Artificial Intelligence (CS F407)

Programming Assignment 1

Madhav Bajaj
2019B3A70256G

Part A:

Basic Genetic Algorithm from the textbook utilized

Parameters:

Population Size = 100
Number of Generations = 50
Probability of Mutation = 0.02

Note1:

The crossover (reproduce) function produces two children, but the textbook Genetic Algorithm only adds one child to the new population at a time. Thus, the one with higher fitness has been chosen during the implementation, and the same has been presented in the analysis.

```
function GENETIC-ALGORITHM(population, FITNESS-FN) returns an individual
  inputs: population, a set of individuals
          FITNESS-FN, a function that measures the fitness of an individual

  repeat
    new_population ← empty set
    for i = 1 to SIZE(population) do
      x ← RANDOM-SELECTION(population, FITNESS-FN)
      y ← RANDOM-SELECTION(population, FITNESS-FN)
      child ← REPRODUCE(x, y)
      if (small random probability) then child ← MUTATE(child)
      add child to new_population
    population ← new_population
  until some individual is fit enough, or enough time has elapsed
  return the best individual in population, according to FITNESS-FN

function REPRODUCE(x, y) returns an individual
  inputs: x, y, parent individuals

  n ← LENGTH(x); c ← random number from 1 to n
  return APPEND(SUBSTRING(x, 1, c), SUBSTRING(y, c + 1, n))
```

Note2:

Under the random selection function, the fitness values of all the individuals in the population are normalized to probabilities. Under my analysis, I have used two methods to do so.

Method 1:

$$P_i = \frac{FitnessFN(i)}{\sum_{i=1}^{N}(FitnessFN(i))}$$

where P_i is the probability of selection for an individual i in the population of size N

Method 2:

$$P_i = \frac{FitnessFN(i) + 1}{\sum_{i=1}^{N}(FitnessFN(i) + 1)}$$

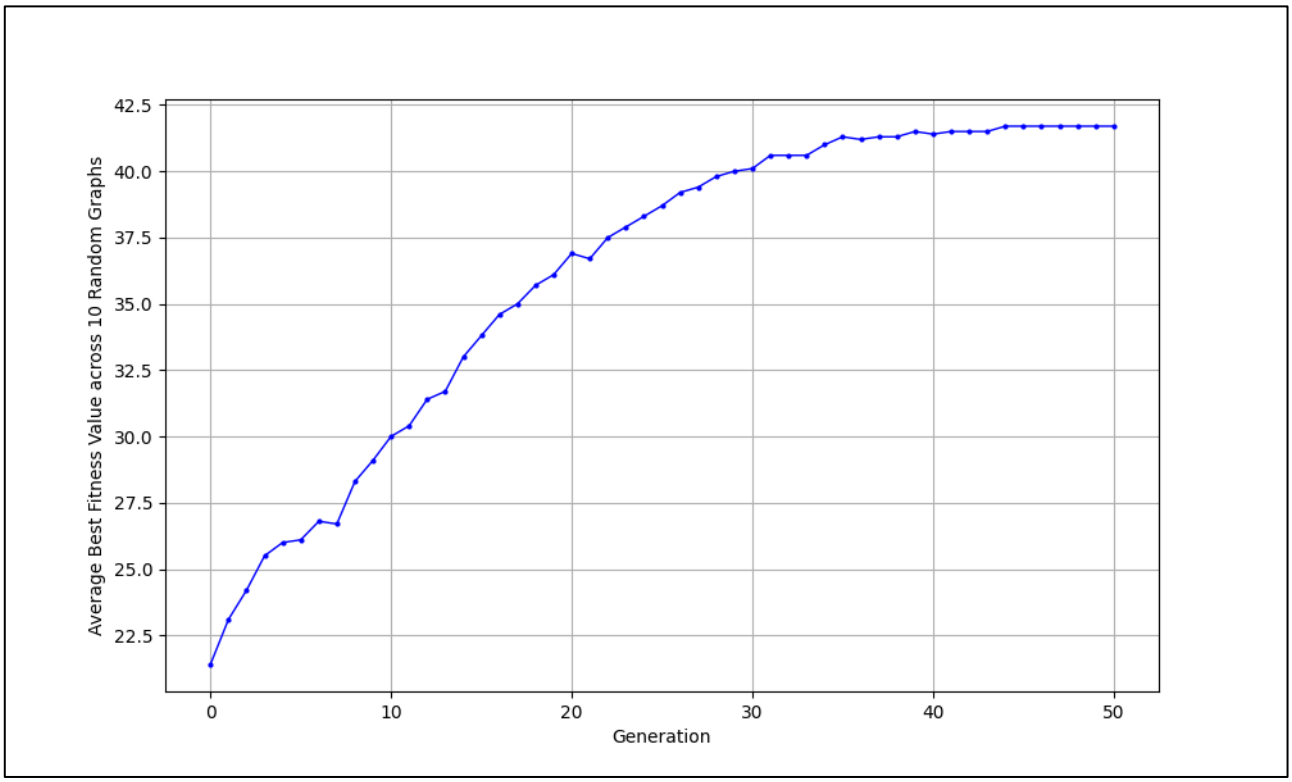
The reason for this dual approach is that in the case where the graph consists of a considerable number of edges, the sum of the fitness function of all the individuals in the population could be zero.

Results:

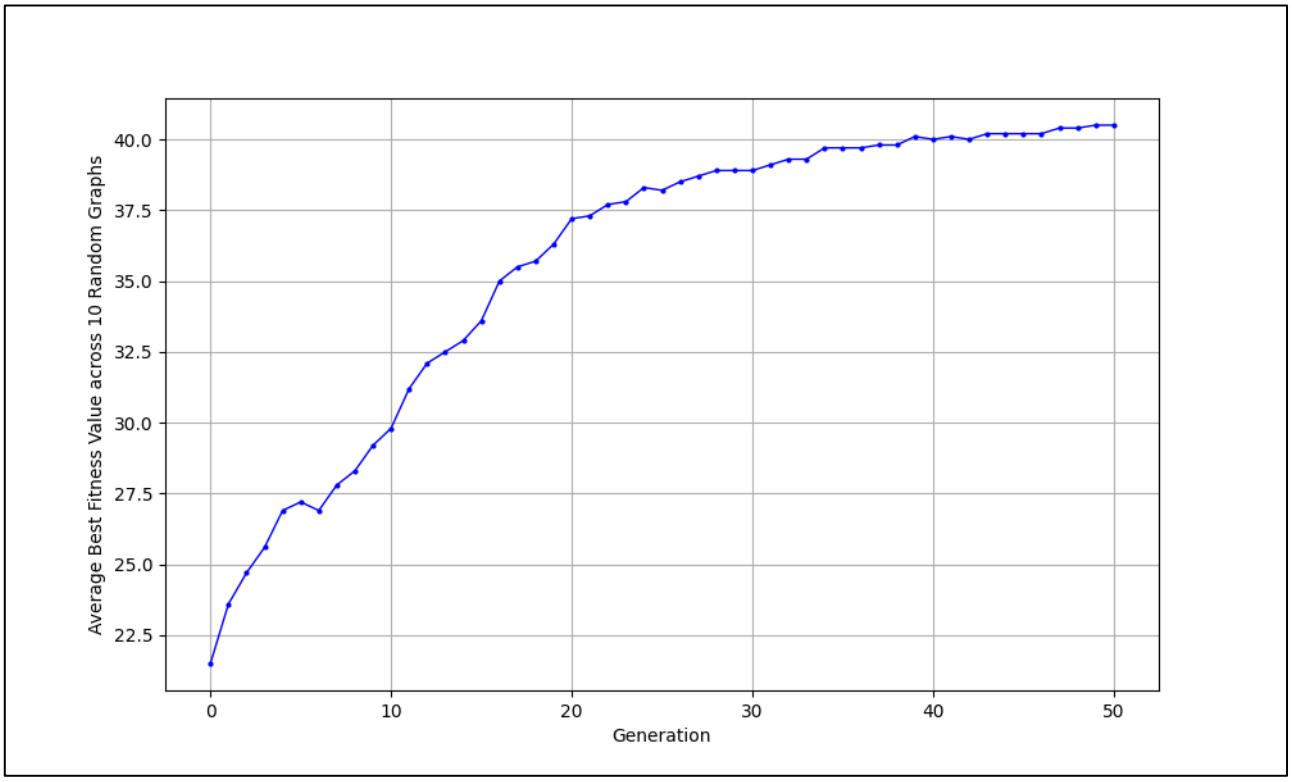
Each edge set size tested against 10 randomly generated graphs

A) Test case: 100 Edges

Graph 1: Using Method 1 Normalization, Average value of best fitness value after 50 generations = 41.7

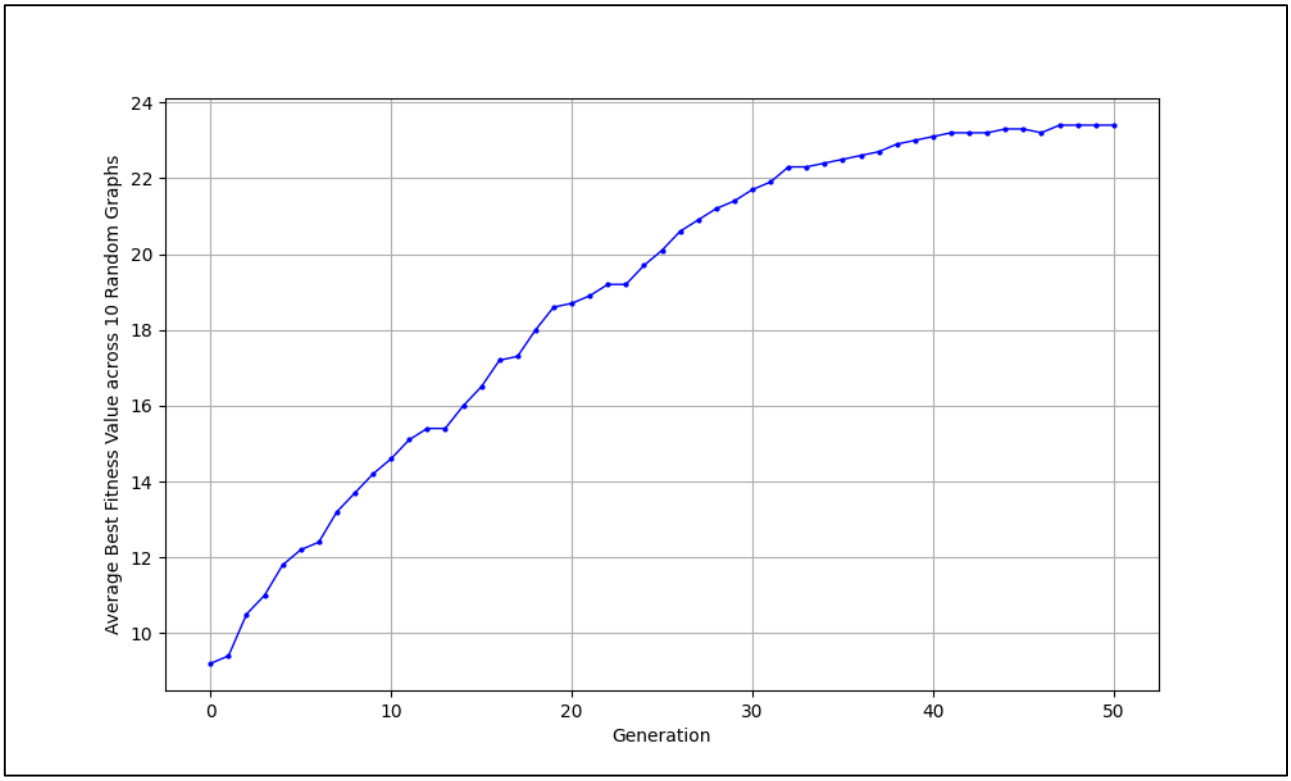


Graph 2: Using Method 2 Normalization, Average value of best fitness value after 50 generations =40.5

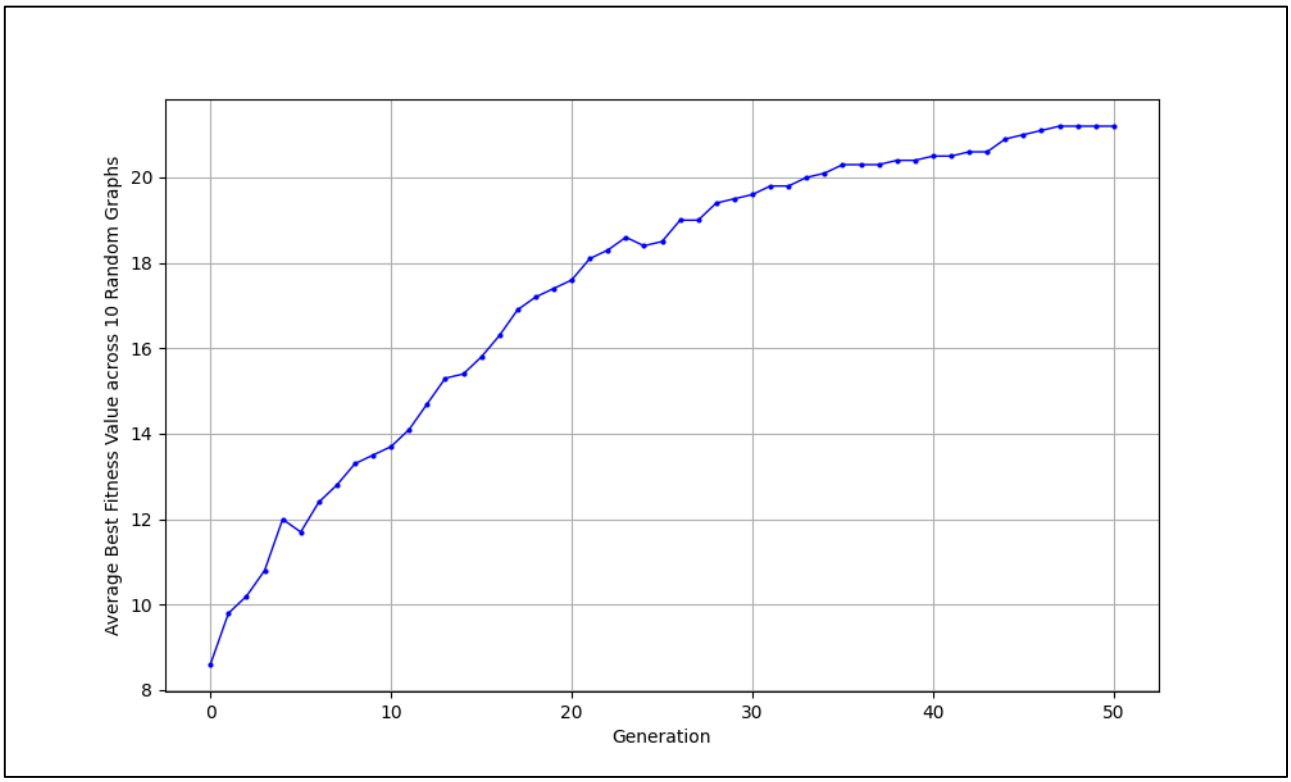


B) Test case: 200 Edges

Graph 3: Using Method 1 Normalization, Average value of best fitness value after 50 generations = 23.4

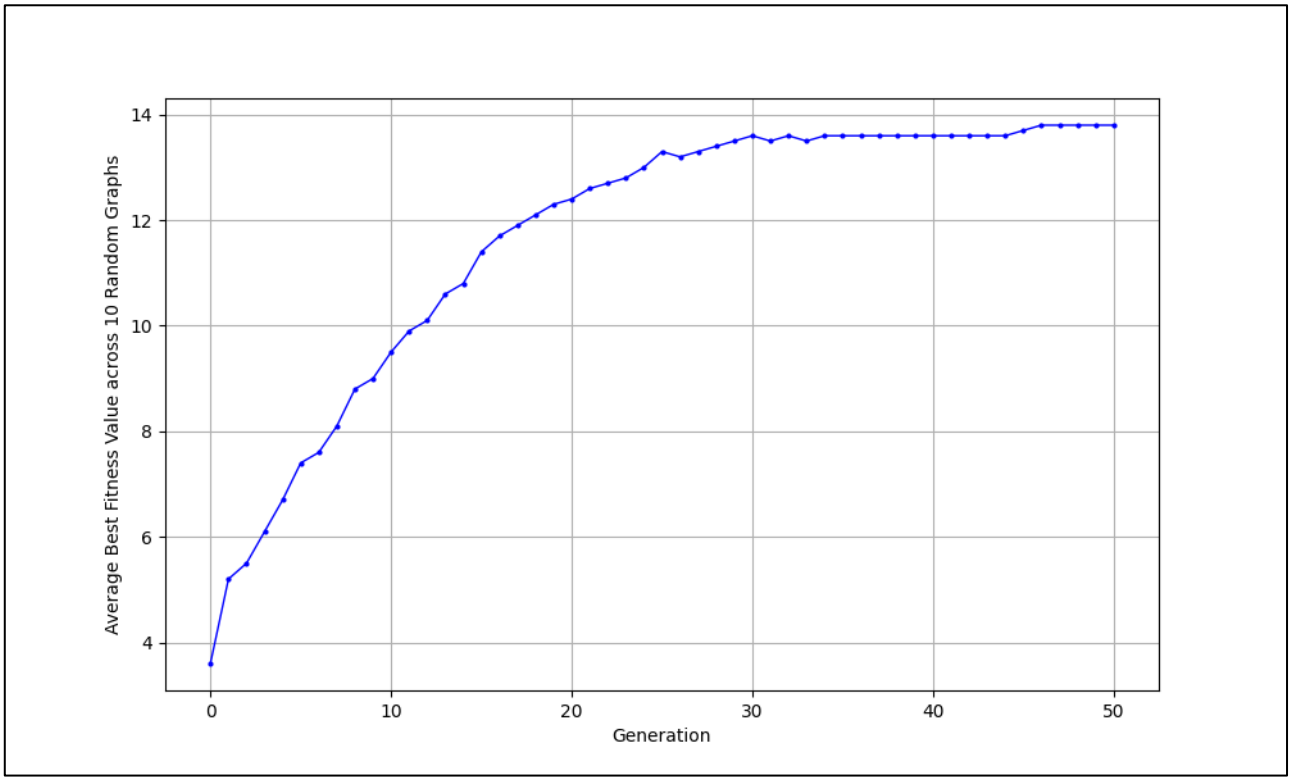


Graph 4: Using Method 2 Normalization, Average value of best fitness value after 50 generations = 21.2

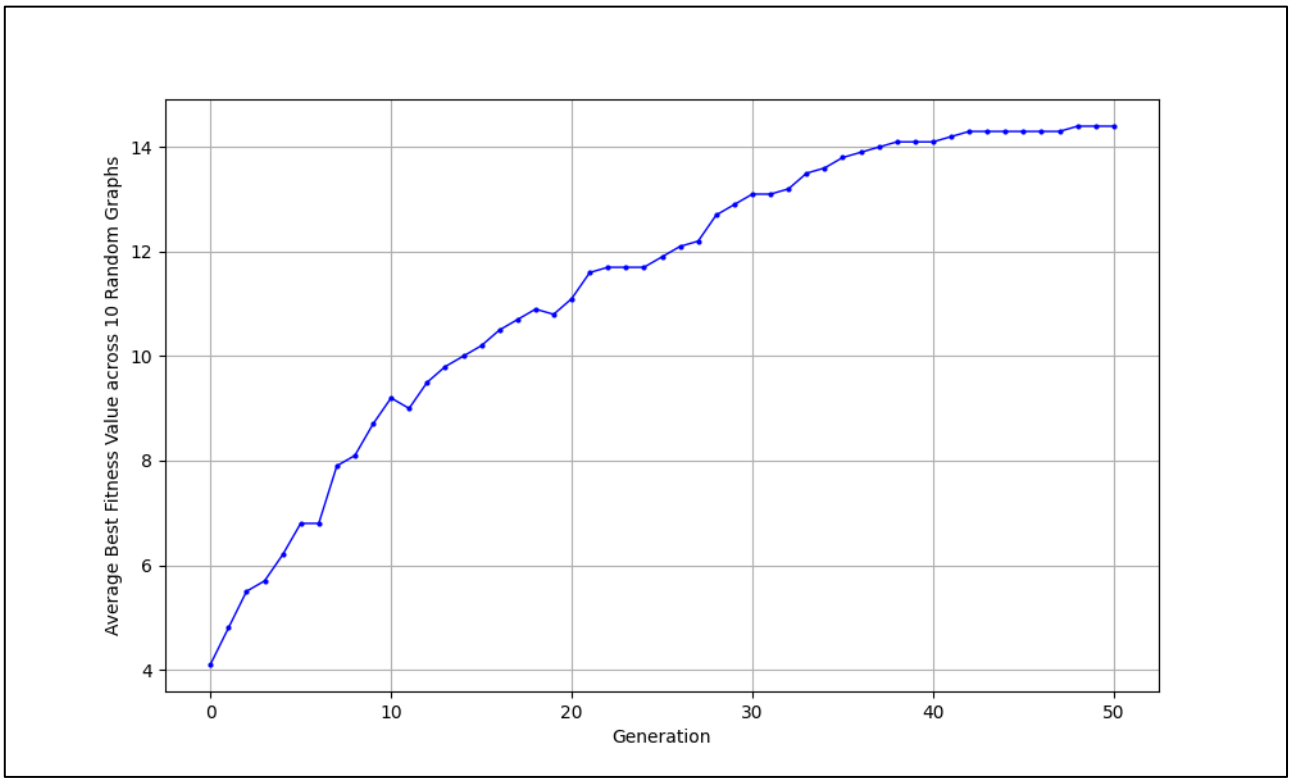


C) Test case: 300 Edges

Graph 5: Using Method 1 Normalization, Average value of best fitness value after 50 generations = 13.8

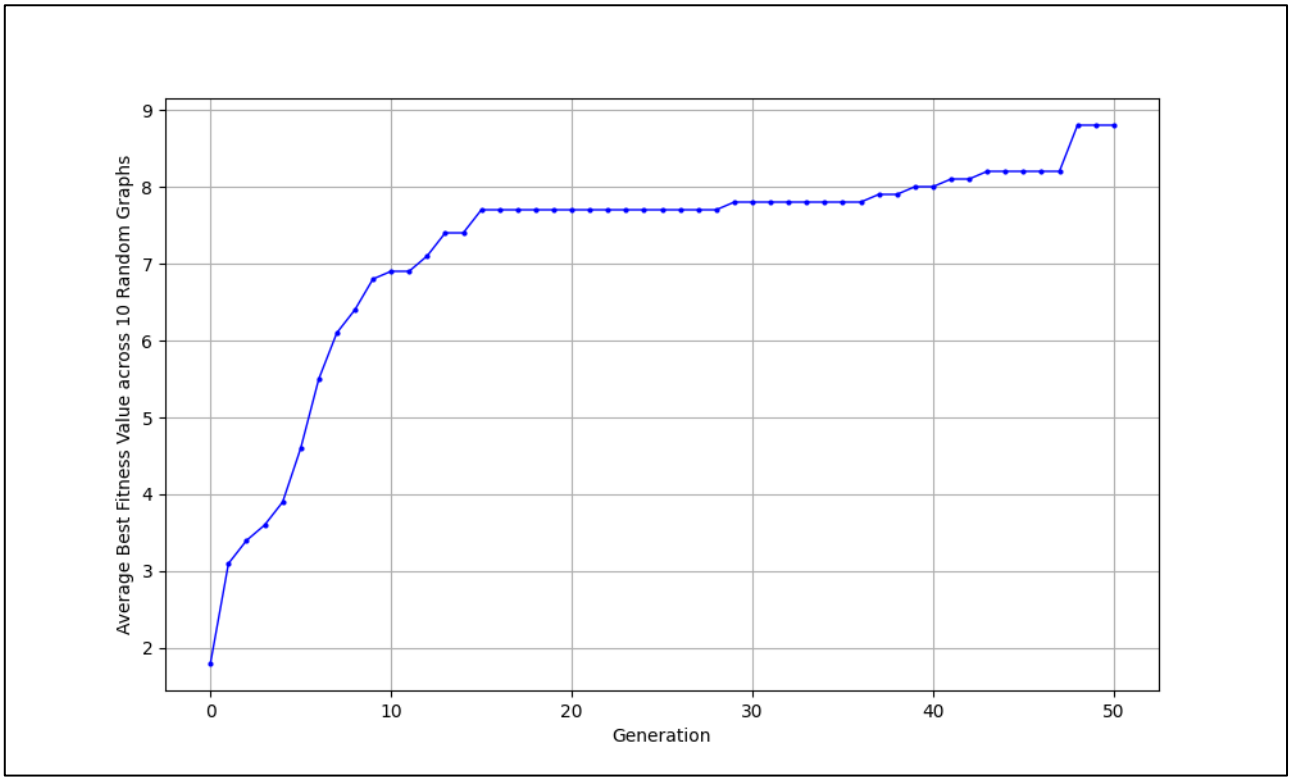


Graph 6: Using Method 2 Normalization, Average value of best fitness value after 50 generations = 14.4

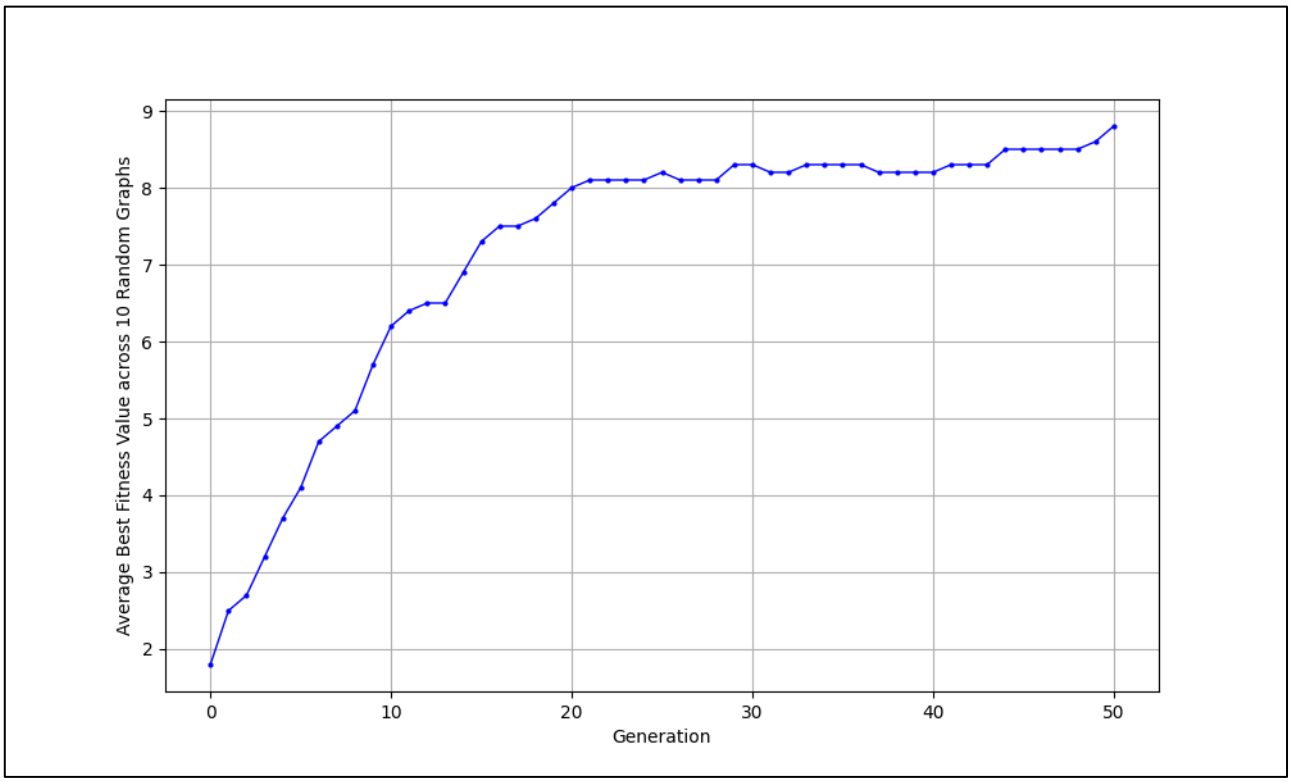


D) Test case: 400 Edges

Graph 7: Using Method 1 Normalization, Average value of best fitness value after 50 generations = 8.8



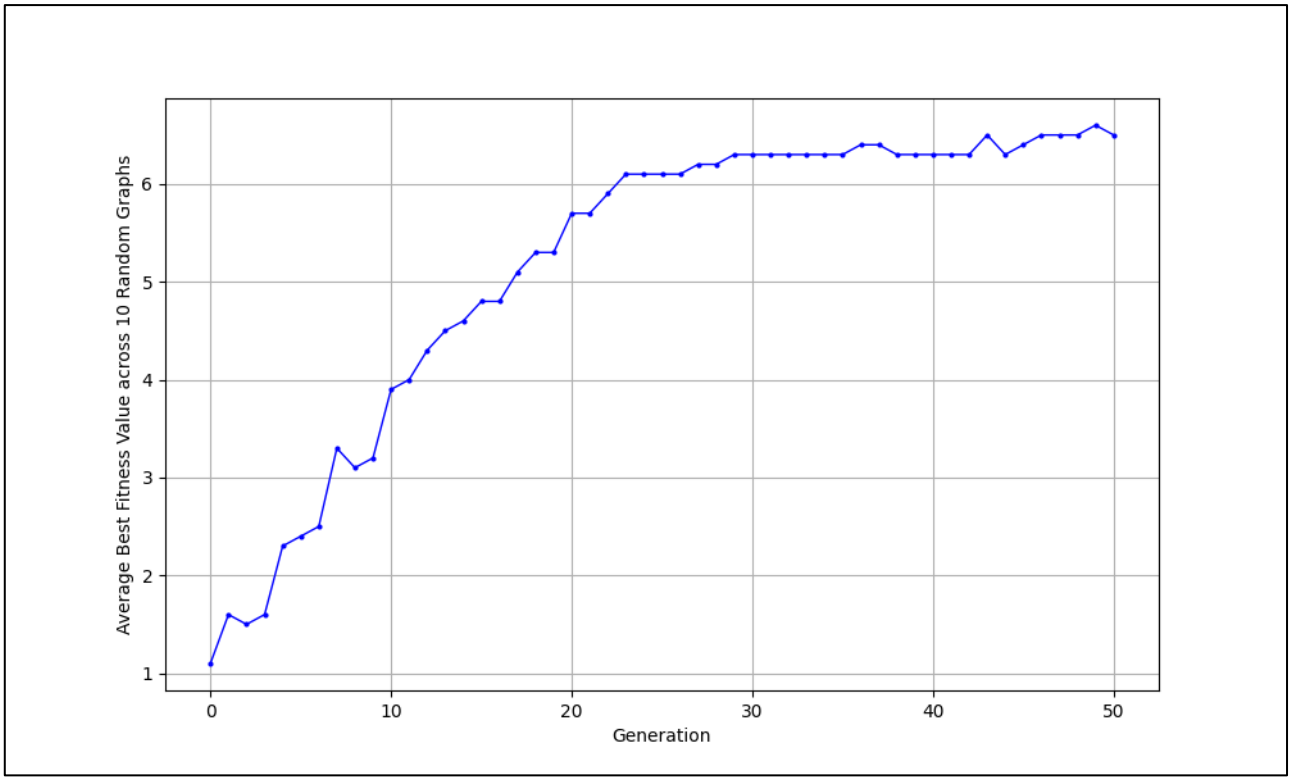
Graph 8: Using Method 2 Normalization, Average value of best fitness value after 50 generations = 8.8



E) Test case: 500 Edges

Method 1 Normalization is invalid for test cases with 500 edges because the sum of the fitness value of all individuals in the population turned out to be zero on multiple instances.

Graph 9: Using Method 2 Normalization, Average value of best fitness value after 50 generations = 6.5



Observation:

Average Best Fitness Value after 50 generations

Edge Set Size	Normalization Method	
	1	2
100	41.7	40.5
200	23.4	21.2
300	13.8	14.4
400	8.8	8.8
500	-	6.5

Average Best Fitness Value decreased drastically as the edge set size increased.

Part B:

This submission presents a two-phased analysis to find an improved version of the genetic algorithm.

- In the first phase, multiple methods have been explored to conduct elementary operations of **Selection** and **Crossover** against a generic set of parameters which consisted of population size, mutation probability, and crossover probability.

- In the second stage of the analysis, parameter optimization for the best-performing genetic algorithm from the first section has been conducted.

Choosing Selection Strategy:

In the analysis presented here, I have considered two selection strategies:

A) Fitness Proportionate (Roulette Wheel) selection (FP)
B) Tournament selection (T)

I made a fundamental change to the genetic algorithm prescribed in the textbook. Instead of choosing two individuals to generate a child *N* times (for population size *N*), my selection function returns a complete population array of size *N*. After shuffling the population array from the selection function, a loop selects pairs of individuals from the array to produce two children from the crossover function. The change was primarily made to ease the implementation of tournament selection which is a highly popular and efficient selection strategy [1].

Choosing Crossover Strategy:

Selecting a crossover function for improving the genetic algorithm depends on several factors, including the state encoding and the targeted problem [2]. Since we are trying to solve the vertex colouring problem with a fixed number of colours (three), each state is represented by a string of 50 digits where each digit can either be 0, 1, or 2. I have disconsidered prevalent crossover strategies such as OX1 [3], PMX [4], and Cycle [5] crossover due to the encoding and have presented the analysis for three crossover strategies:

A) One-point crossover (1P)
B) Two-point crossover (2P)
C) Uniform crossover (U)

Testing Strategy:

Six algorithms were considered to represent all possible combinations of selection and crossover functions. All algorithms were tested against the three test cases provided (edge set size: 50, 100, and 200). All programs were run 25 times, and each instance of the program was terminated after 45 seconds, as specified, with no limitation on the number of generations. The average value of the final best fitness value and the number of generations across 25 runs have been presented.

Phase 1 Results:

The generic set of parameters against which the algorithms were tested:

Population Size = 100
Mutation Probability = 0.4
Crossover Probability = 1

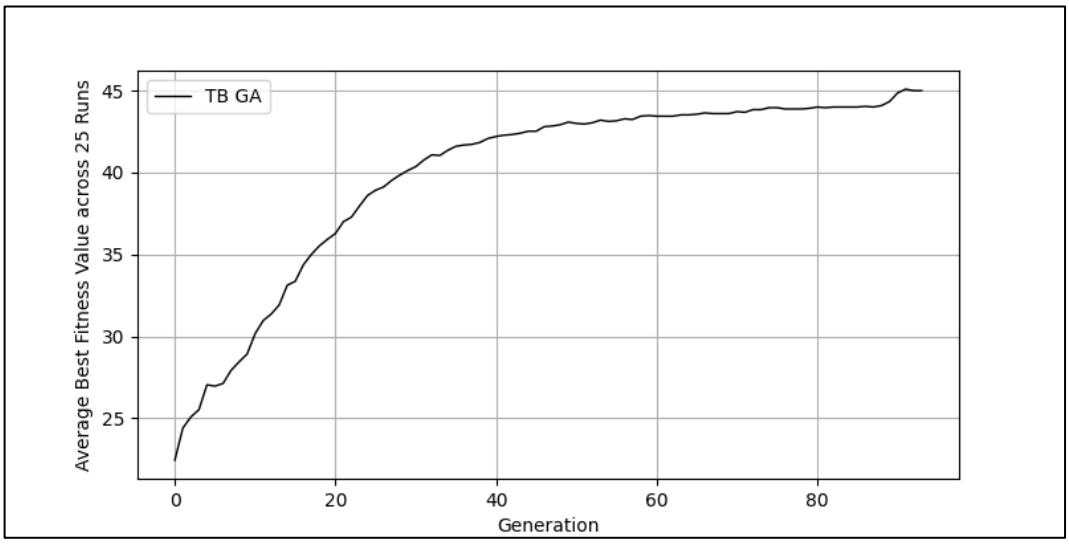
Reasoning: These parameters performed reasonably well for the textbook genetic algorithm.

			Edge Set Size 50		Edge Set Size 100		Edge Set Size 200	
	Selection	Crossover	Avg BFV	Avg No. of Generations	Avg BFV	Avg No. of Generations	Avg BFV	Avg No. of Generations
TB GA	-	-	50	22.72	44.16	90.04	26.4	45.8
GA1	T	1P	50	37.96	48.12	8754.32	29.52	5736.12
GA2	FP	1P	50	170.44	47.4	3395.8	28.4	3479.92
GA3	T	2P	50	28.04	48	8311.64	28.96	5823.4
GA4	FP	2P	50	127.6	47.16	3820.32	29.24	3472.08
GA5	T	U	50	27.28	47.64	6588.56	28.68	4845.32
GA6	FP	U	50	121.28	48.16	3242.56	29.21	3300.2

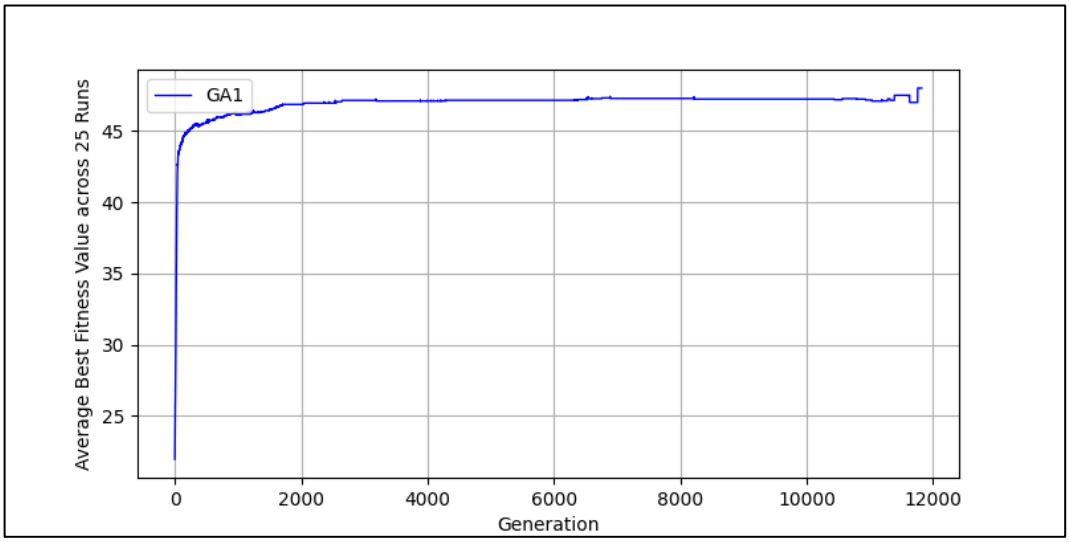
Note: Avg BFV denotes the Best fitness value obtained after the algorithm was terminated after 45 seconds, averaged across 25 runs.

Phase 1 Graphs:

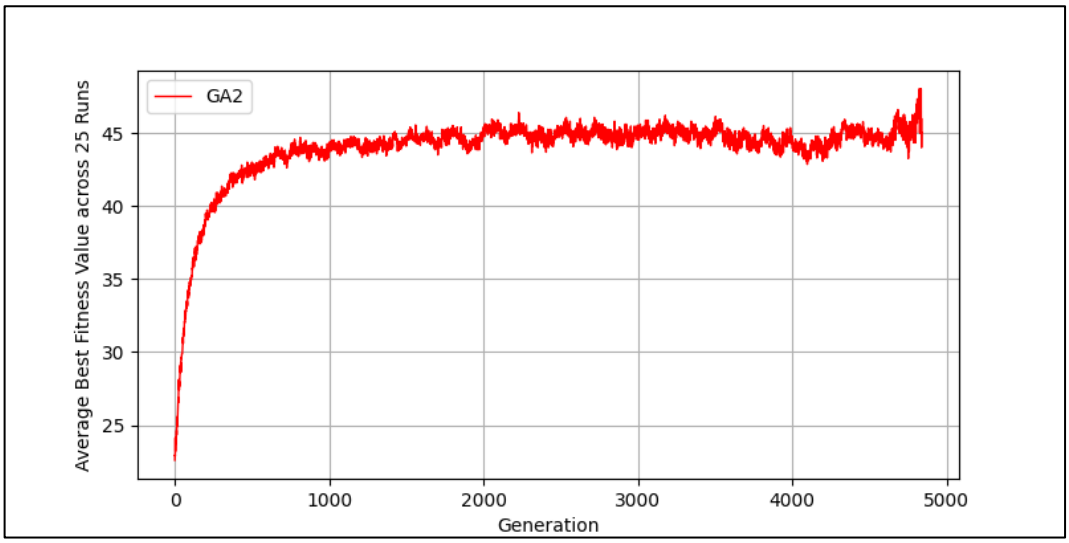
Graph 10: Textbook Genetic Algorithm – Edge Set Size: 100



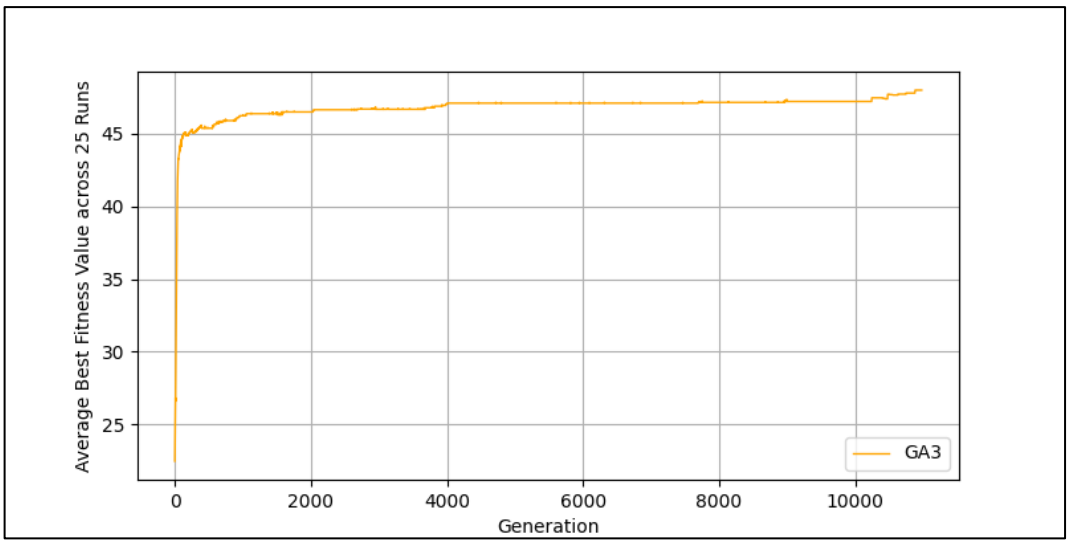
Graph 11: Genetic Algorithm 1 – Edge Set Size: 100



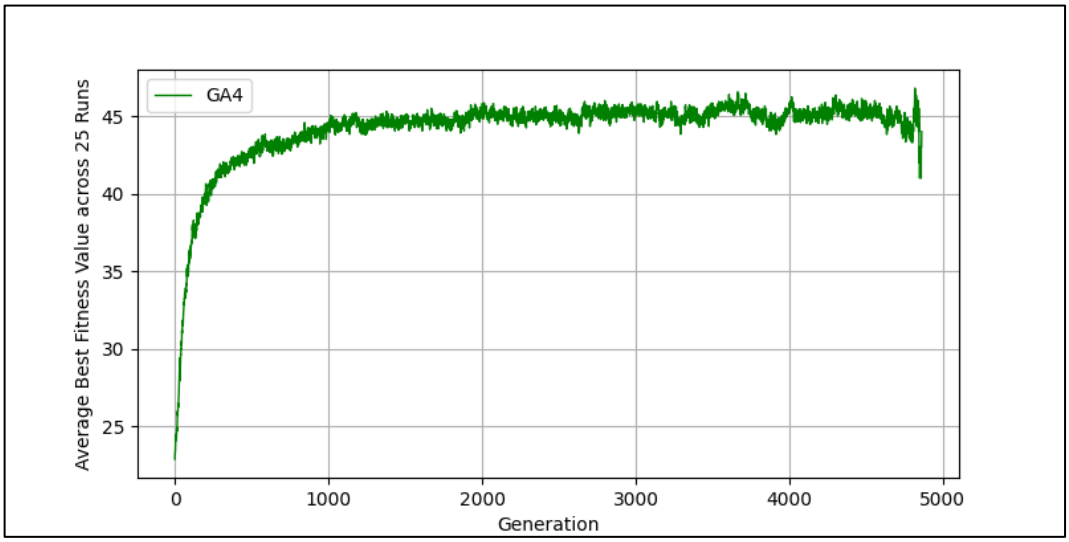
Graph 12: Genetic Algorithm 2 – Edge Set Size: 100



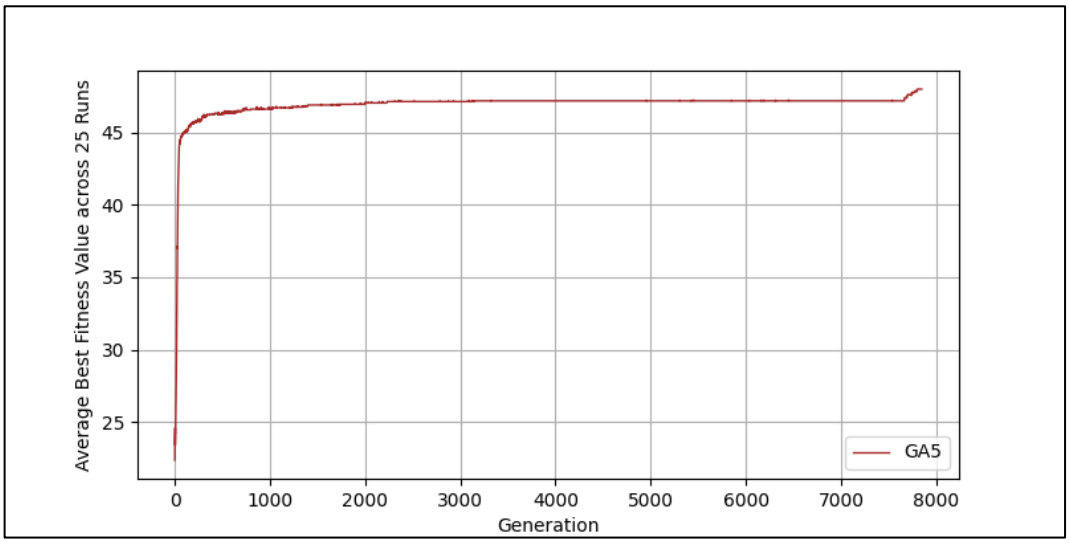
Graph 13: Genetic Algorithm 3 – Edge Set Size: 100



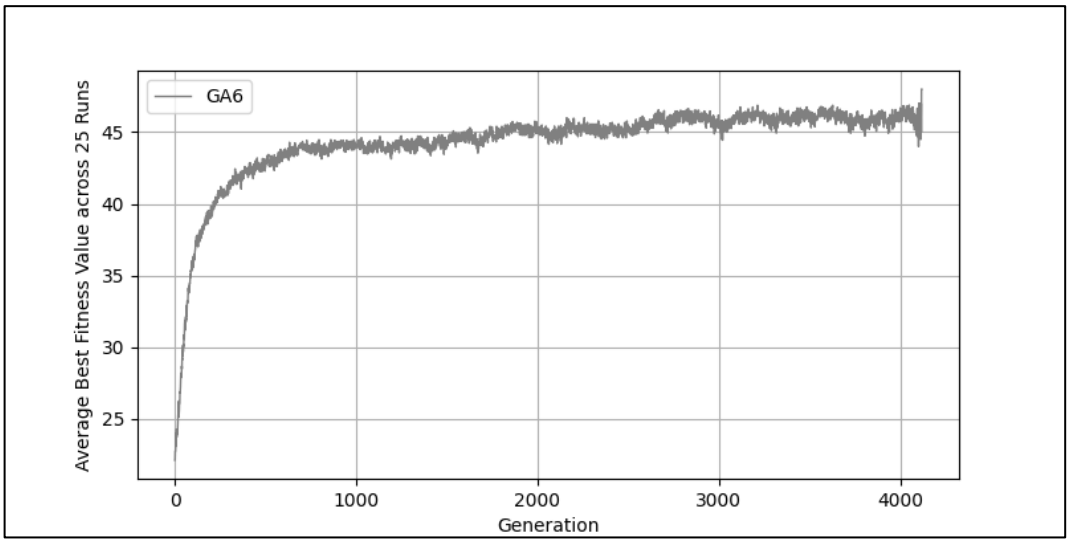
Graph 14: Genetic Algorithm 4 – Edge Set Size: 100



Graph 15: Genetic Algorithm 5 – Edge Set Size: 100



Graph 16: Genetic Algorithm 6 – Edge Set Size: 100



Phase 2 Results:

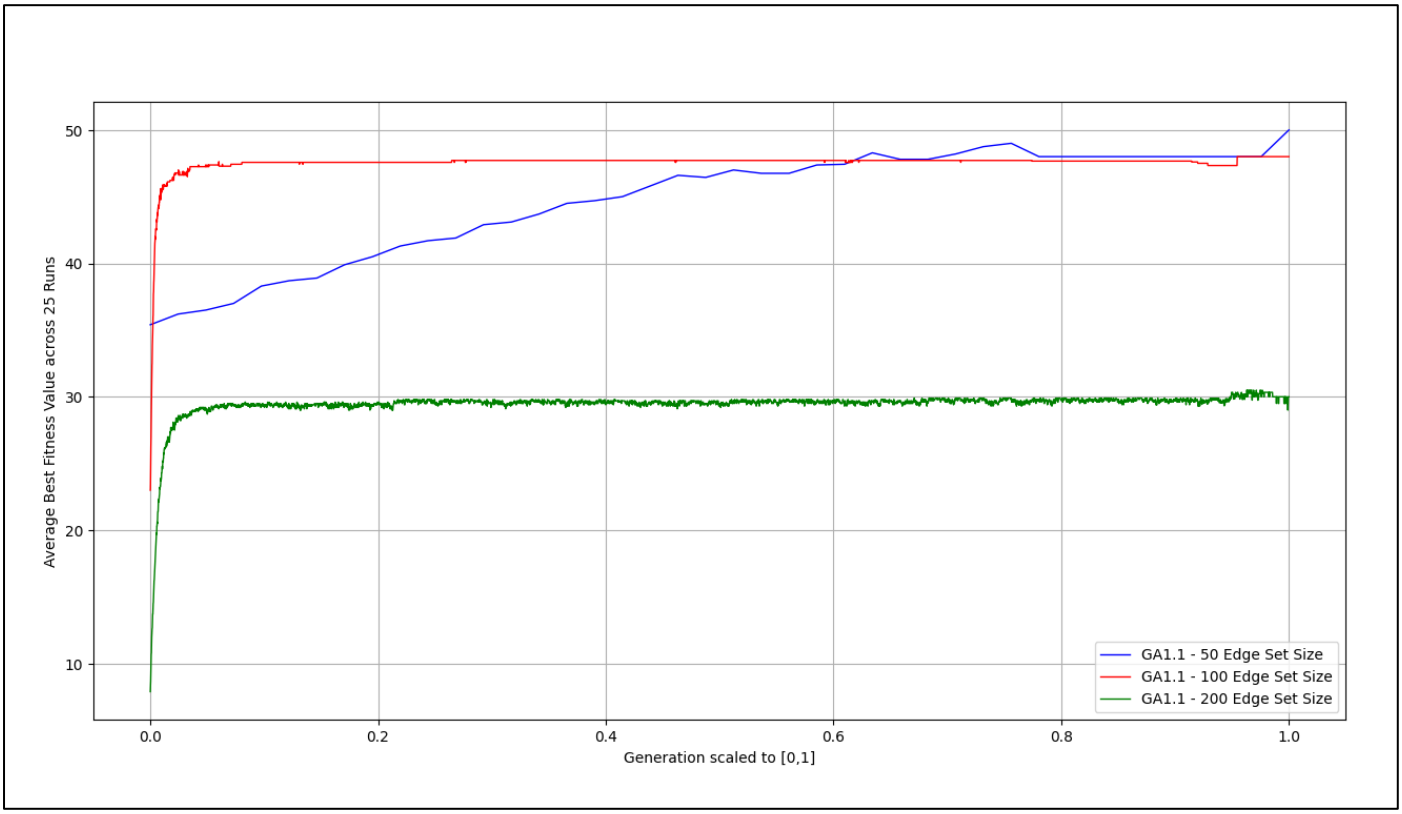
GA1 and **GA6** were the best-performing algorithms from Phase 1. Analyzing the plots for these respective algorithms, GA1 and GA6 algorithms were tested against 6 different sets of parameters.

	Population Size	Mutation Probability	Crossover Probability	Edge Set Size 50		Edge Set Size 100		Edge Set Size 200	
				Avg BFV	Avg No. of Generations	Avg BFV	Avg No. of Generations	Avg BFV	Avg No. of Generations
GA1	100	0.4	1	50	37.96	48.12	8754.32	29.52	5736.12
GA1.1	100	0.8	0.2	50	28.2	49	6228.9	30.3	6156.5
GA1.2	100	0.5	0.5	50	34.6	48	8405.6	29.7	6088.5
GA1.3	100	0.5	1	50	28.6	48.1	9192	28.9	6058
GA1.4	200	0.8	0.2	50	28	48.4	3923.8	30.6	3147.9
GA1.5	200	0.5	0.5	50	29.6	47.9	4348.4	29.1	3031.8
GA1.6	200	0.4	1	50	26.1	47.9	4639	29.2	3017.7
GA1.7	200	0.5	1	50	29	46.9	4345.1	29.3	3111.6

	Population Size	Mutation Probability	Crossover Probability	Edge Set Size 50		Edge Set Size 100		Edge Set Size 200	
				Avg BFV	Avg No. of Generations	Avg BFV	Avg No. of Generations	Avg BFV	Avg No. of Generations
GA6	100	0.4	1	50	121.28	48.16	3242.56	29.21	3300.2
GA6.1	100	0.1	0.9	50	28.8	48.77	6916.4	27.6	5989.6
GA6.2	100	0.2	0.8	50	40.1	47.7	9163.3	28.4	6116.5
GA6.3	100	0.3	1	50	25.6	47.8	9336.9	29.3	6109.3
GA6.4	200	0.1	0.9	50	27.2	47.4	4288.5	27.8	2982.7
GA6.5	200	0.2	0.8	50	27.8	46.9	5271.4	28.9	3104.2
GA6.6	200	0.3	1	50	27.1	47.6	4704.7	29.3	2975
GA6.7	200	0.4	1	50	26.4	47.2	4784.9	28.5	2983.5

After observing the results from phase 2, the GA1.1 algorithm with (Population size, Mutation probability, Crossover probability) = (100, 0.8, 0.2) is selected as the final algorithm for submission

Graph 17: GA1.1 (100, 0.8, 0.2) G against Edge Set Size: 50, 100 and 200



Outputs for Test cases given:

GA1.1 (100, 0.8, 0.2) Output – Edge Set Size = 50

```
C:\Users\Madhav\AppData\Local\Microsoft\WindowsApps\python3.9.exe D:/Academics/SEM7/AI/Assignment-1/2019B3A70256G_Madhav.py
Roll no : 2019B3A70256G
Number of edges : 50
Best state :
0:B, 1:G, 2:G, 3:G, 4:R, 5:B, 6:B, 7:B, 8:R, 9:R, 10:B, 11:G, 12:G, 13:R, 14:B, 15:G, 16:G, 17:R, 18:R, 19:G, 20:G, 21:R,
22:G, 23:G, 24:G, 25:B, 26:G, 27:R, 28:R, 29:B, 30:R, 31:B, 32:G, 33:B, 34:B, 35:B, 36:B, 37:R, 38:R, 39:B, 40:G, 41:G,
42:R, 43:B, 44:R, 45:R, 46:G, 47:R, 48:B, 49:B
Fitness Value of best state : 50
Time taken : 0.13 seconds
```

GA1.1 (100, 0.8, 0.2) Output – Edge Set Size = 100

```
C:\Users\Madhav\AppData\Local\Microsoft\WindowsApps\python3.9.exe D:/Academics/SEM7/AI/Assignment-1/2019B3A70256G_Madhav.py
Roll no : 2019B3A70256G
Number of edges : 100
Best state :
0:G, 1:R, 2:G, 3:B, 4:G, 5:R, 6:G, 7:B, 8:G, 9:B, 10:B, 11:G, 12:R, 13:G, 14:B, 15:B, 16:R, 17:B, 18:R, 19:R, 20:G, 21:B,
22:B, 23:R, 24:R, 25:R, 26:B, 27:R, 28:G, 29:B, 30:R, 31:B, 32:B, 33:G, 34:R, 35:G, 36:G, 37:R, 38:G, 39:B, 40:G, 41:R,
42:G, 43:R, 44:R, 45:B, 46:B, 47:R, 48:B, 49:R
Fitness Value of best state : 50
Time taken : 5.65 seconds
```

GA1.1 (100, 0.8, 0.2) Output – Edge Set Size = 200

```
Roll no : 2019B3A70256G
Number of edges : 200
Best state :
0:G, 1:B, 2:G, 3:B, 4:B, 5:B, 6:G, 7:G, 8:G, 9:G, 10:G, 11:R, 12:R, 13:G, 14:G, 15:B, 16:B, 17:R, 18:B, 19:B, 20:G, 21:B,
22:G, 23:R, 24:R, 25:R, 26:B, 27:R, 28:R, 29:G, 30:G, 31:R, 32:R, 33:G, 34:R, 35:G, 36:G, 37:G, 38:R, 39:B, 40:G, 41:B,
42:B, 43:G, 44:B, 45:G, 46:G, 47:R, 48:R, 49:B
Fitness Value of best state : 31
Time taken : 44.51 seconds
```

References:

1) https://ieeexplore.ieee.org/abstract/document/7154916
2) https://arxiv.org/ftp/arxiv/papers/1801/1801.02335.pdf
3) https://www.rubicite.com/Tutorials/GeneticAlgorithms/CrossoverOperators/Order1CrossoverOperator.aspx
4) https://www.rubicite.com/Tutorials/GeneticAlgorithms/CrossoverOperators/PMXCrossoverOperator.aspx
5) https://www.rubicite.com/Tutorials/GeneticAlgorithms/CrossoverOperators/CycleCrossoverOperator.aspx