A REPORT
ON
VERTEX COLOURING USING GENETIC ALGORITHM

BY

Rishi Jain 2020A8PS1772G

For Programming Assignment 1 of Artificial Intelligence (CS F407)

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

(OCTOBER, 2022)

Problem Statement

Vertex colouring problem: You are given a randomly generated undirected graph (V, E) where the vertex set V always contains 50 vertices. The edge set E contains edges that are randomly selected from all possible edges in the graph. You can assume that there are no self-loop edges in the graph. The fifty vertices are numbered from 0 to 49. The problem is to colour each vertex with any of the three colours — Red, Green and Blue — such that no two adjacent vertices have the same colour. A state contains information about colour assignments for all the fifty vertices. The fitness function value for a state is the number of vertices that are coloured such that no two adjacent vertices have the same colour. The goal state (if exists) will be the solution for the vertex colouring problem and will have a fitness function value of 50.

Introduction

A genetic algorithm (or GA) is a variant of stochastic beam search in which successor states are generated by combining two parent states rather than by modifying a single state. The analogy to natural selection is the same as in stochastic beam search, except that now we are dealing with sexual rather than asexual reproduction [1].

Part a

For Part a, I implemented the version of the Genetic Algorithm given in the textbook.

Parameters Used for Part a

Mutation Rate: 20%

Population size: 100

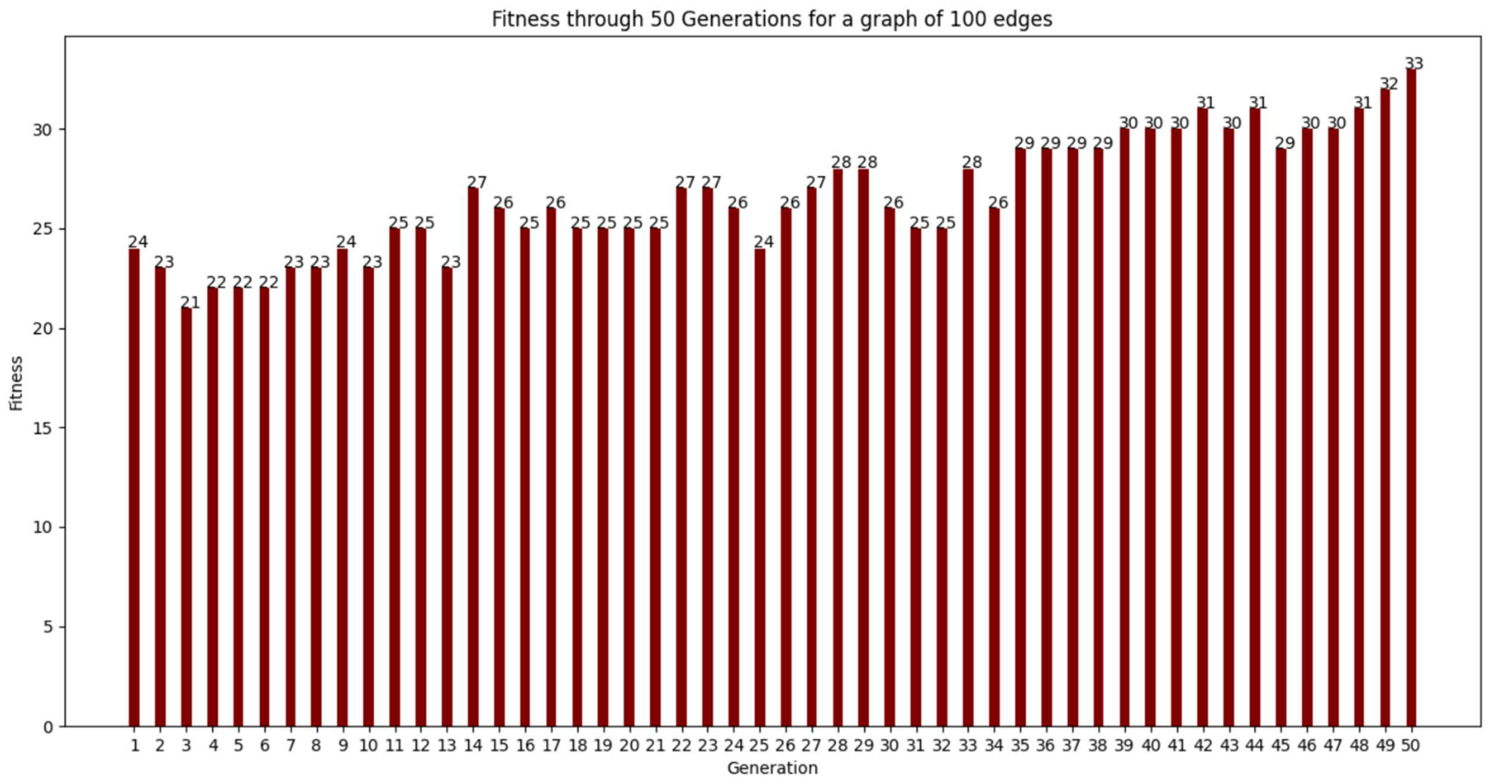
Generations: 50

Parent Selection: Roulette wheel selection

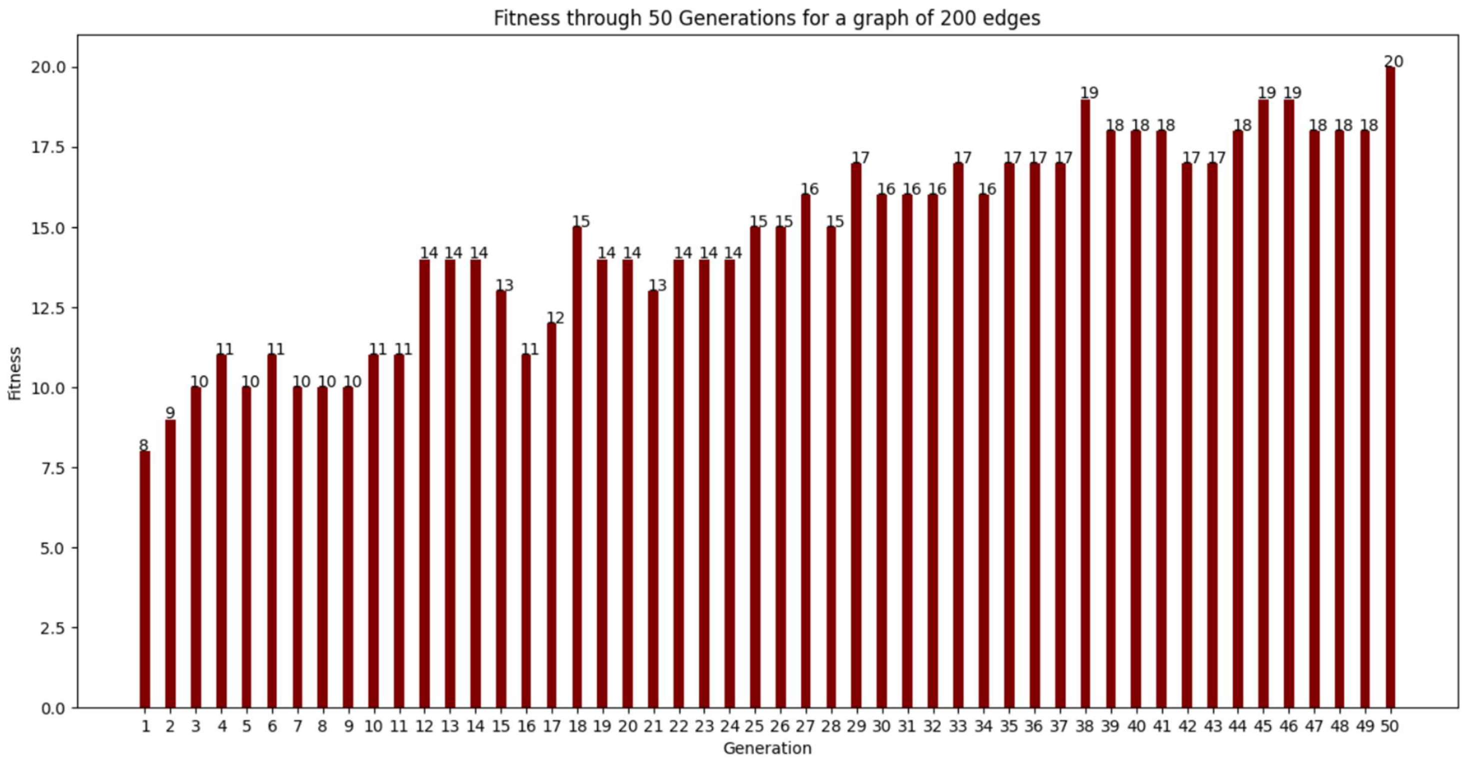
Results Obtained for Part a

Fitness through 50 Generations for 100, 200, 300, 400, 500 edges

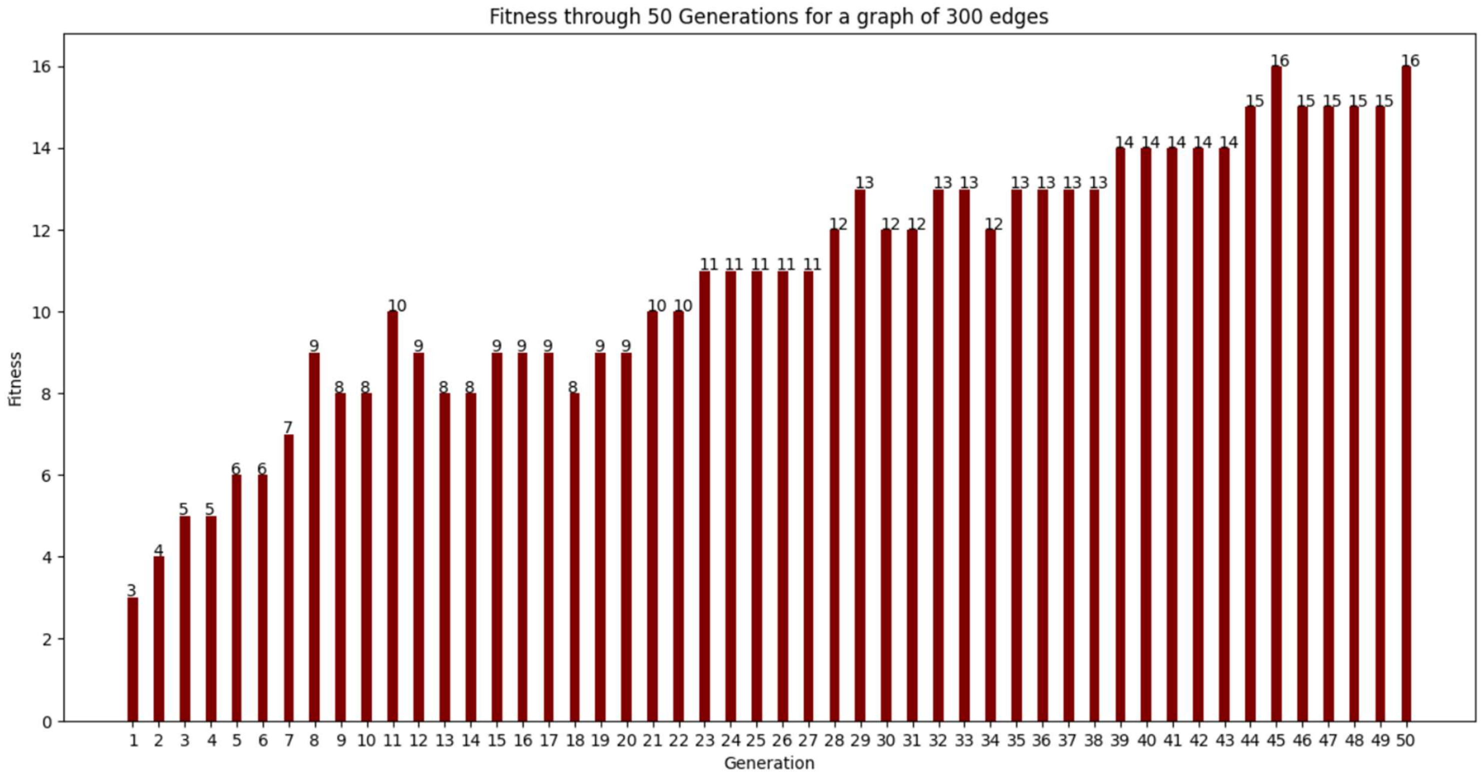
1. For 100 Edges



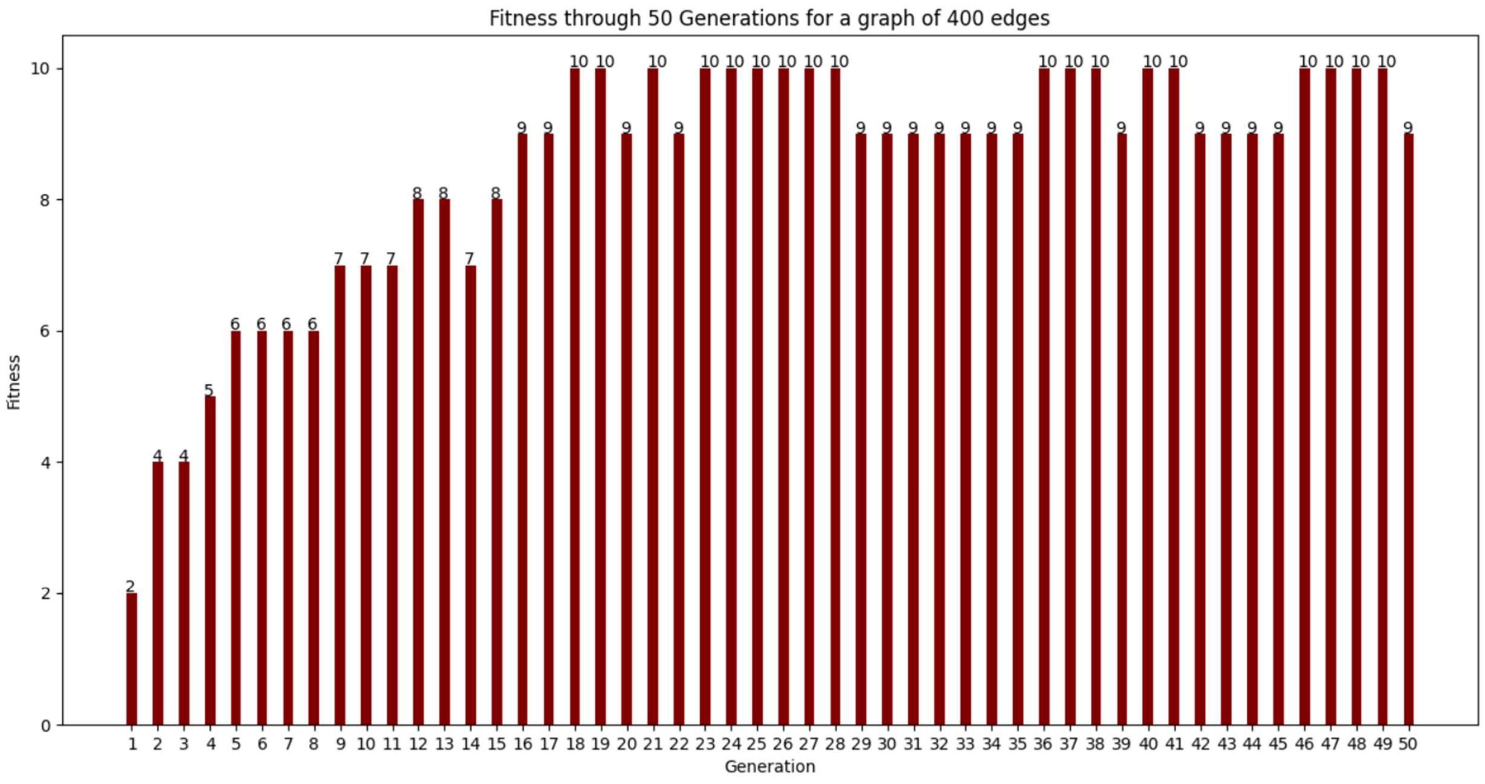
2. For 200 Edges



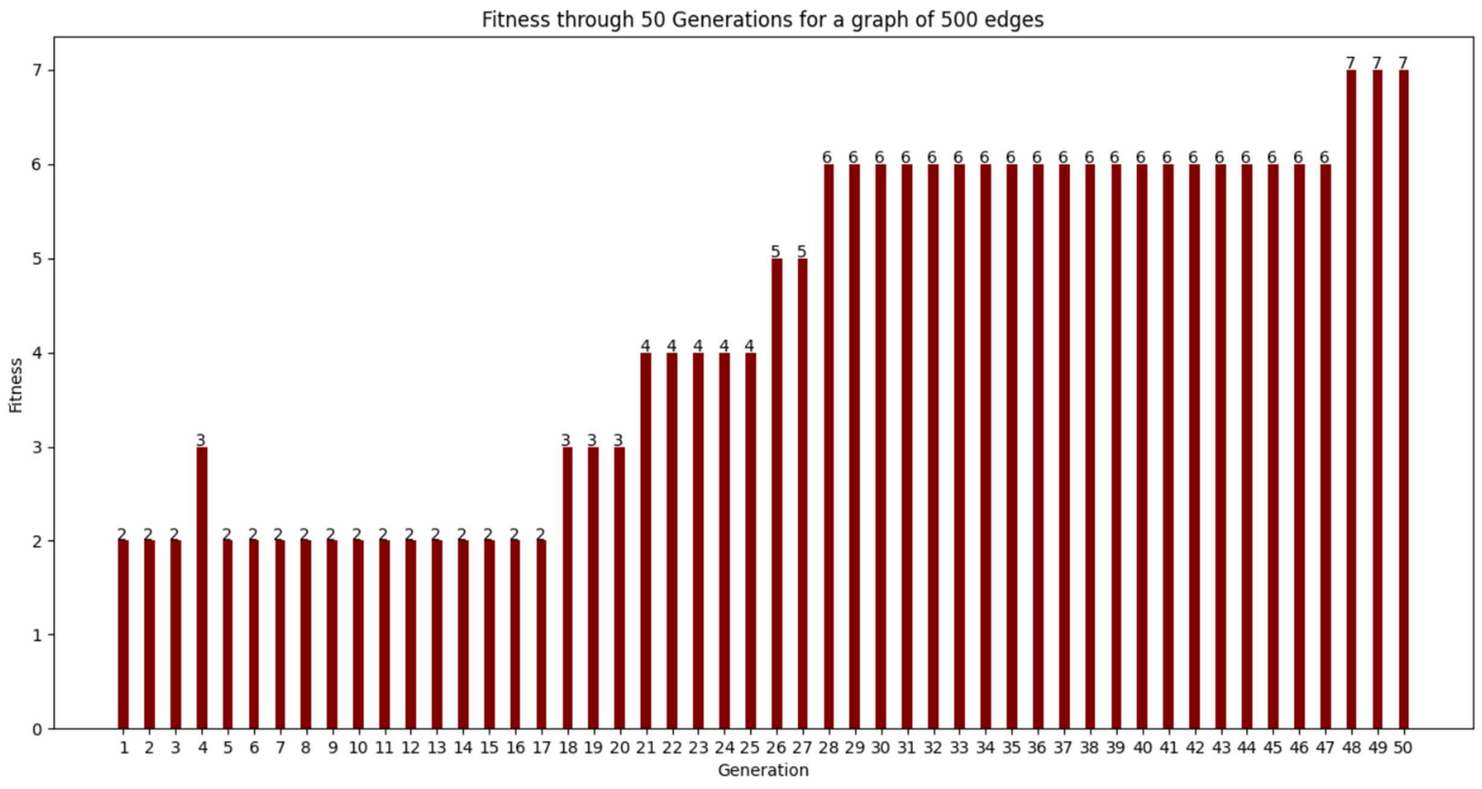
3. For 300 Edges



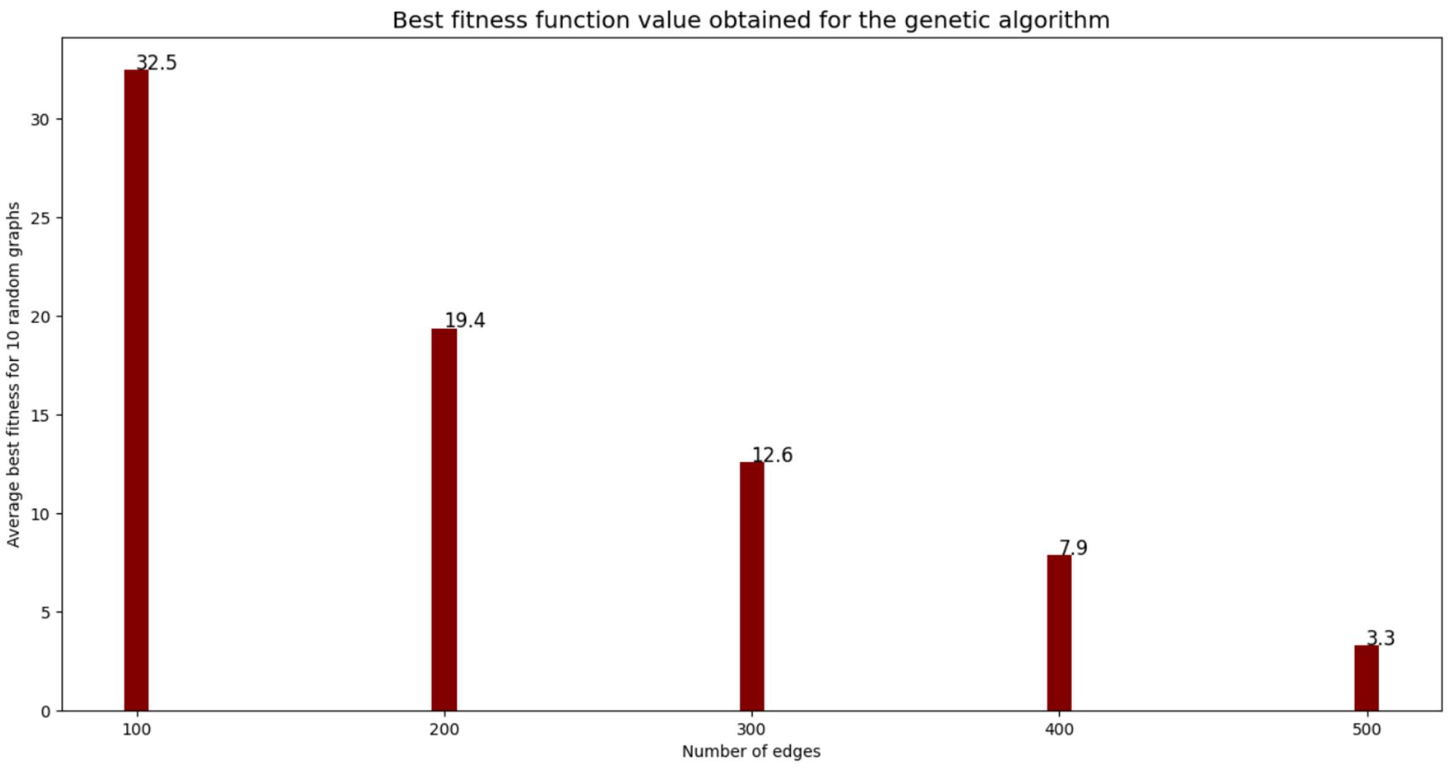
4. For 400 Edges



5. For 500 Edges



Best fitness obtained over 50 generations for different numbers of edges



Observation made in Part a

By looking at the first five graphs, we can deduce that the fitness of the best state in the population increases gradually with increasing generation. But after increasing the number of edges the final fitness (after the 50th generation) goes down. Also, if we increase the number of edges, then the change in the fitness between first-generation and the last-generation goes down.

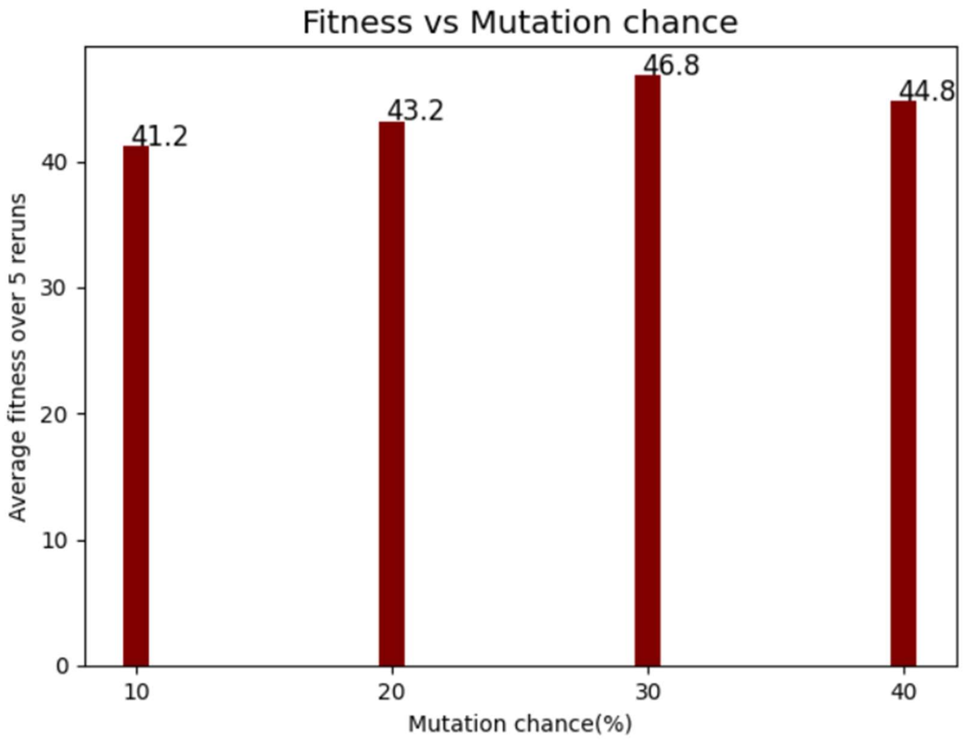
From the last graph (plotted between the Average best fitness and the Number of edges) we can see that the best fitness obtained goes down significantly.

Part b

For Part b of the assignment, we were required to improve the already implemented Genetic Algorithm from Part a

Improvements Tried

1. Number of generations – One of the biggest limitations of the previous implementation was the number of generations, so I decided to keep the algorithm running for the maximum allowed time, and that itself will decide the number of generations.
2. Mutation Rate – After playing around with different values of the Mutation Rate for 100 edges and a population size of 100, I got the Following Graph:

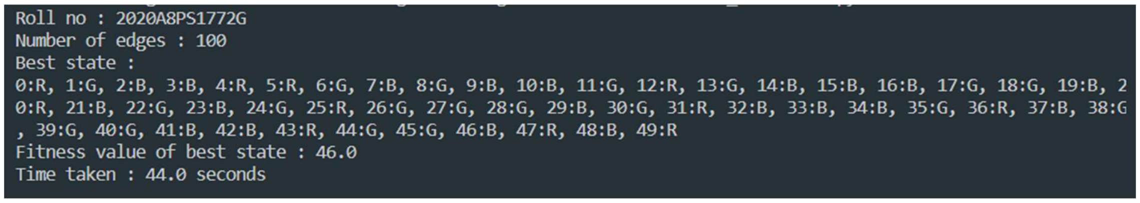


It is clearly visible from the above graph that the best average best fitness is the highest when the Mutation Rate is equal to 30%, so I have kept the same value in the final solution.

3. Population size – Initially the population size was 100, but after experimenting with different values I found out that the average best fitness increased slightly by changing the population size to 150 so I decided to keep the population size at 150 for the final solution.
4. Elitism/Culling – I took a fixed number (50 in the final solution) with the highest fitness of states and passed them directly to the next generation directly while removing the equal number of states from the new generation with the lowest fitness. This did not give any improvement to the average fitness so I decided to exclude it from the final solution.

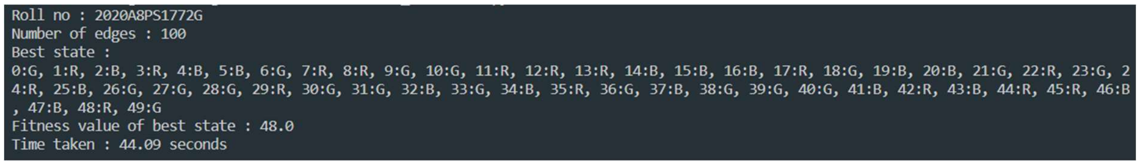
Final Outputs for Test Cases

1. For 50.csv



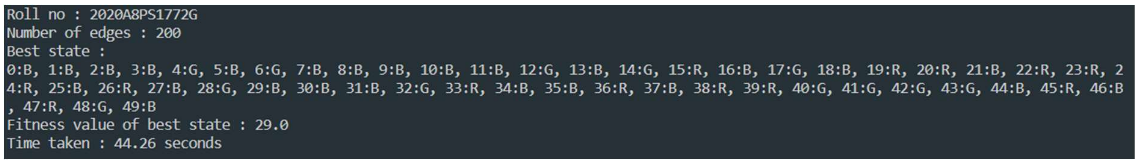
```
Roll no : 2020A8PS1772G
Number of edges : 100
Best state :
0:R, 1:G, 2:B, 3:B, 4:R, 5:R, 6:G, 7:B, 8:G, 9:B, 10:B, 11:G, 12:R, 13:G, 14:B, 15:B, 16:B, 17:G, 18:G, 19:B, 2
0:R, 21:B, 22:G, 23:B, 24:G, 25:R, 26:G, 27:G, 28:G, 29:B, 30:G, 31:R, 32:B, 33:B, 34:B, 35:G, 36:R, 37:B, 38:G
, 39:G, 40:G, 41:B, 42:B, 43:R, 44:G, 45:G, 46:B, 47:R, 48:B, 49:R
Fitness value of best state : 46.0
Time taken : 44.0 seconds
```

2. For 100.csv



```
Roll no : 2020A8PS1772G
Number of edges : 100
Best state :
0:G, 1:R, 2:B, 3:R, 4:B, 5:B, 6:G, 7:R, 8:R, 9:G, 10:G, 11:R, 12:R, 13:R, 14:B, 15:B, 16:B, 17:R, 18:G, 19:B, 20:B, 21:G, 22:R, 23:G, 2
4:R, 25:B, 26:G, 27:G, 28:G, 29:R, 30:G, 31:G, 32:B, 33:G, 34:B, 35:R, 36:G, 37:B, 38:G, 39:G, 40:G, 41:B, 42:R, 43:B, 44:R, 45:R, 46:B
, 47:B, 48:R, 49:G
Fitness value of best state : 48.0
Time taken : 44.09 seconds
```

3. For 200.csv



```
Roll no : 2020A8PS1772G
Number of edges : 200
Best state :
0:B, 1:B, 2:B, 3:B, 4:G, 5:B, 6:G, 7:B, 8:B, 9:B, 10:B, 11:B, 12:G, 13:B, 14:G, 15:R, 16:B, 17:G, 18:B, 19:R, 20:R, 21:B, 22:R, 23:R, 2
4:R, 25:B, 26:R, 27:B, 28:G, 29:B, 30:B, 31:B, 32:G, 33:R, 34:B, 35:B, 36:R, 37:B, 38:R, 39:R, 40:G, 41:G, 42:G, 43:G, 44:B, 45:R, 46:B
, 47:R, 48:G, 49:B
Fitness value of best state : 29.0
Time taken : 44.26 seconds
```