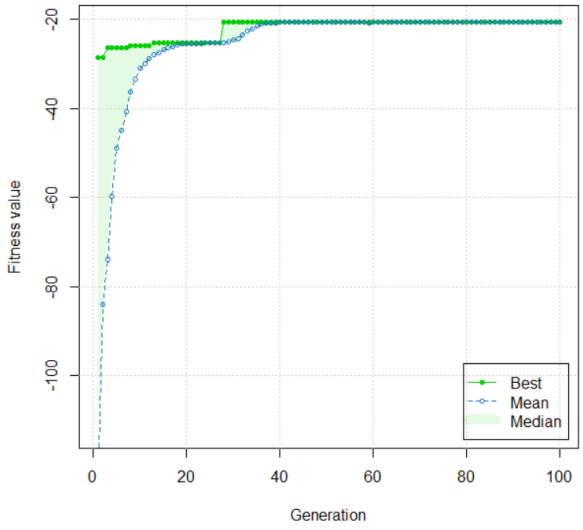
Experimentation:

After running the minimization genetic algorithm on each function separately and the combined weighted function, some experimentation was done to see if the roughly 50/50 split that was obtained when running the genetic algorithm on the weighted function truly gives the best fitness value. The results from the different experiments performed along with their a and b values are shown below with the analysis at the end.

```
── Genetic Algorithm ──────────────────

GA settings:
Type                  =  real-valued
Population size       =  50
Number of generations =  100
Elitism               =  2
Crossover probability =  0.75
Mutation probability  =  0.001
Search domain =
      x1 x2  x3  x4
lower 10 10 0.9 0.9
upper 80 50 5.0 5.0

GA results:
Iterations             = 100
Fitness function value = -20.60801
Solution =
           x1       x2       x3       x4
[1,] 15.26959 15.39786 1.031711 3.660814
```

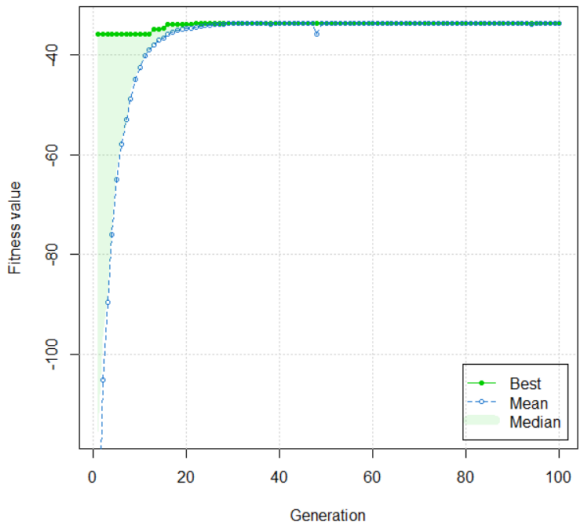


Experiment 1: Summary and graph produced from the genetic algorithm run on the weighted function where a = 0.1 and b = 0.9 giving function 1 (the cross section area) a weight of 10% and function 2 (the static deflection) a weight of 90%

```
── Genetic Algorithm ──────────────────

GA settings:
Type                  =  real-valued
Population size       =  50
Number of generations =  100
Elitism               =  2
Crossover probability =  0.75
Mutation probability  =  0.001
Search domain =
      x1 x2  x3  x4
lower 10 10 0.9 0.9
upper 80 50 5.0 5.0

GA results:
Iterations             = 100
Fitness function value = -33.73392
Solution =
           x1       x2       x3       x4
[1,] 14.11264 21.71178 1.846558 1.208625
```

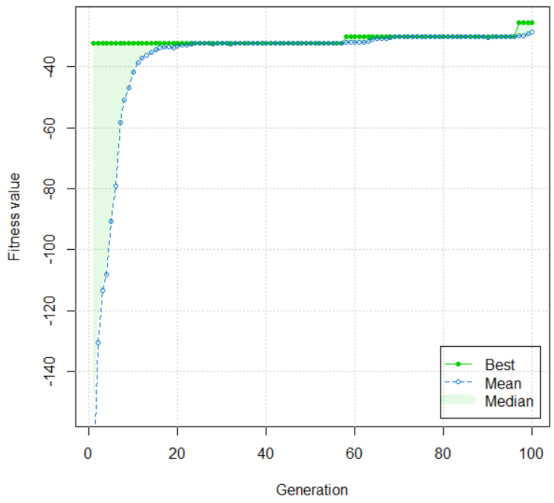


Experiment 2: Summary and graph produced from the genetic algorithm run on the weighted function where a = 0.2 and b = 0.8 giving function 1 (the cross section area) a weight of 20% and function 2 (the static deflection) a weight of 80%

```
── Genetic Algorithm ──────────────────

GA settings:
Type                  =  real-valued
Population size       =  50
Number of generations =  100
Elitism               =  2
Crossover probability =  0.75
Mutation probability  =  0.001
Search domain =
      x1 x2  x3  x4
lower 10 10 0.9 0.9
upper 80 50 5.0 5.0

GA results:
Iterations             = 100
Fitness function value = -25.55226
Solution =
           x1       x2      x3       x4
[1,] 10.68752 10.32075 1.42488 1.727873
```

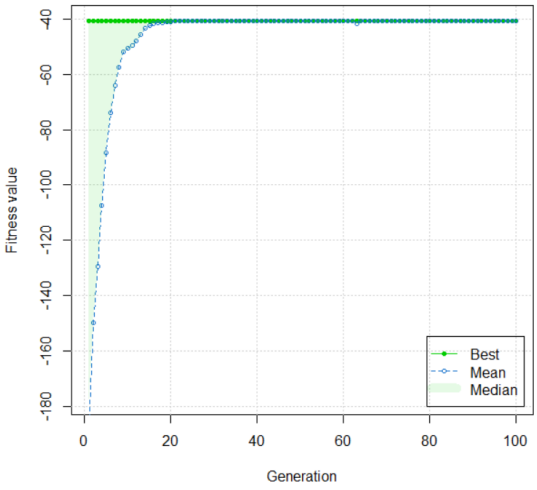


Experiment 3: Summary and graph produced from the genetic algorithm run on the weighted function where a = 0.3 and b = 0.7 giving function 1 (the cross section area) a weight of 30% and function 2 (the static deflection) a weight of 70%

```
── Genetic Algorithm ──────────────────

GA settings:
Type                  =  real-valued
Population size       =  50
Number of generations =  100
Elitism               =  2
Crossover probability =  0.75
Mutation probability  =  0.001
Search domain =
      x1 x2  x3  x4
lower 10 10 0.9 0.9
upper 80 50 5.0 5.0

GA results:
Iterations             = 100
Fitness function value = -40.53436
Solution =
           x1       x2      x3       x4
[1,] 17.4787 19.7094 1.38039 1.195994
```

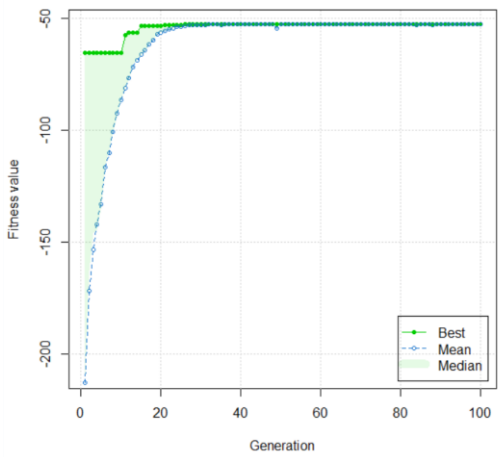


Experiment 4: Summary and graph produced from the genetic algorithm run on the weighted function where a = 0.4 and b = 0.6 giving function 1 (the cross section area) a weight of 40% and function 2 (the static deflection) a weight of 60%

```
── Genetic Algorithm ──────────────────

GA settings:
Type                  =  real-valued
Population size       =  50
Number of generations =  100
Elitism               =  2
Crossover probability =  0.75
Mutation probability  =  0.001
Search domain =
      x1 x2  x3  x4
lower 10 10 0.9 0.9
upper 80 50 5.0 5.0

GA results:
Iterations             = 100
Fitness function value = -52.43202
Solution =
           x1       x2       x3      x4
[1,] 27.39203 15.93041 1.074857 1.65005
```

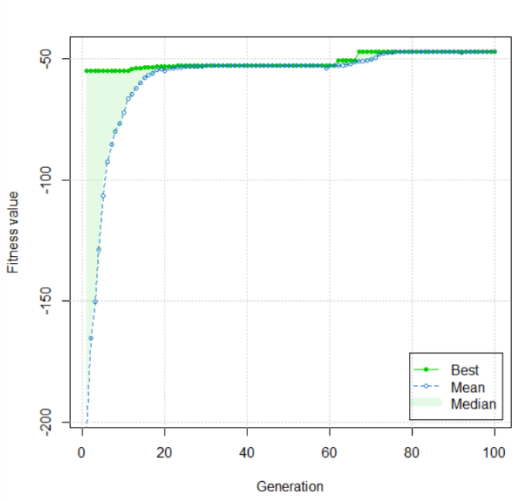


Experiment 5: Summary and graph produced from the genetic algorithm run on the weighted function where a = 0.5 and b = 0.5 giving function 1 (the cross section area) a weight of 50% and function 2 (the static deflection) a weight of 50%

```
── Genetic Algorithm ──────────────────

GA settings:
Type                  =  real-valued
Population size       =  50
Number of generations =  100
Elitism               =  2
Crossover probability =  0.75
Mutation probability  =  0.001
Search domain =
      x1 x2  x3  x4
lower 10 10 0.9 0.9
upper 80 50 5.0 5.0

GA results:
Iterations             = 100
Fitness function value = -46.92177
Solution =
           x1       x2       x3       x4
[1,] 14.74603 14.2563 2.180338 1.071324
```

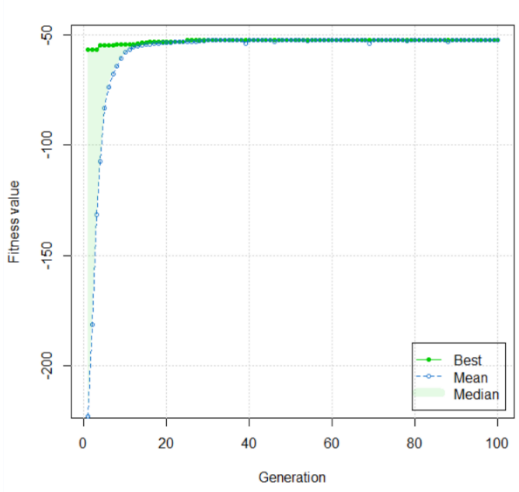


Experiment 6: Summary and graph produced from the genetic algorithm run on the weighted function where a = 0.6 and b = 0.4 giving function 1 (the cross section area) a weight of 60% and function 2 (the static deflection) a weight of 40%

```
── Genetic Algorithm ──────────────────

GA settings:
Type                  =  real-valued
Population size       =  50
Number of generations =  100
Elitism               =  2
Crossover probability =  0.75
Mutation probability  =  0.001
Search domain =
      x1 x2  x3  x4
lower 10 10 0.9 0.9
upper 80 50 5.0 5.0

GA results:
Iterations             = 100
Fitness function value = -52.3766
Solution =
           x1       x2       x3       x4
[1,] 20.49077 15.28022 1.014688 1.615204
```

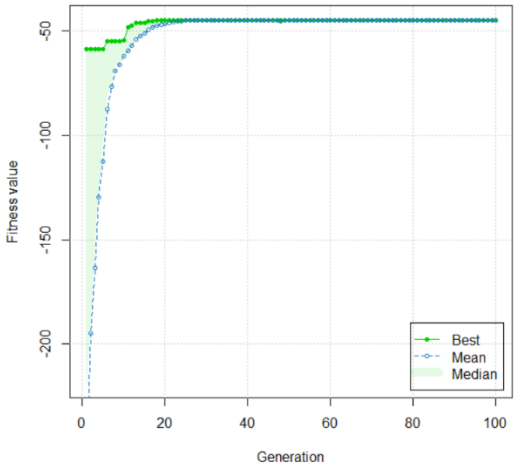


Experiment 7: Summary and graph produced from the genetic algorithm run on the weighted function where a = 0.7 and b = 0.3 giving function 1 (the cross section area) a weight of 70% and function 2 (the static deflection) a weight of 30%

```
── Genetic Algorithm ──────────────────

GA settings:
Type                  =  real-valued
Population size       =  50
Number of generations =  100
Elitism               =  2
Crossover probability =  0.75
Mutation probability  =  0.001
Search domain =
      x1 x2  x3  x4
lower 10 10 0.9 0.9
upper 80 50 5.0 5.0

GA results:
Iterations             = 100
Fitness function value = -44.72225
Solution =
           x1       x2       x3       x4
[1,] 14.61113 12.67608 1.453733 1.317428
```

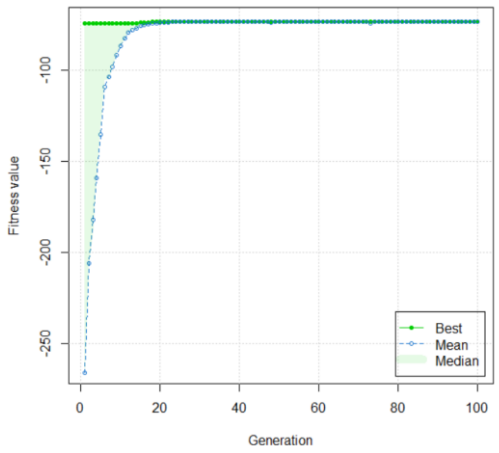


Experiment 8: Summary and graph produced from the genetic algorithm run on the weighted function where a = 0.8 and b = 0.2 giving function 1 (the cross section area) a weight of 80% and function 2 (the static deflection) a weight of 20%

```
── Genetic Algorithm ──────────────────

GA settings:
Type                  =  real-valued
Population size       =  50
Number of generations =  100
Elitism               =  2
Crossover probability =  0.75
Mutation probability  =  0.001
Search domain =
      x1 x2  x3  x4
lower 10 10 0.9 0.9
upper 80 50 5.0 5.0

GA results:
Iterations             = 100
Fitness function value = -73.17636
Solution =
           x1       x2       x3       x4
[1,] 11.74786 22.91231 2.177013 1.282708
```



Experiment 9: Summary and graph produced from the genetic algorithm run on the weighted function where a = 0.9 and b = 0.1 giving function 1 (the cross section area) a weight of 90% and function 2 (the static deflection) a weight of 10%

In this experiment, every fitness value was a negative number. The fitness value that was closest to zero was -20.61 when a was 0.1 and b was 0.9. This means that, objectively, the best weights to give the functions were 10% for the cross section area and 90% for the static deflection. When the genetic algorithm was run on the weighted function as a whole and tasked with finding its own a and b values, it gave values closer to a 50/50 split. In theory, the results say that the algorithm should be heavily biased towards the static deflection. In reality, when building these support beams, the 50/50 split makes more sense because going too far to one end of the spectrum ends up worse for the building as a whole. This accounts for the discrepancies between when the genetic algorithm was tasked to find the weighted values itself versus when it was manually done. When finding the weight itself, the algorithm has control over experimenting with the weights. When forced to account for weights that are hardcoded, the algorithm is only focusing on minimizing the 4 input values. It could be said that the algorithm is more informed when it is finding the weights itself because it has more information to experiment with and to use to understand the equations.