FAKULTI TEKNOLOGI MAKLUMAT DAN KOMUNIKASI

SEMESTER II 2019/20

BITI 3143 EVOLUTIONARY COMPUTING

PROJECT TITLE:

MOVIE SCHEDULING

GROUP CD2

PREPARED BY:

NAME	MATRIC NO
Lai Zi Wei	B031710192
Loh Jun Wen	B031710157
Shazana Binti Mohammed Affandi	B031710303
Norsyakirah Binti Mohd Rasid	B031710326
Hariz Taqiuddin Bin Rahim	B031710391

PREPARED FOR:

Ts. Dr. Zeratul Izzah Mohd Yusoh

1.0 PROBLEM STATEMENT

Cinema is place that where people will watch the movie. They can look the schedule of the movie online or at the cinema. Customer can book the ticket and know the hall that they will watch the movie.

For the cinema, the staff need to schedule the movie that want to show based on time and rating. Usually the high rating will many people want to watch so that the schedule needs to show the movie frequently compare to the low rating movie.

The staff is struggling to schedule the movie with multiple rating and time that want to fit in one day. Thus, in the way to solve the problem, we decided to implement with an Evolutionary Algorithm based project. Having 34 movies with different time and rating will be use for the best scheduling.

2.0 PROBLEM INSTANCE

In this project will uses 30 different movies in 2018 year with different rating and different duration of time. The data has been collected from the IMDb website, it is an online database of information related to films, television programs, home videos, video games, and streaming content online. To make sure the algorithm running smoothly the different duration of movie time and different rating for all the movies are taken without any bias. Table below shows the list of movies that have been collected.

No	Movie name	Duration in minutes	Rating
1	PASKAL: The Movie	115	6.8
2	Flavors of Youth	75	6.8
3	The Meg	113	5.7
4	Be With you	132	7.6
5	Psychokinesis	101	5.9
6	Searching	102	7.6
7	Fifty Shades Freed	110	4.5
8	Slender Man	91	3.2
9	High Society	120	5.4
10	TAG	95	6.5
11	Sabrina	112	4.2
12	The Nun	96	5.3
13	Revenger	102	5.7
14	A Quiet Place	216	7.5
15	Jurassic World: Fallen Kingdom	130	6.2
16	Maze Runner 3	143	6.2
17	Ready Player One	139	7.5
18	Venom	150	6.7
19	Aquaman	142	7
20	Along with the Gods 2	142	7.1
21	Bird Box	124	6.6
22	Illang: The Wolf Brigade	139	6
23	Adrift	99	6.6
24	Rampage	115	6.1
25	Teman Tapi Menikah	102	7
26	Extinction	95	5.8
27	Deep Blue Sea 2	94	3.3
28	Breaking In	88	5.4
29	Night School	112	5.6
30	Insidious: The Last Key	102	5.7

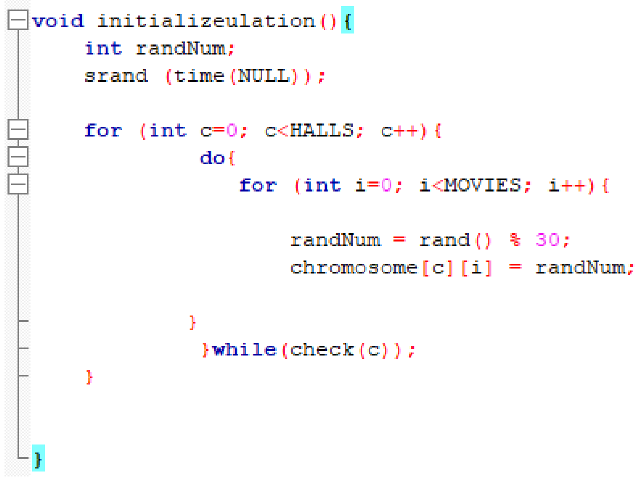
3.0 PSEUDOCODE

Pseudocode is an artificial and informal language which helps to develop algorithms for programmers. It is essentially an overview of the algorithm which makes it simple to understand and serves as a reference to project creation along the way. Following is the pseudo-code for this project:

```
START
    Initialize population with 30
    Display Print Chromosome
    Evaluate Chromosome by calculating duration and rate
    REPEAT
        ParentSelection() Tournament Selection
        Crossover() using One-point Crossover
        Mutation() using Floating Point Mutation
        SurvivalSelection() Childrent replace parent
    UNTIL generation 30
STOP
```

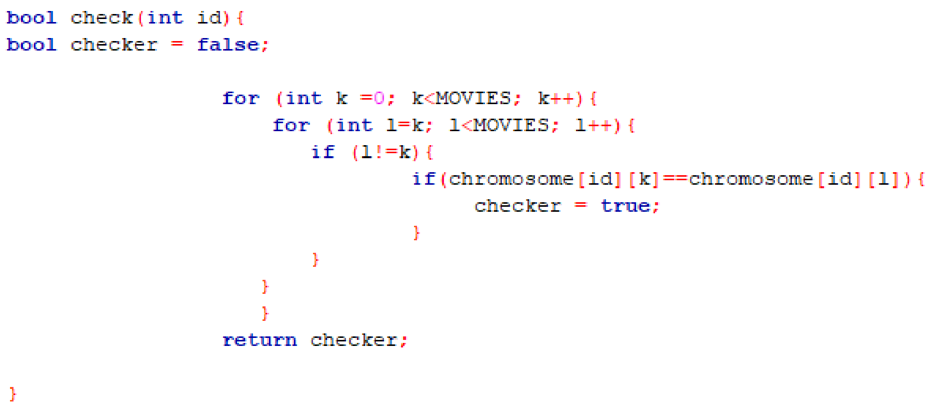
4.0 EXPLANATION OF STRATEGY

4.1 Population Initialization



```
void initializeulation(){
    int randNum;
    srand (time(NULL));

    for (int c=0; c<HALLS; c++){
            do{
                for (int i=0; i<MOVIES; i++){

                    randNum = rand() % 30;
                    chromosome[c][i] = randNum;

            }
            }while(check(c));
    }


}
```

Figure 1 shows the initialization of the population



```
bool check(int id){
bool checker = false;

                for (int k =0; k<MOVIES; k++){
                    for (int l=k; l<MOVIES; l++){
                       if (l!=k){
                               if(chromosome[id][k]==chromosome[id][l]){
                                    checker = true;
                               }
                       }
                    }
                }
                return checker;

}
```

Figure 2 shows the check function

Each chromosome represents the number of movies that will show per day. Each gene in the chromosome represent the id number of the movie.

Figure shows the coding function for initialization where we generate a random value from 0 to 29 for the gene which based on the movie id. The population size is 10, meaning that there are 10 set of chromosomes and 6 genes in each of the chromosomes.

The checking function is the implementation of hybridization of making a certain heuristic for initializing population as to makes sure that the first population does not have repeating movies in them. Because if we have repeating movies in them it would effect the fitness score of the chromosome.

4.2 Fitness Evaluation

For genetic algorithms, every solution is usually interpreted as a set of real numbers, known as a chromosome. We need to analyze these solutions to find the right combination of solutions to solve a particular problem. Therefore, a ranking must be given to each solution, to show how close it came to achieving the set requirements of the desired solution.

For this genetic algorithm the fitness function is used to measure how close a given solution is to the optimal solution of the problem. Those are the method used to evaluate the timeOptimization and ratingOptimization for fitness function. The best chromosome has a higher value for fitness function. So, the higher the value of fitness value, the better the solution.

The fitness function formula:

total_time=MOVIES*120;
double total_r=MOVIES*10;

First equation (Time Optimization):

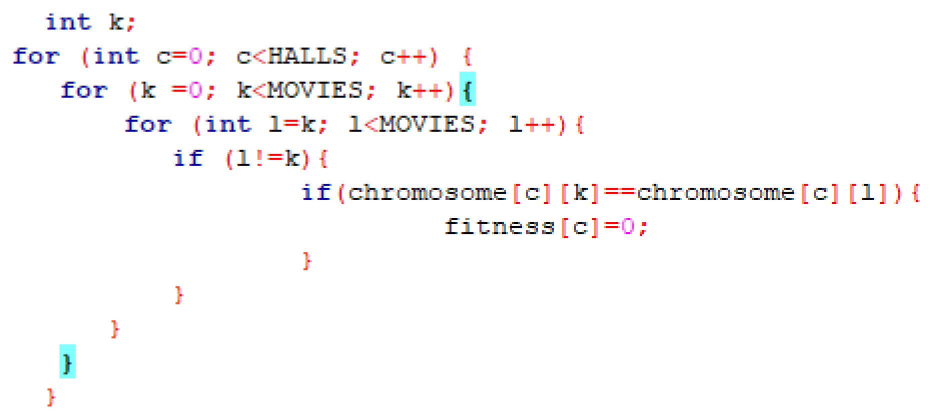
p1 = $\frac{1-(|\text{total\_time}-total\ runtime|)}{\text{total\_time}}$

Second equation (Rating Optimization):

p2 = $\frac{total\ rating}{\text{total\_r}}$

Fitness Function: $\frac{p1+p2}{2}$

Constrain



```
  int k;
for (int c=0; c<HALLS; c++) {
   for (k =0; k<MOVIES; k++){
       for (int l=k; l<MOVIES; l++){
          if (l!=k){
                  if(chromosome[c][k]==chromosome[c][l]){
                           fitness[c]=0;
                  }
          }
      }
   }
  }
```

Figure 3 The constrain

To ensure that the chromosome with repetitive movies in the population would have a bad fitness we have used this constrain. If a chromosome happens to have a repeated movie in its chromosome the fitness value of it would be 0.

4.3 Parent Selection

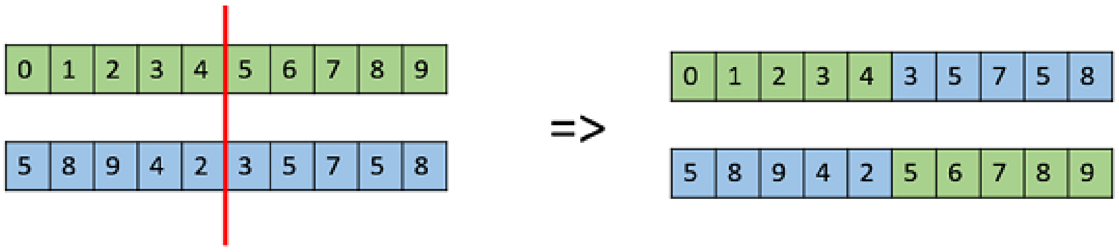
Tournament-based is a strategy used for parent selection. Parent selection is the process by which parents are selected and recombined to create off-springs for next generation.

Two chromosomes will be selected randomly from the population. Then, the two chromosomes in the pair will be competing together. If else statement is used to choose the winner based on the higher fitness value between three chromosomes. The winner from each pair will become a parent. The same process is repeated for number_of_times. Where number_of_times is the population divide by half.

4.4 Crossover

In this project, One-point crossover is being use as the crossover algorithm. One-point crossover will randomly pick an index and divide the parent chromosome into two. After the process of dividing done the chromosome will combine to form the new chromosome. To make sure there is no repeating for the movie scheduling for each of the children the one-point crossover has been used as the crossover algorithm in this project. For generating random numbers, we use the following formula and some of the example of one-point crossover:

$$prob = ((randomindex \% 10) + 1)/10$$



4.5 Mutation

For the mutation process, Floating Point Mutation will be used in this project as the mutation algorithm. For our cases we will use movie as the gene and set by the variable 6 gene for each chromosome. To ensure the movie id for each chromosome are not same we randomly generate number from 1 to 30. For example, the Floating Point Mutation will randomly pick a random number in the chromosome and flip them into any of random number. The advantage of floating point is it can avoid the problem of mutation for the movie scheduling.

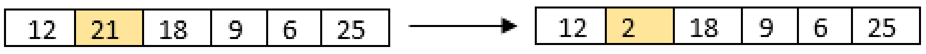


Figure above are the example of bit flip mutation for our project. The bit randomly chooses second column and change into random number id from 1-30. The random number for this project represented as movie id.

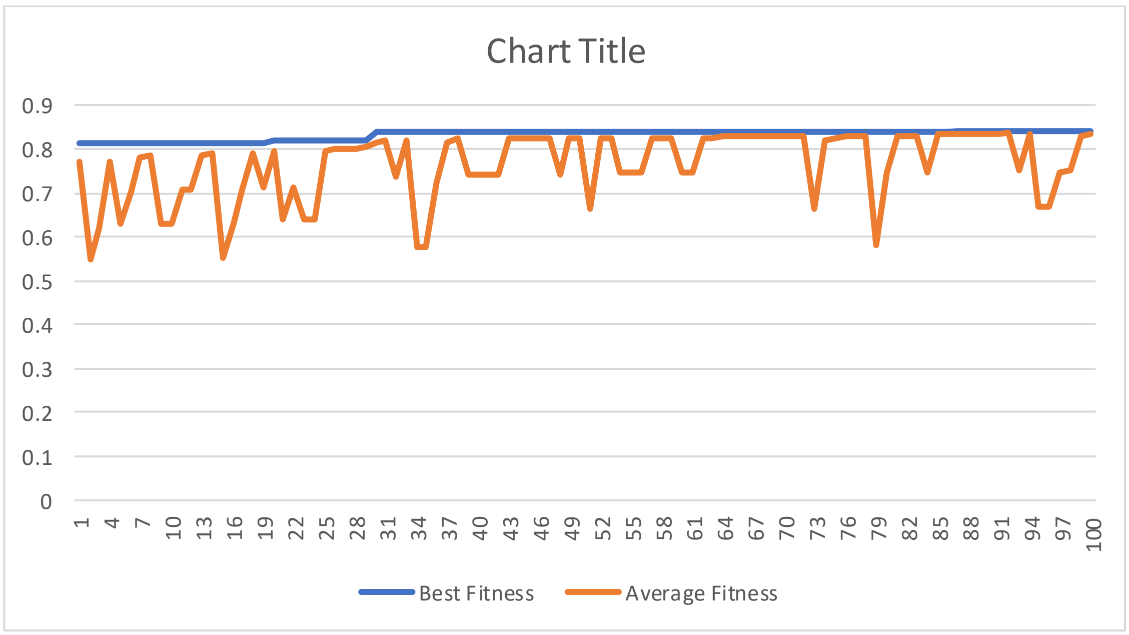
4.6 Survival Selection

In Survival Selection will determine which individual will be kicked out and which will be kept for the next generation. For this project, we would be practicing survival of the fittest. For the next generation we would have the combination of parents and children in the present generation.

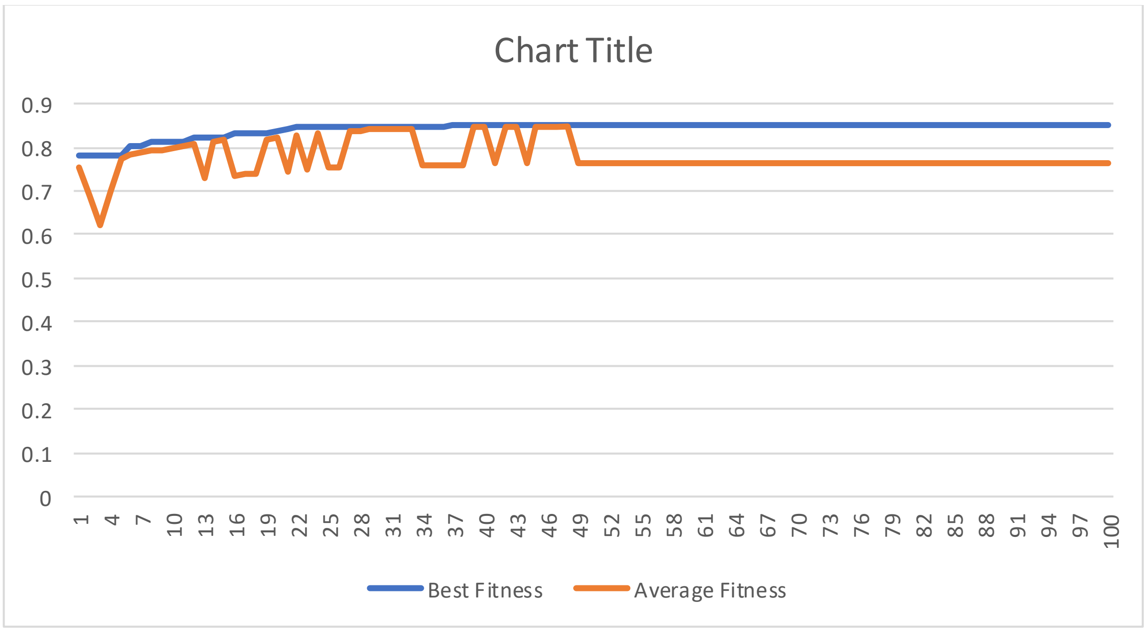
To achieve this, we would create an container array to combine the parents solution with the children solution. When this array is created we sort the chromosome according to their fitness value, then we take a population from the container array and pass it to the next generation.

5.0 ANALYSIS OF GRAPH PERFORMANCE

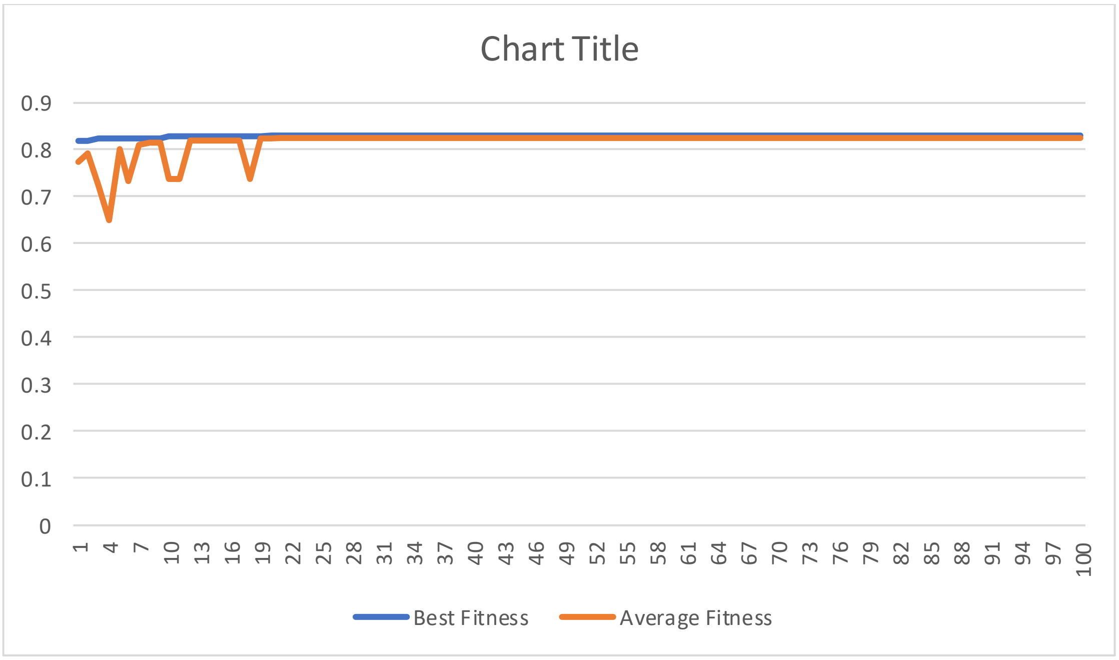
The graph of evolution performance is based on best fitness and average fitness values of algorithm. The comparison of each of the graphs based on the changing value of populations and mutations. The results achieved are run three times for each changing value. The maximum generation values are at constant which is 100. The result is shown below:



Graph 1.1

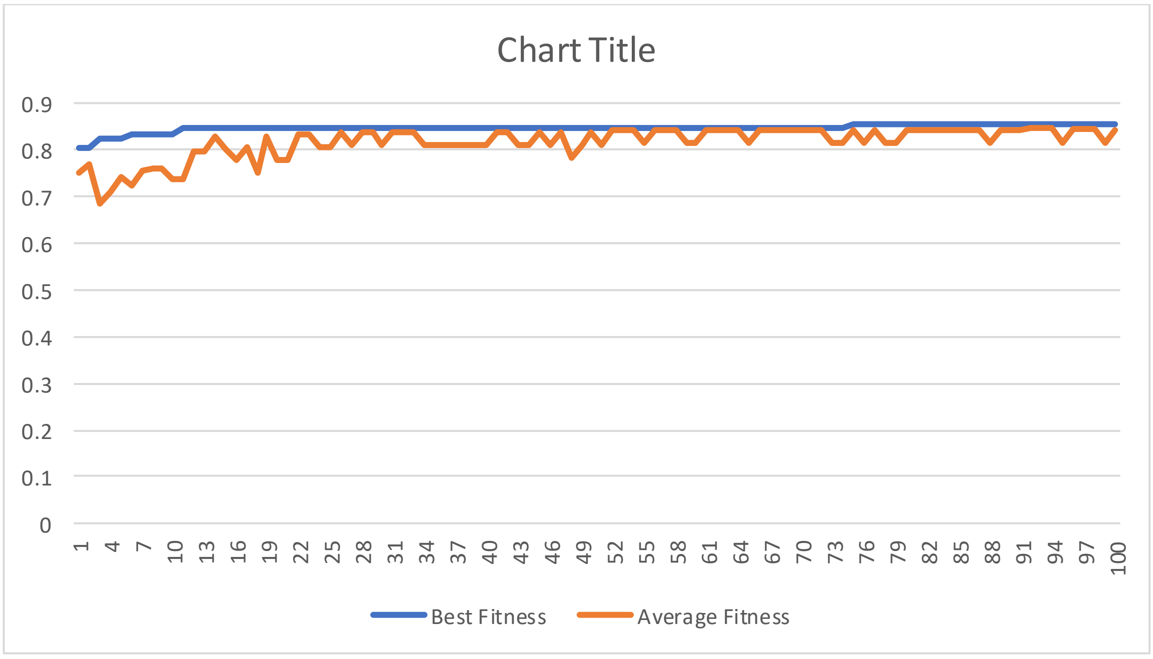


Graph 1.2

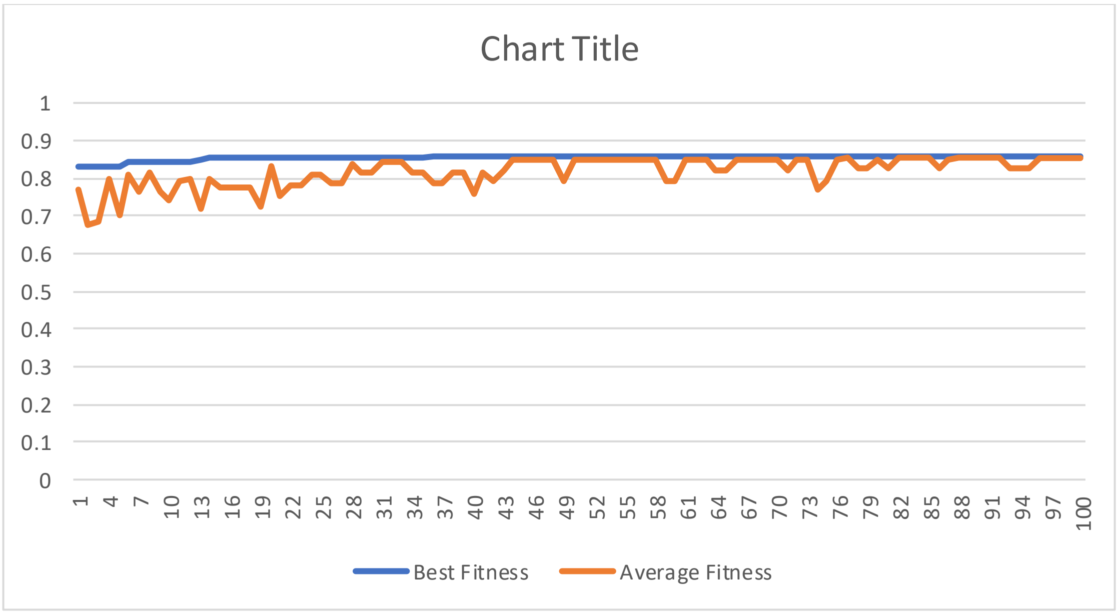


Graph 1.3

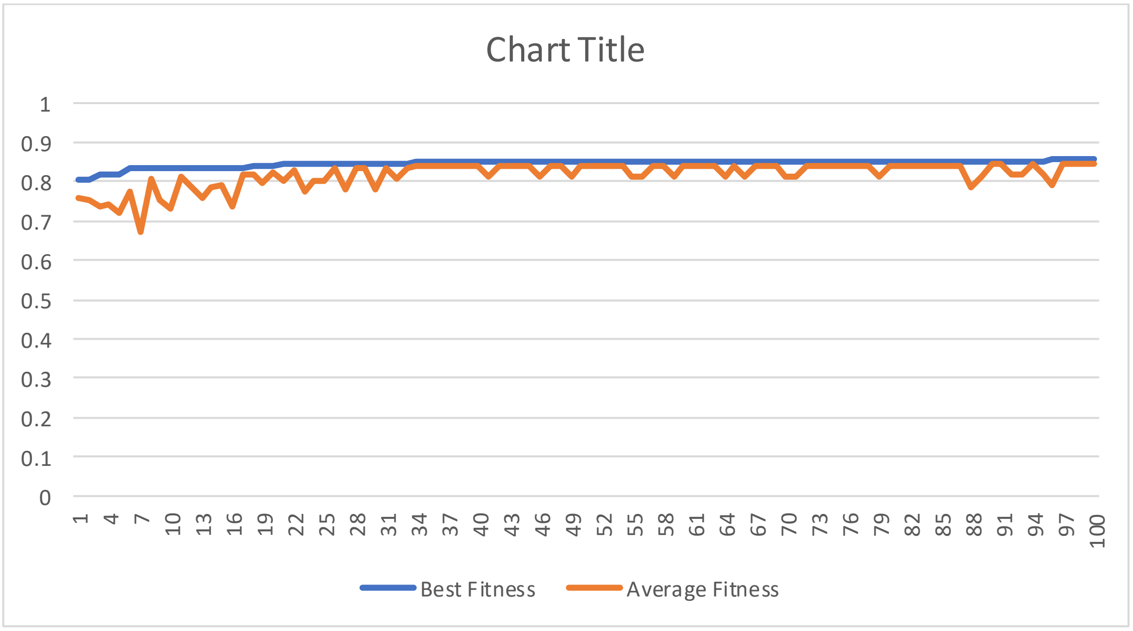
The three graphs above shows the result for 10 populations and 0.9 mutation values. From the results, Graph 1.1 shows the better evolutions than graph 1.2 and 1.3. This is because, the average fitness values are increasing at 100th generations. But it still no the best evolutions since the best fitness values is increasing at 29th generations and starting shows the constant line from 30th generations until 100th generations.



Graph 2.0

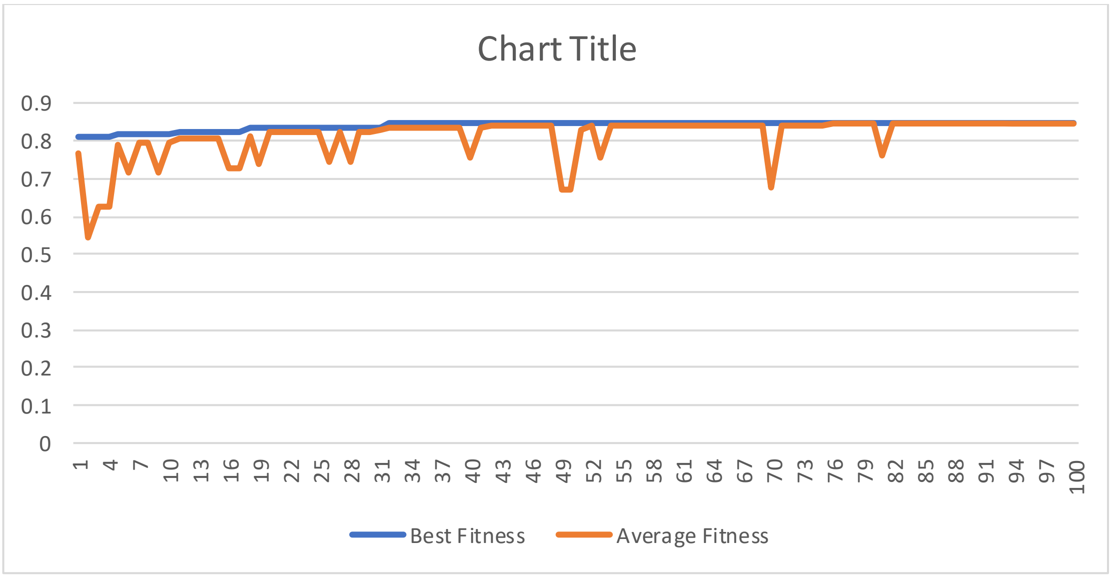


Graph 2.1

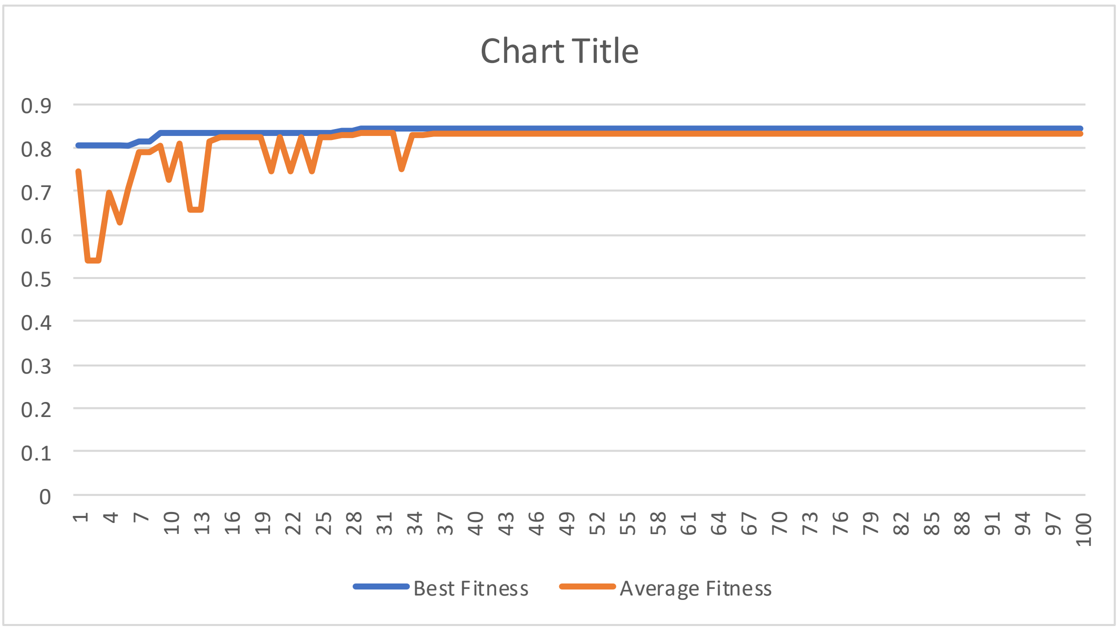


Graph 2.2

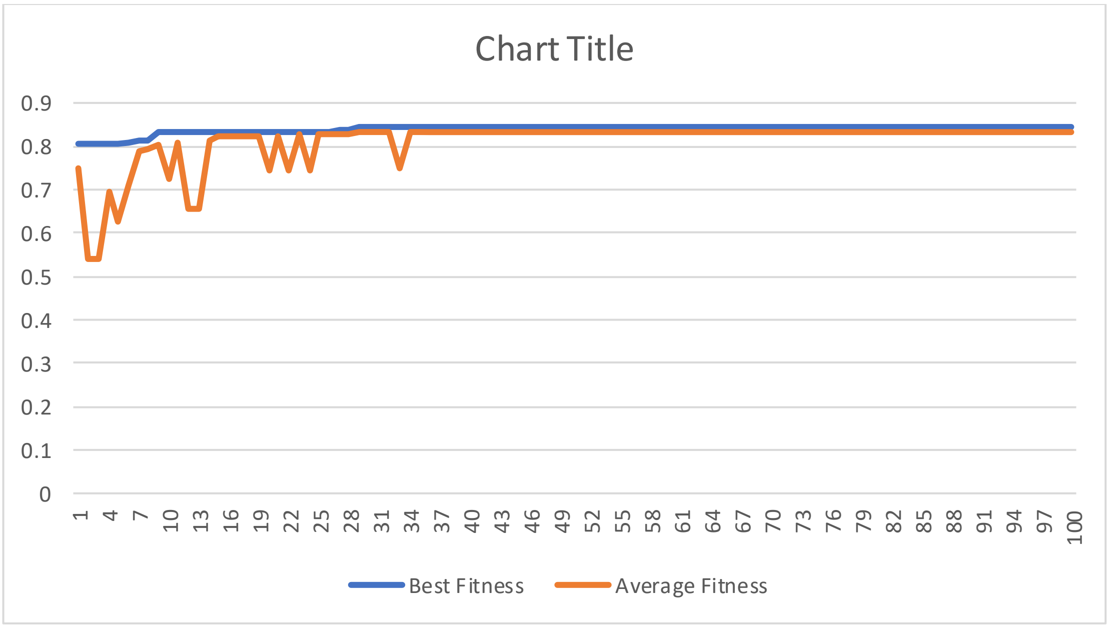
The three graphs above shows the result for 30 populations and 0.9 mutation values. From the results, all three graphs show a better evolution. This is because, the average fitness values are increasing at 100th generations. The best fitness also shows increasing bit by bit but in certain generation the values are constant.



Graph 3.1



Graph 3.2



Graph 3.3

The three graphs above shows the result for 10 populations and 0.3 mutation values. From the results, all three graphs do not show the better evolution. Even though the values of average fitness are increasing, but the results are constant started on certain generation until 100th generation. The best fitness also shows increasing but only a little bit and constant until 100th generation.

To conclude the analysis from the all graphs above, our teams observed that the values of 30 populations and 0.9 mutation gives the best results of evolutions. The values of best fitness and average fitness shows the increasing at 100th generations.