##
 # Author: Michael Wood
 # Purpose: Use the genetic algorithm to minimize the functions f1, f2, and y = a*f1 + b*f2. "f1" being the Cross Section Area and "f2" being the Static Deflection for the design of an I-beam. This is problem 1 of Assignment 5 for CSC 362.
 # Class: AI, CSC 362, Spring 2024
##

All code was done through R and displayed in Google Colab in the Problem1.ipynb file in the same folder as this report, HW5.

I started by defining the necessary functions for this problem.

- f1 = $2x_2x_4 + x_3(x_1 - 2x_4)$
- f2 = $60000 / (x_3(x_1 - 2x_4)^3 + 2x_2x_4[4x_4^2 + 3x_1(x_1 - 2x_4)])$
- F = y = a(f1) + b(f2)

Then, I defined the geometric side constraints for the x values:

- $10 \leq x_1 \leq 80$
- $10 \leq x_2 \leq 50$
- $0.9 \leq x_3 \leq 5.0$
- $0.9 \leq x_4 \leq 5.0$
- The sum of a and b must be 1

After defining the functions and parameters, I used the genetic algorithm to minimize each of the functions. In the algorithm, I multiplied the fitness functions by -1 in order to minimize them.

I also applied the following population parameters to the algorithm:

The population parameters:

1. numGenerations=100
2. populationSize=50
3. Pc=0.75
4. Pm=0.001
5. type="real-valued"
6. lower = c(10, 10, 0.9, 0.9)
7. upper = c(80, 50, 5.0, 5.0)

Below are the following results and observations of each function the algorithm was applied to:

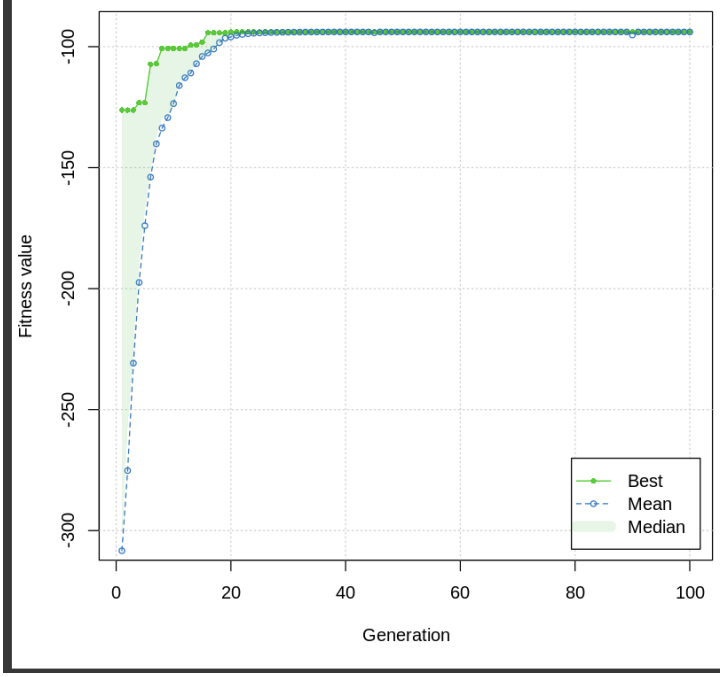
##
 # FUNCTION 1 - f1
##

Figure 1: The result of the genetic algorithm to minimize the Cross Section Area function only.

```
── 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 = -93.91707
Solution =
           x1       x2       x3       x4
[1,] 29.75616 13.53837 1.753486 1.770916
```



In figure 1, a fitness value of -93.92 was produced in this particular generation. The solution of X values are closer to their minimum values, which is to be expected for this function.

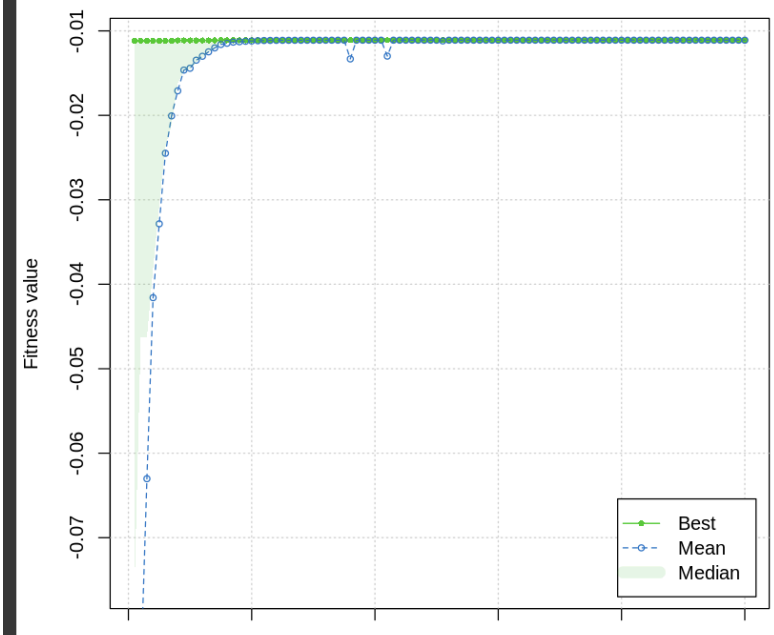
```
###############################################
# FUNCTION 2 - f2
###############################################
```

Figure 2: The result of the genetic algorithm to minimize the Static Deflection function only

```
── 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 = -0.01110412
Solution =
          x1       x2      x3       x4
[1,] 71.03904 36.84498 3.90629 4.578368
```

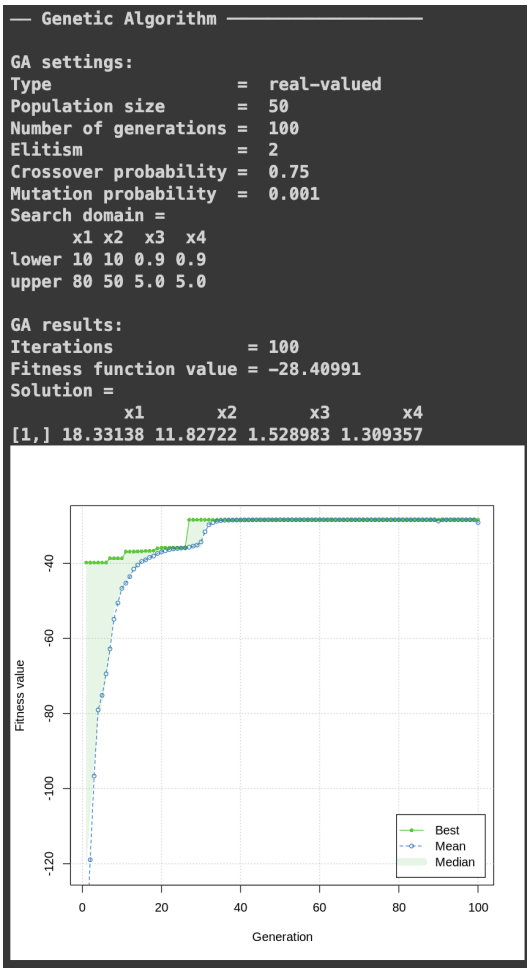


In figure 2, a fitness value of -0.01 was produced in this particular generation. The solution of X values are closer to their maximum values, which is to be expected for this function.

##
 # FUNCTION 3 - y
##

It is clear from figures 1 and 2 that the functions f1 and f2 conflict with each other as their minimum values differ dramatically. In order to find a compromise, the weighted approach is needed to combine the first two functions with the equation y = a*f1 + b*f2, where a+b=1.

Figure 3: The result of the genetic algorithm to minimize the weighted function where a, b=0.5.



In figure 3, a fitness value of -28.41 was produced in this particular generation. Giving both functions an equal weight generates a value that is now more in the middle of the previous two functions. This creates more optimal X values in the solution.

I then experimented with different weights of a and b and the results for each generation are shown below:

Figure 4: The result of the genetic algorithm to minimize the weighted function where a=1, b=0.

```
── 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 = -78.92303
Solution =
          x1       x2       x3       x4
[1,] 13.94762 15.47975 2.570214 1.668327
```

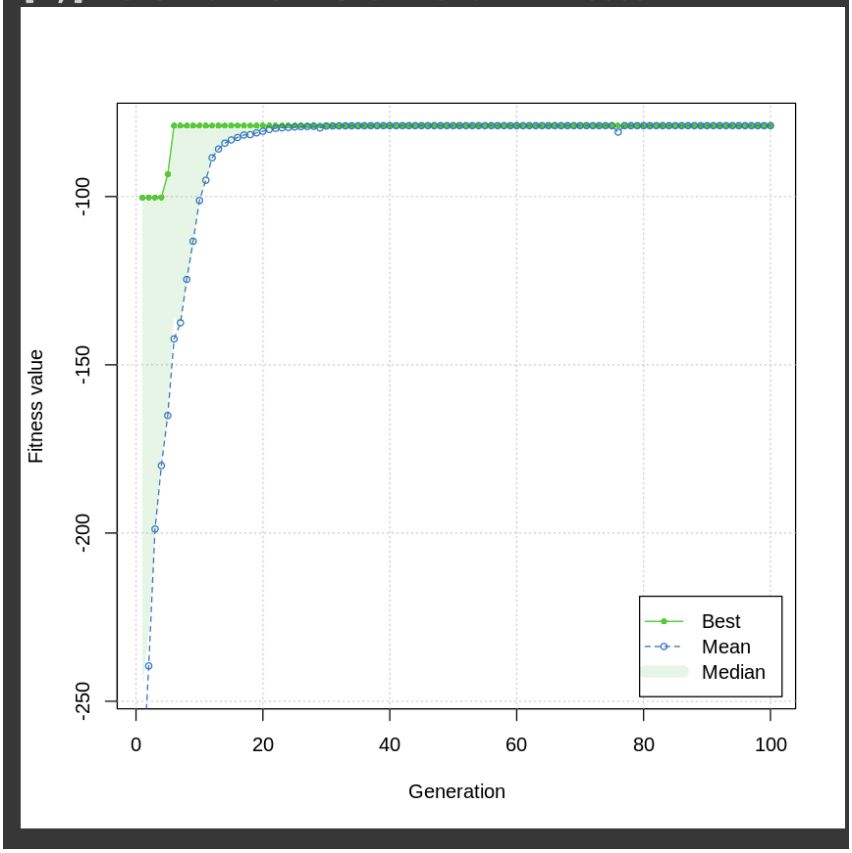


Figure 5: The result of the genetic algorithm to minimize the weighted function where a=0, b=1.

```
── 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 = -0.007498014
Solution =
           x1       x2       x3       x4
[1,] 79.43715 43.20409 3.642775 4.645453
```

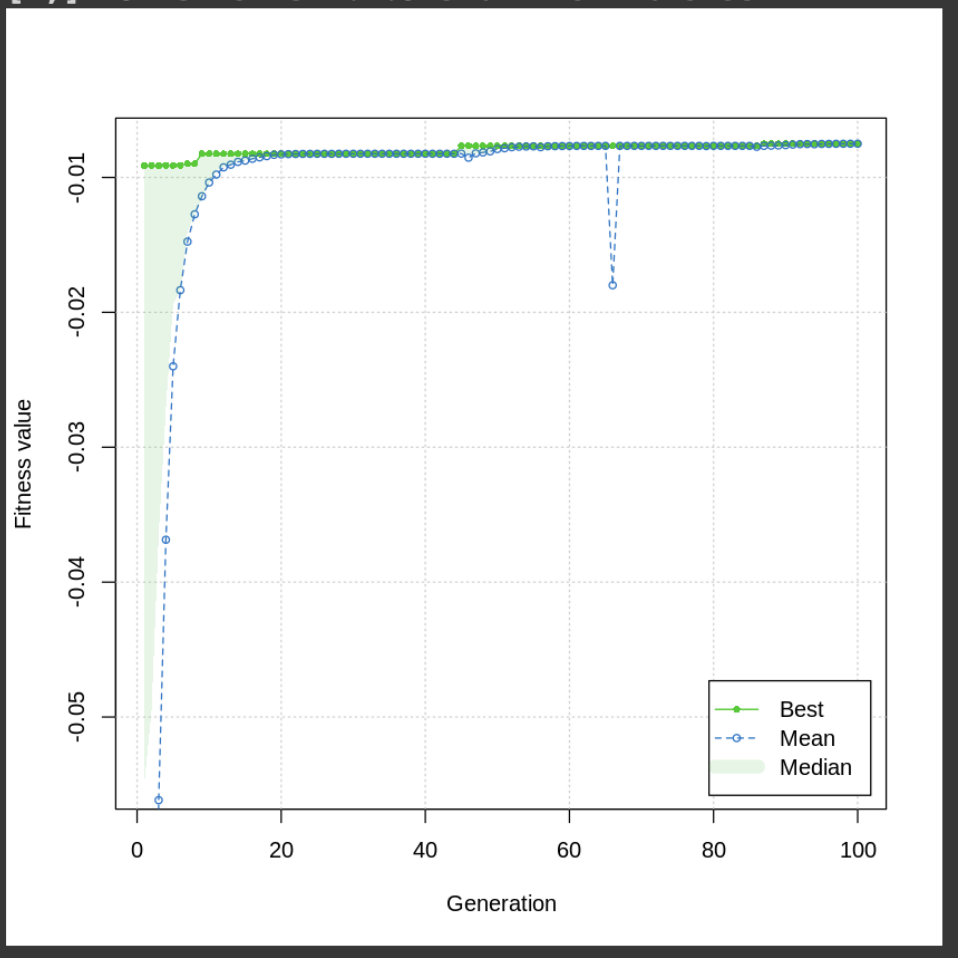


Figure 6: The result of the genetic algorithm to minimize the weighted function where a=0.1, b=0.9

```
── 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 = -8.836605
Solution =
          x1       x2       x3       x4
[1,] 19.08627 14.21021 2.355546 1.340628
```

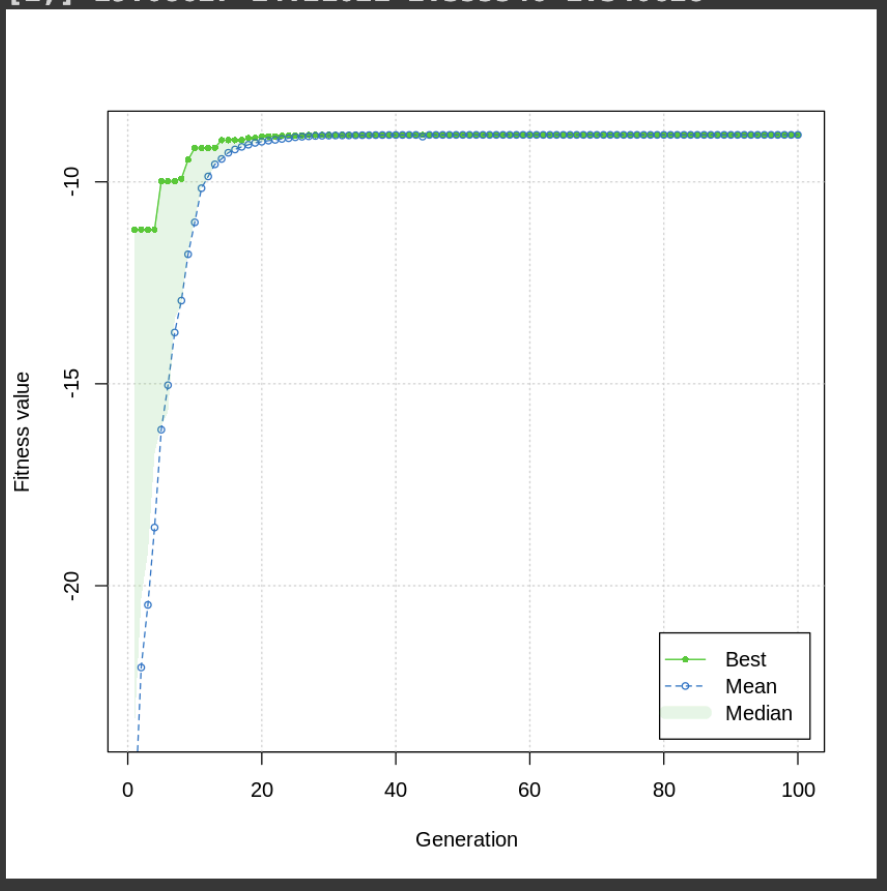


Figure 7: The result of the genetic algorithm to minimize the weighted function where a=0.2, b=0.8

```
── 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 = -14.16104
Solution =
           x1       x2       x3       x4
[1,] 17.56358 15.32713 1.272906 1.509051
```

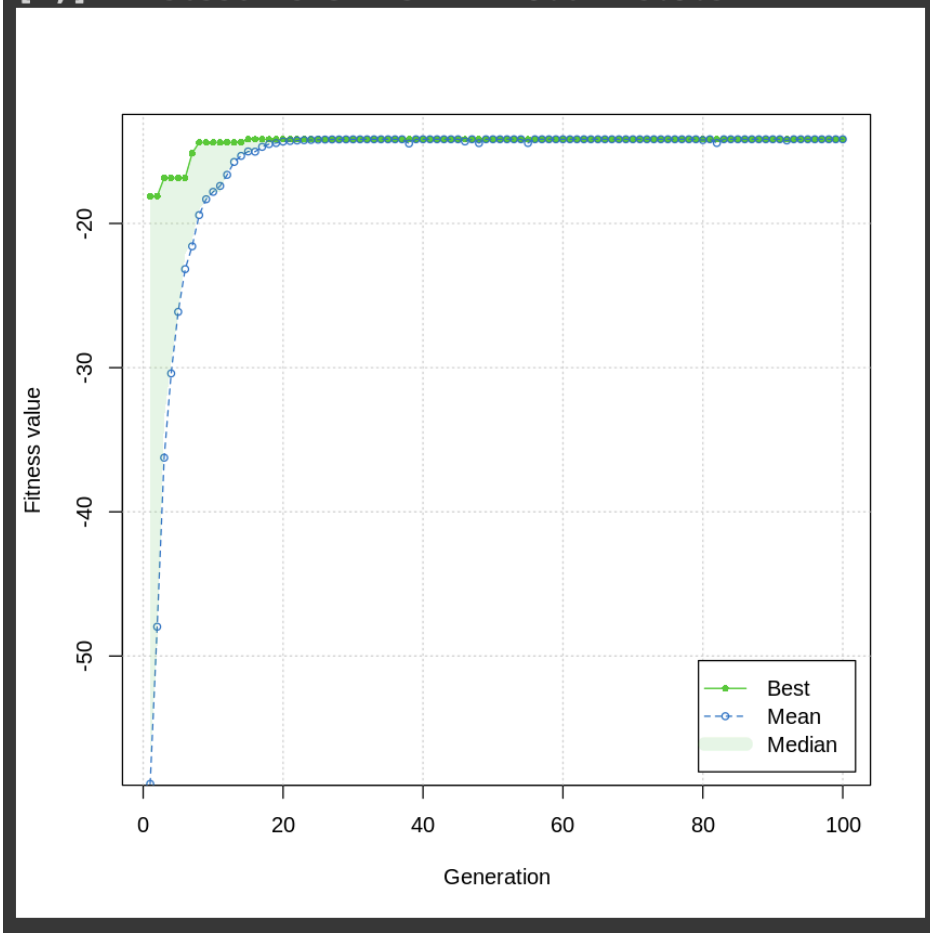


Figure 8: The result of the genetic algorithm to minimize the weighted function where a=0.3, b=0.7

```
── 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 = -16.45167
Solution =
          x1       x2       x3       x4
[1,] 12.74388 17.00054 1.064478 1.012709
```

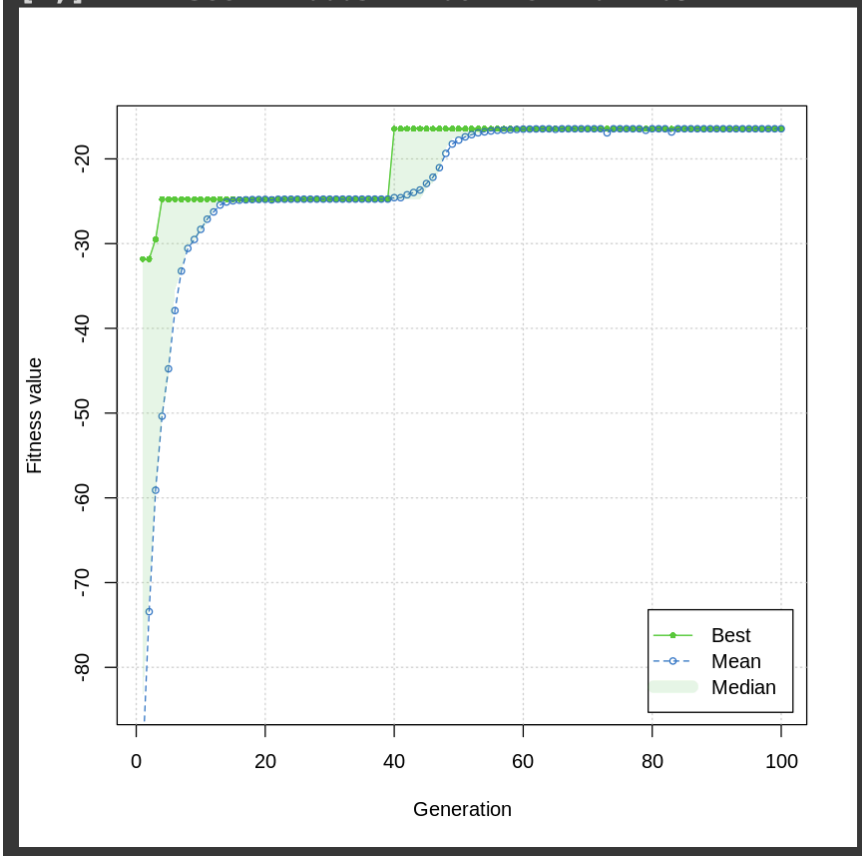


Figure 9: The result of the genetic algorithm to minimize the weighted function where a=0.4, b=0.6

```
── 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 = -29.21468
Solution =
           x1       x2       x3       x4
[1,] 18.82024 11.33308 2.323953 1.500871
```

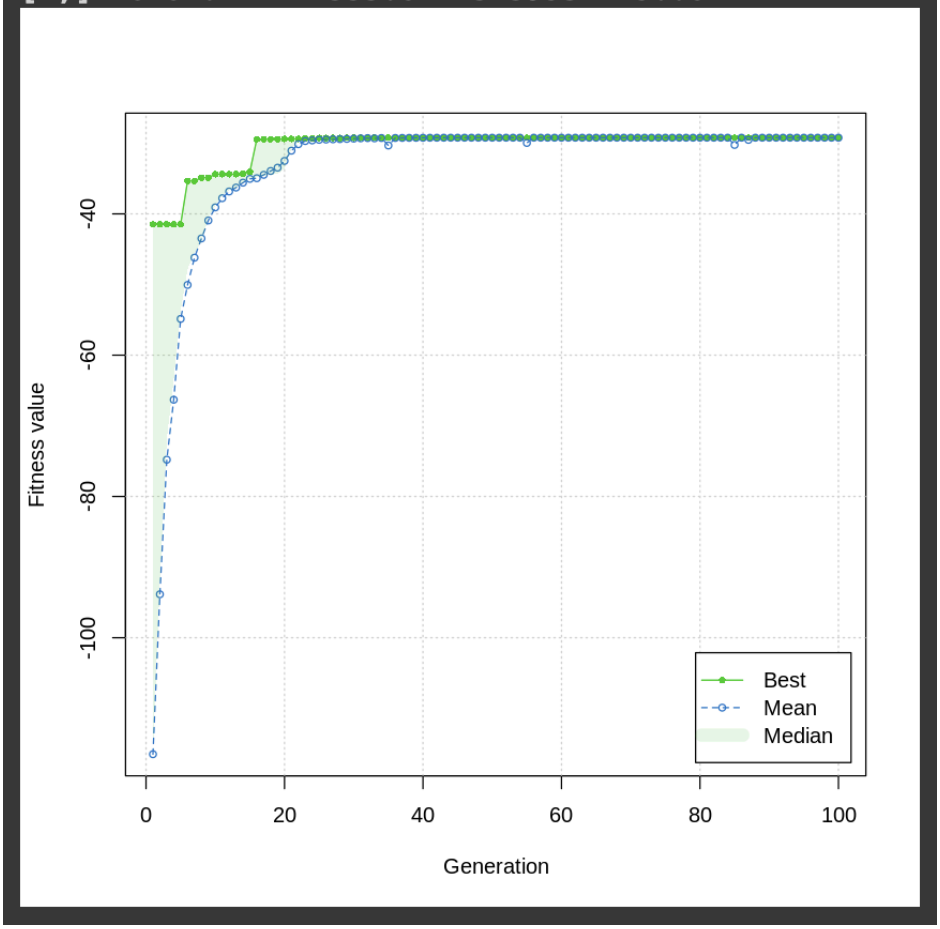


Figure 10: The result of the genetic algorithm to minimize the weighted function where a=0.6, b=0.4

```
── 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 = -39.39593
Solution =
          x1       x2       x3      x4
[1,] 20.3804 14.65394 1.653981 1.19711
```

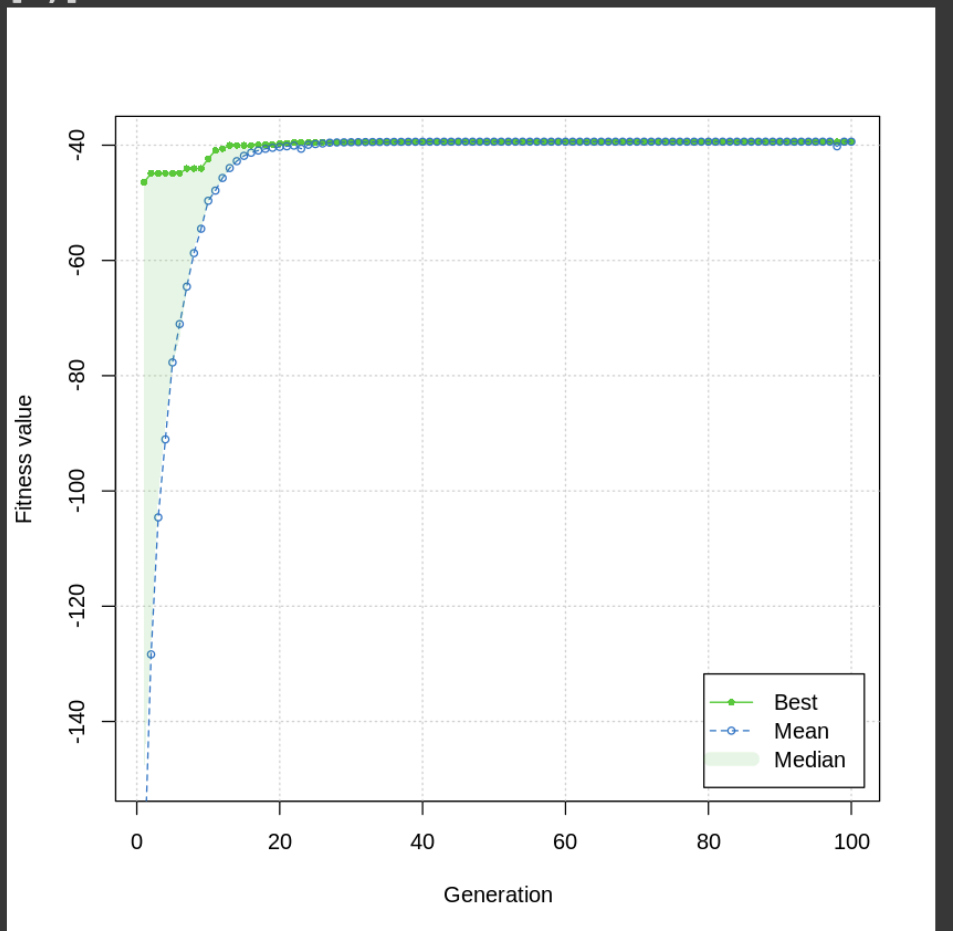


Figure 11: The result of the genetic algorithm to minimize the weighted function where a=0.7, b=0.3

```
── 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 = -47.76292
Solution =
           x1       x2       x3       x4
[1,] 32.08687 14.07736 1.156406 1.196679
```

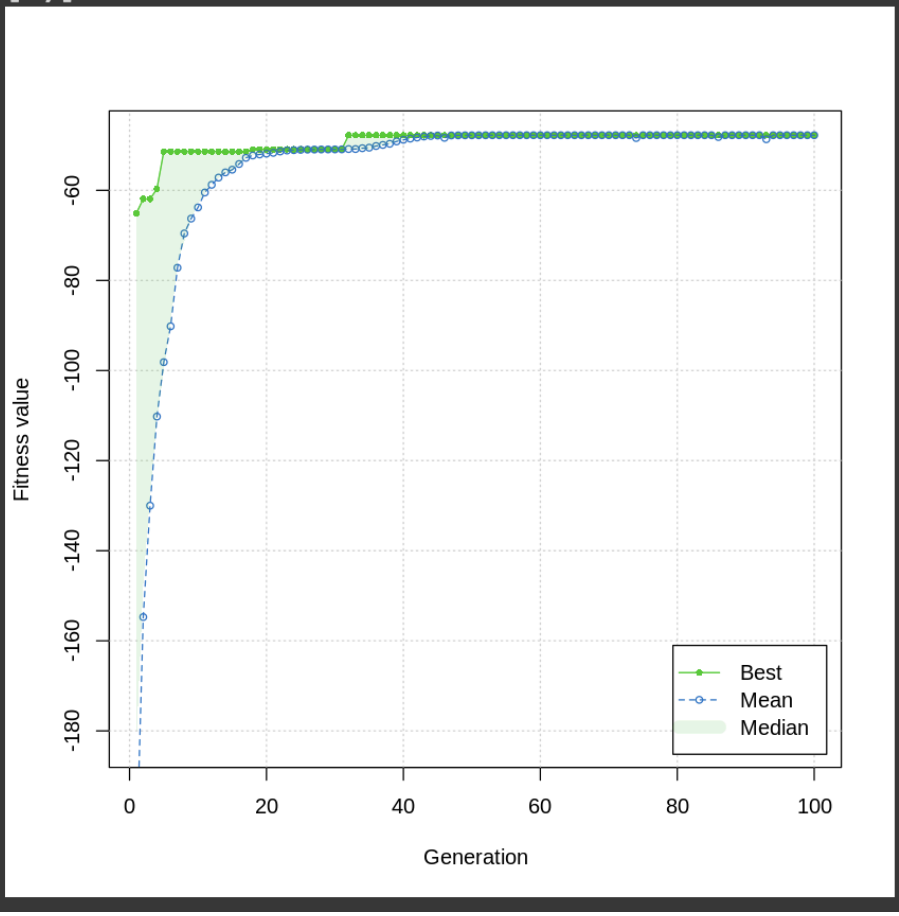


Figure 12: The result of the genetic algorithm to minimize the weighted function where a=0.8, b=0.2

```
── 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 = -50.98452
Solution =
           x1      x2      x3      x4
[1,] 19.88443 13.9845 1.91359 1.04893
```

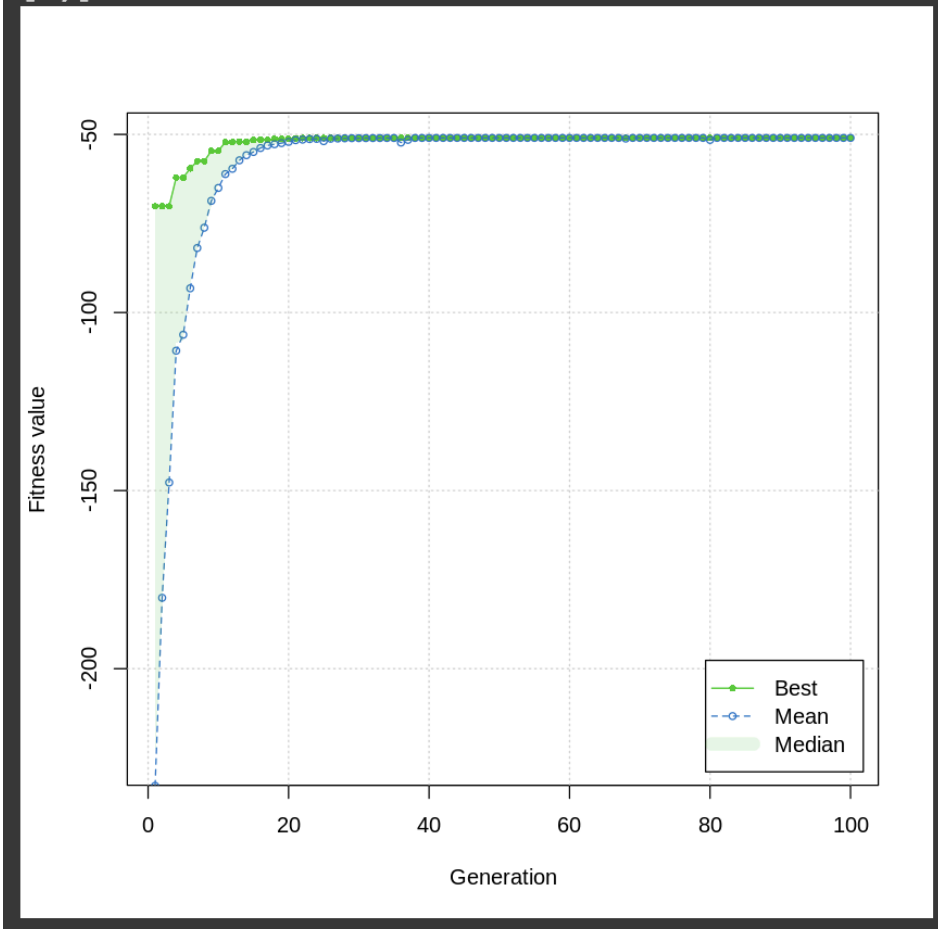
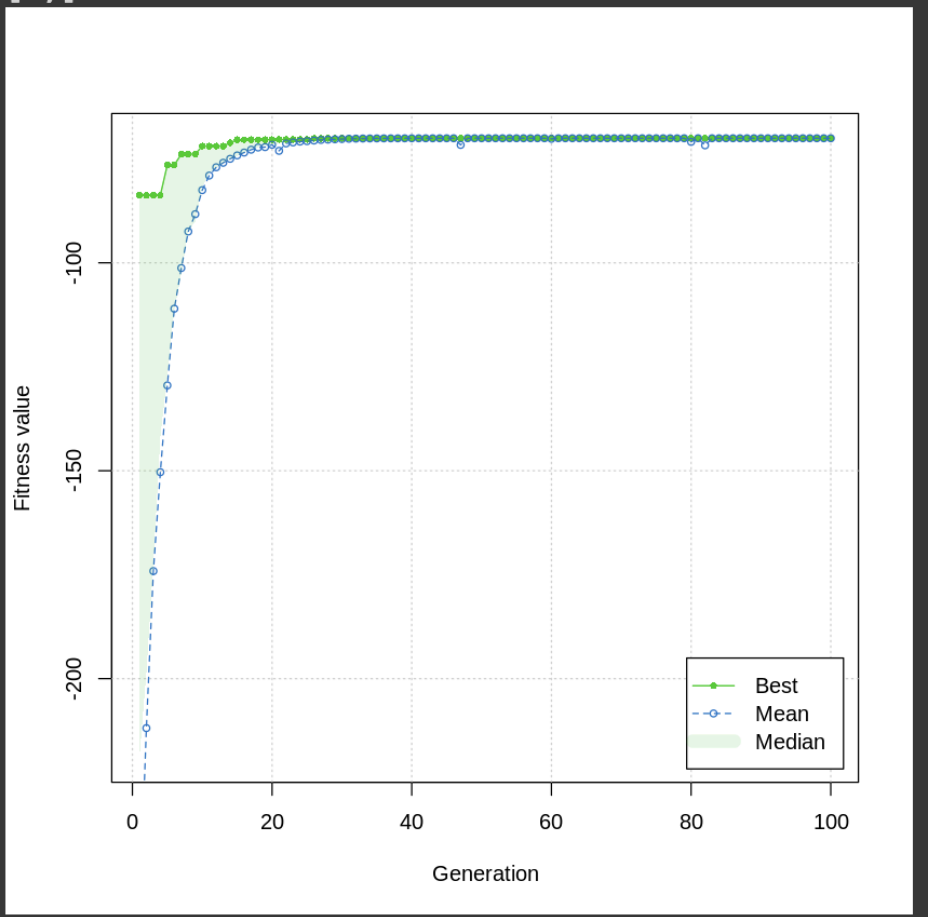


Figure 13: The result of the genetic algorithm to minimize the weighted function where a=0.9, b=0.1

```
── 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 = -70.04563
Solution =
          x1       x2       x3       x4
[1,] 17.3083 16.91795 1.488753 1.682376
```



##
 # OVERALL CONCLUSIONS
##

Looking at the results of different a and b values, we can make the following observations:

- Giving the f1 function a higher weight usually leads to lower fitness values.
    - For example, the result when a=0 and b=1 displayed a higher fitness value than even the Static Deflection function alone in this particular generation.
- Giving the f2 function a higher weight usually leads to higher fitness values.
    - None of them gave a lower fitness value than the Cross Section Area alone in this particular generation
- The point of convergence between best and average fitness behavior is lower with a higher weight given to the f1 function as well.
    - This behavior is partly expected due to the behavior of f1 and f2 alone displaying the lower and higher fitness values respectively
- There seems to be trade-offs between the X values between the different weights.
    - For example, in one generation, x_1 may be closer to its maximum compared to the other X values in the solution. However, in other generations, x_1 may get closer to its minimum, but then there would usually be an increase in x_2.
    - It is interesting to see the algorithm try to balance out the values while attempting to find the optimal solution for that particular generation. The different weights being applied to both functions could help find the optimal solution more than if the functions were separate.
    - Obviously, values x_1 and x_2 will be significantly larger than x_3 and x_4, but it is interesting to note that the algorithm usually tries to keep x_3 and x_4 as minimum as possible even when a higher weight is given to f2 in the weighted function. This results in a tradeoff between keeping those values at a minimum and possibly receiving higher x_1 and x_2 values to offset that.