KING SAUD UNIVERSITY
COLLEGE OF COMPUTER AND INFORMATION SCIENCES
INFORMATION TECHNOLOGY DEPARTMENT

جامعة الملك سعود
King Saud University
كلية علوم الحاسب والمعلومات
قسم تقنية المعلومات

IT426
Artificial Intelligence Systems
Riyadh Season Trip Planner Using GA Optimization

Prepared by

Student name:	**ID:**
Nouf Alsadhan	
Lama Alshaya	
Jumanah Aldawsari	
Aljawharah Alzamil	

Supervised by

L. Reem Algifary

1. Solution Representation

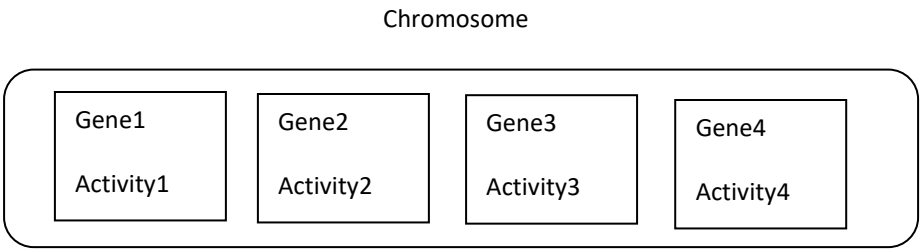
The code is a genetic algorithm (GA) designed to help tourists plan a tour by choosing activities that meet their preferences, we used python programming language. The GA uses a random selection of activities from a pre-determined list and uses optimization techniques to find the best solution for the tourist. The solution representation in this code is a chromosome, and it is considered as a one possible solution, which is a list of activities selected for the tour. Each chromosome has a set of genes, which is a list of tour activities, and each gene represents a single tour activity.

For example:

Chromosome1: has 5 genes which is 5 activities

Chromosome2: has 4 genes which is 4 activities

The user first will be asked to enter the budget, num of activities and the type of activities, Then each chromosome will be evaluated(wither it suits the preference of the user) using the fitness function.



Initial population

We used a function to create the initial population, it creates a list of Chromosome objects(tours) with randomly selected genes(activities) from the list of activities, this function takes the population size as a parameter and starts a loop to create the initial population by selecting a random sample from the list specified and creating a chromosome which is a possible tour.

2. Fitness function

To calculate the fitness function we used a function that calculates a score for the tour based on how well it fits certain criteria. The criteria are the total budget for the tour, the type of experience(that has more weight than the other two) the user wants (exciting, shopping & restaurants, or nature), and the number of activities in the tour. The method returns a score between 0 and 1, with 1 being the best fit by calculating the overall fitness using the weighted sum.

For the budget, we calculated the total budget for the chromosome then we compared it with the budget that the user entered, if it is more then the solution is not suitable.

For the type of experience, we calculated the number of types in the chromosome that matches what the user entered and we divided it by the number of all activities in the chromosome(genes).

For the number of activities, if the chromosome has number of activities exactly like what the user entered, then it is going to be 1, otherwise we will calculate the probability.

Lastly with all the result from the above steps, we will calculate the whole fitness function for the chromosome(tour) by using the weighted sum, that mean we will multiply each weight with its value.

For example:

Chromosome1: Total budget is 1000, number of activities 4, it has 2 Exciting,1 shopping and 1 nature

Chromosome2: Total budget is 1300, number of activities 5, it has 2 Exciting,1 shopping and 2 nature

The user input: Total budget is 1200, number of activities 4, the type is exciting

By calculating the fitness, the second chromosome will not be suitable since the budget is more than the user can afford, so in this case the first chromosome will have the best fit.

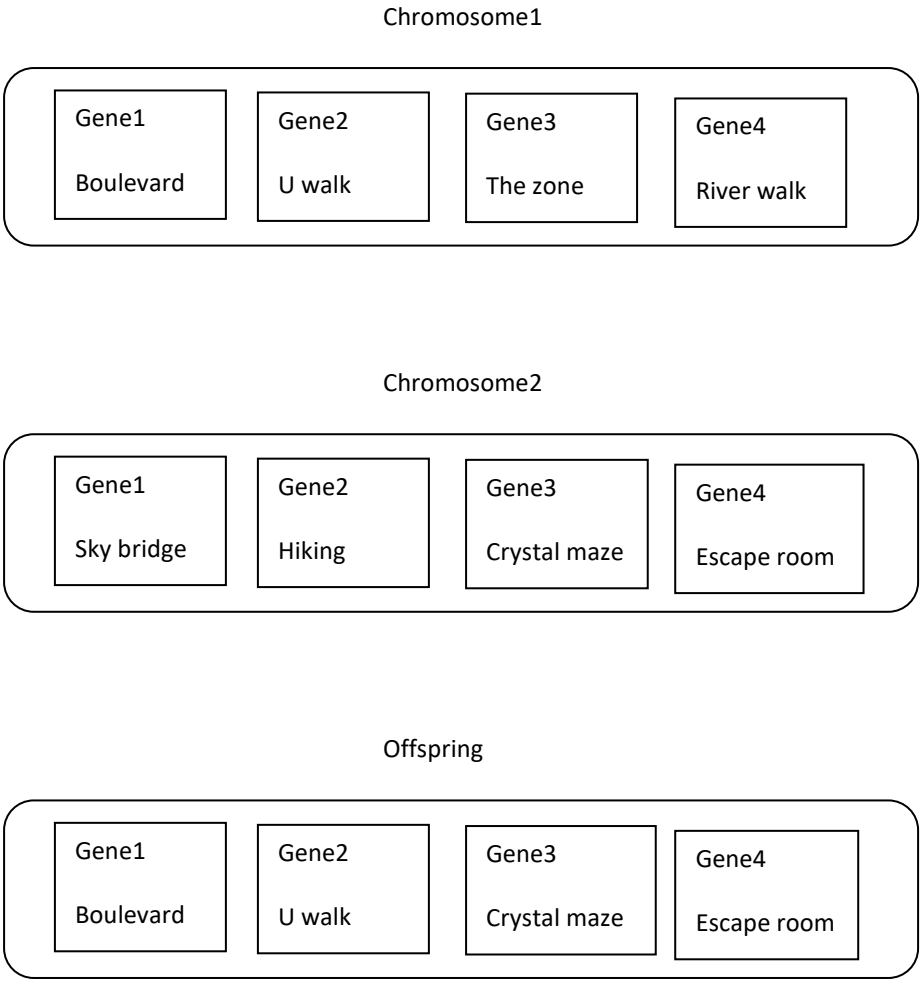
3. Genetic operators

- Crossover

Crossover is a genetic operator that combines two chromosomes to create a new chromosome. In our solution, the crossover operator uses the one-point crossover method, which randomly selects a point in the chromosomes and swaps the genes between them.

We will send two parents to the method as parameters to calculate and find the offspring(the child), the offspring will be the first half of the genes for the first parent and the second half of the genes for the second parent, then for each offspring has been created we will calculate its fitness function

For example:



- Mutation

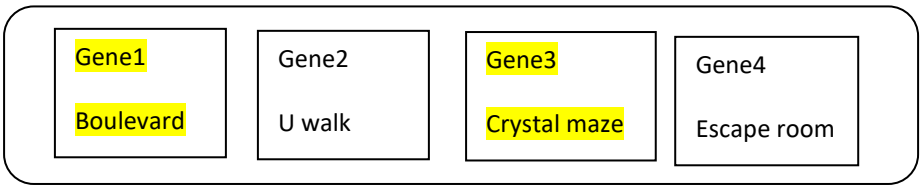
Mutation(طفرة)is a genetic operator that introduces random changes in the chromosome. In this code, mutation is performed by randomly selecting a gene in the chromosome and changing it to a different activity.

The mutation will happened if the random generated number for each chromosome(between 0 and 1) is less than the mutation rate(which in our case is .3), in this case the mutation will happen, the function will take this chromosome then Mutate the chromosome by randomly swapping two gene.

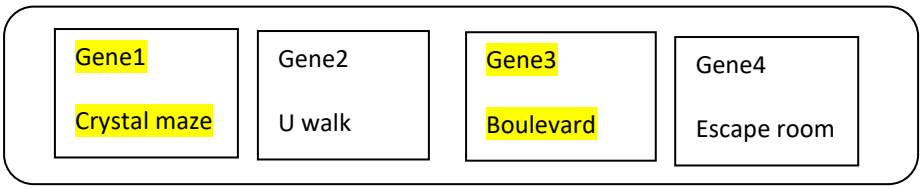
Since mutation introduces diversity into our population, it prevents chromosomes within the population from becoming similar to one another, increasing the likelihood of reaching a global optimal solution rather than a local optimal solution.

The mutation method is shown below

Before mutation



After mutation



- Selection by Roulette Wheel

Roulette wheel selection is a selection method that is used to select the chromosomes for the next generation based on their fitness. In our code, the roulette wheel selection method calculates the probability of each chromosome being selected based on its fitness and selects the chromosomes accordingly.

- Replacement

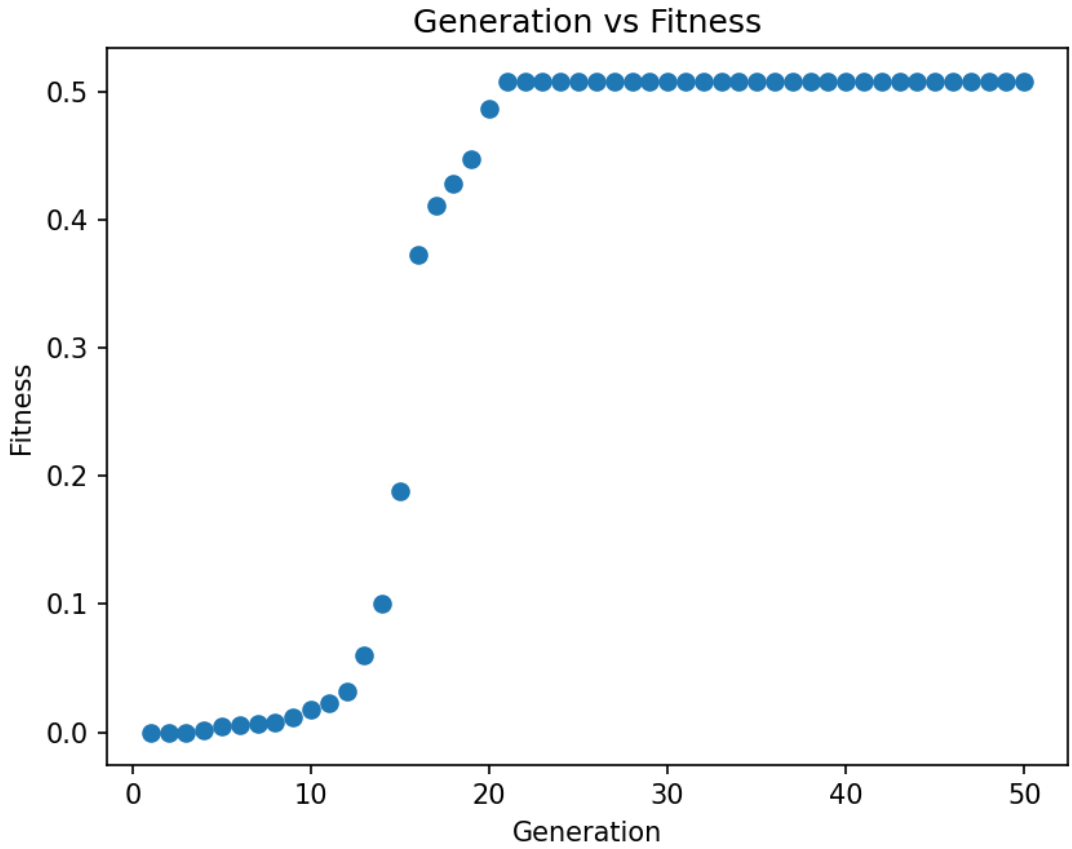
Replacement is a genetic operator that replaces the old population with the new population after the genetic operations have been performed. In our code, the replacement operator selects the new population by selecting the best chromosomes from the current population and the offspring generated from the genetic operators.

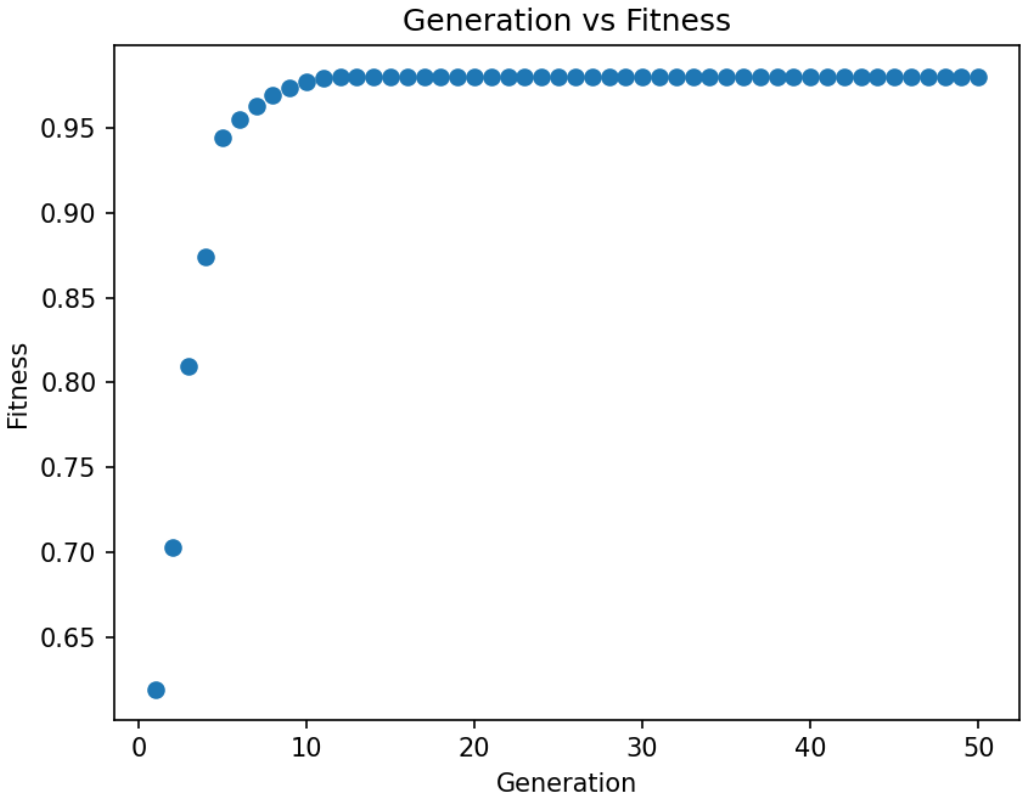
4. Termination condition

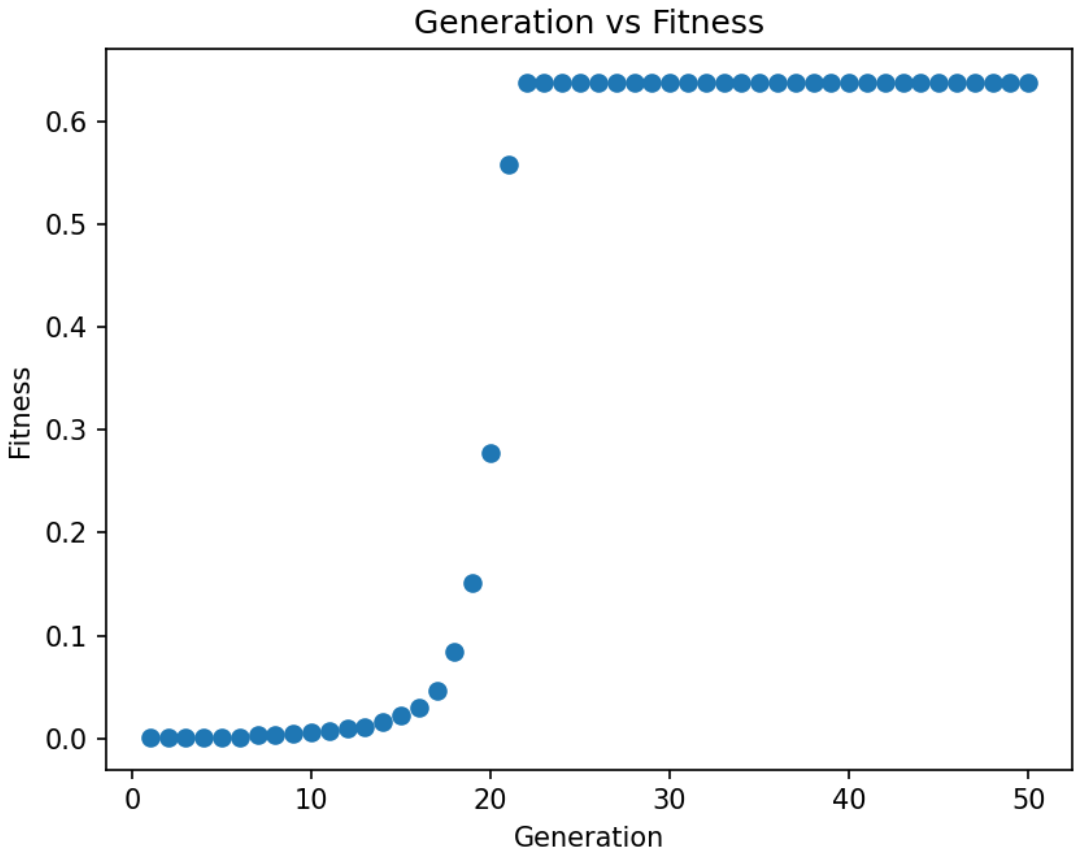
For the termination condition we tried different number of generations and different type of tourist preference to see exactly how it will affect the plot.

5. Result

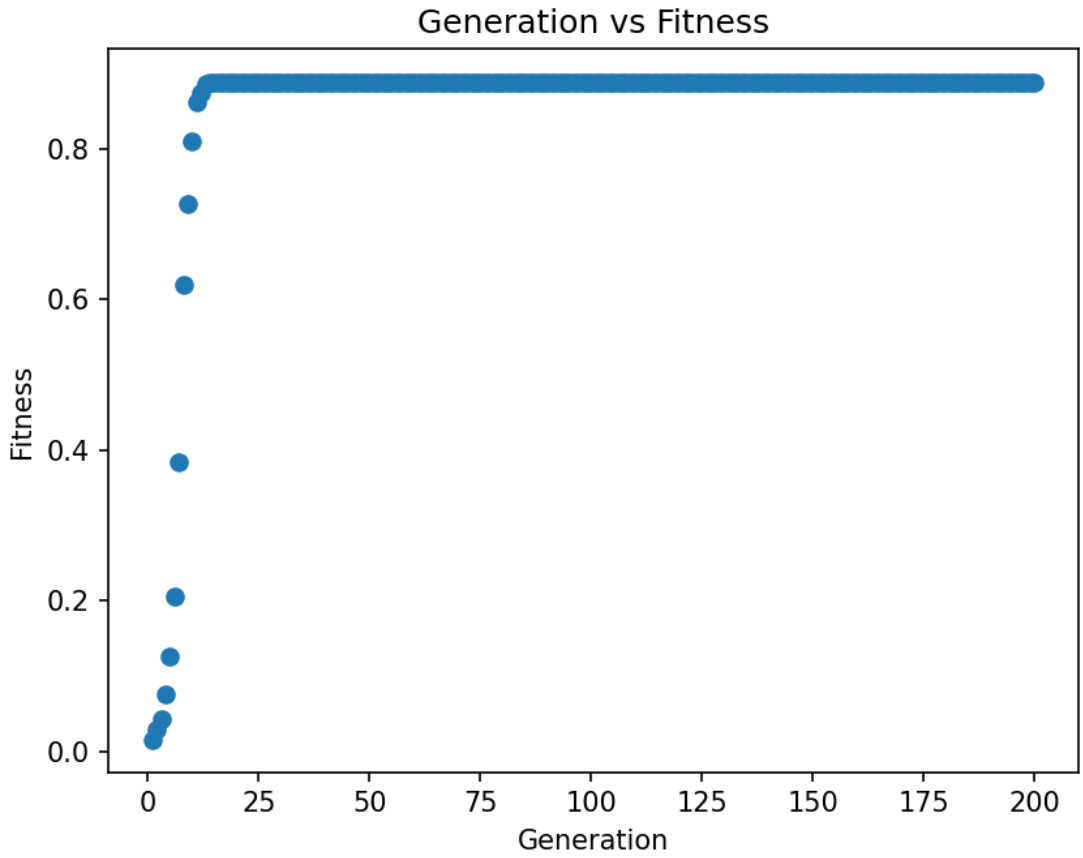
Generations=50

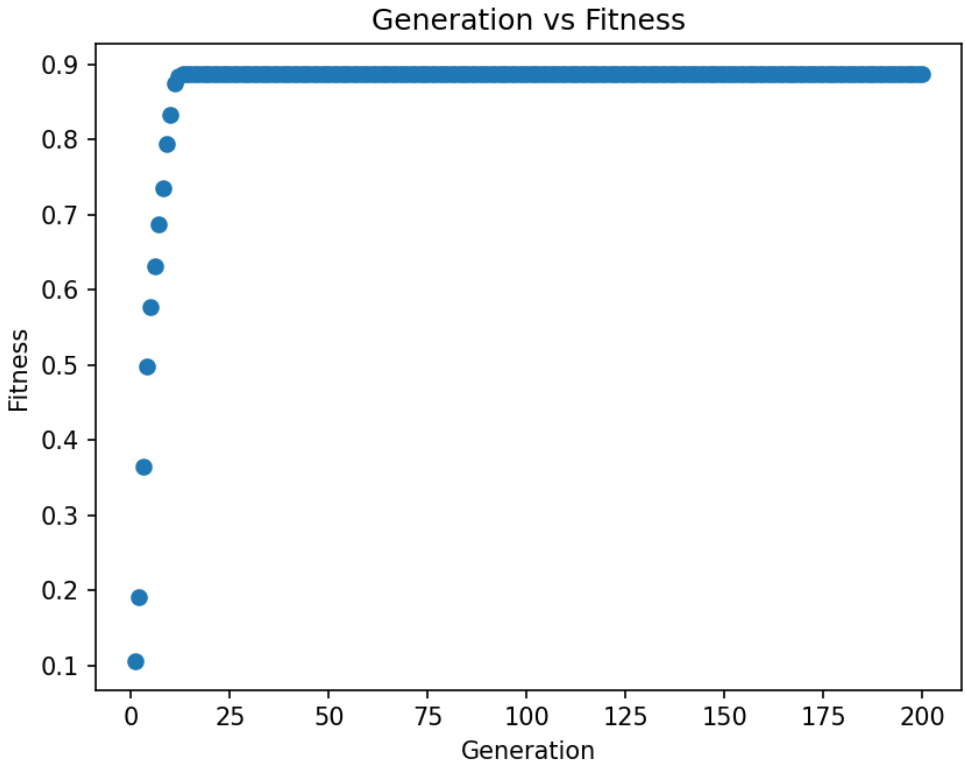


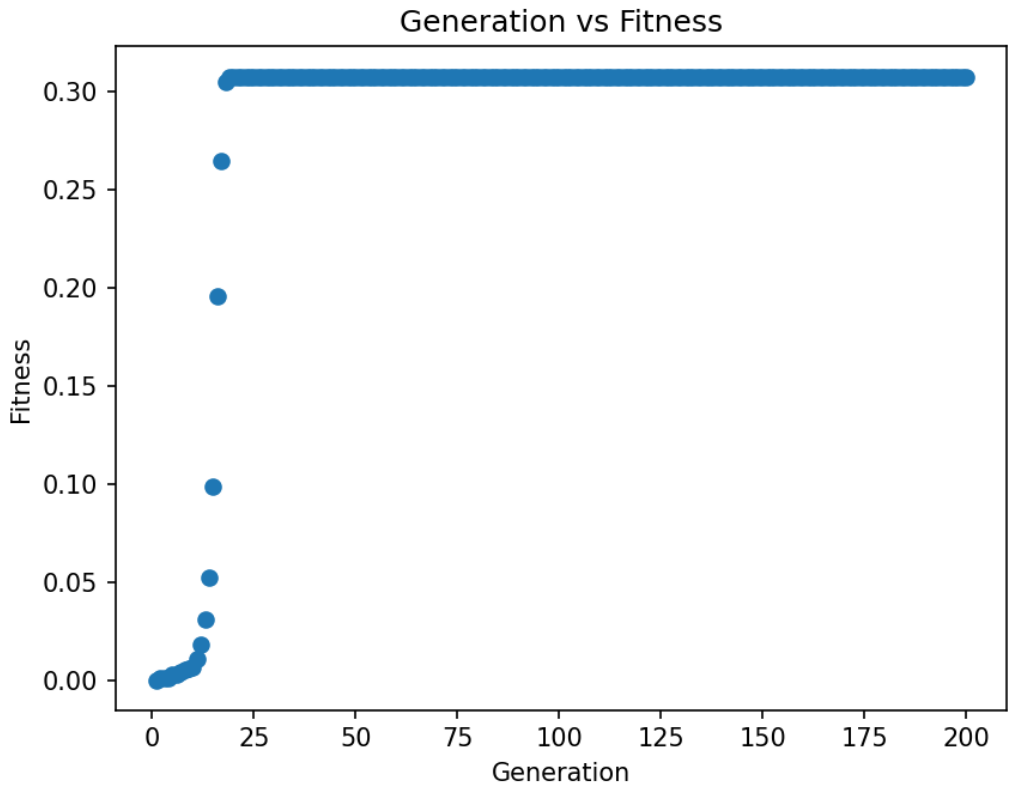

Generation vs Fitness
Fitness
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0
10
20
30
40
50
Generation


Generation vs Fitness
Fitness
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
10
20
30
40
50
Generation

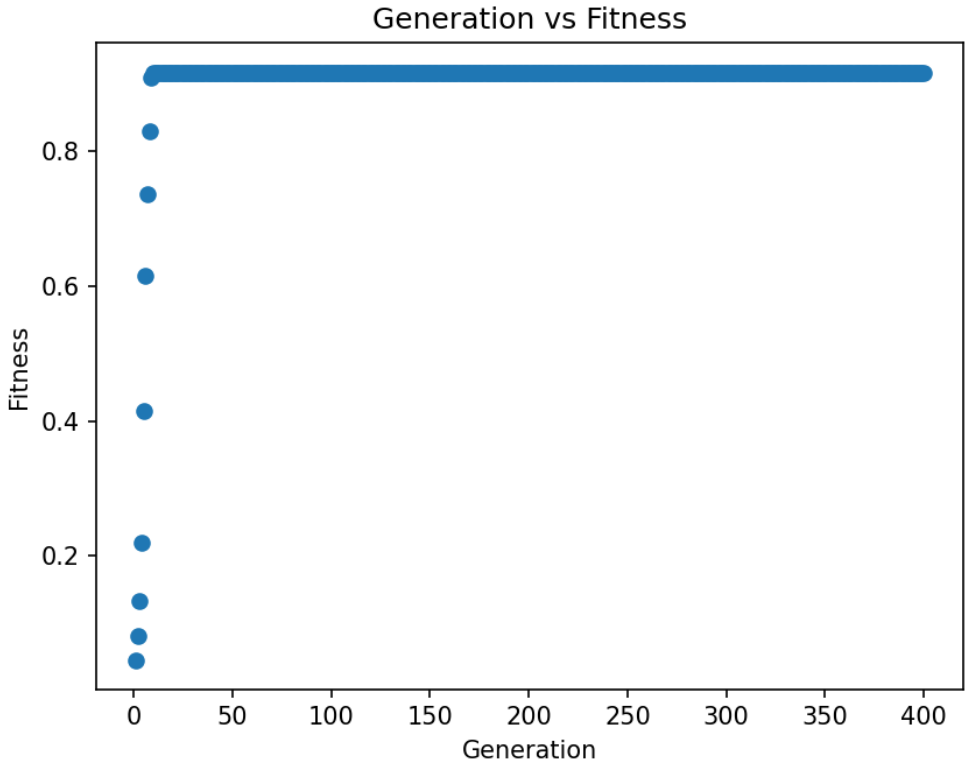
Generations=200

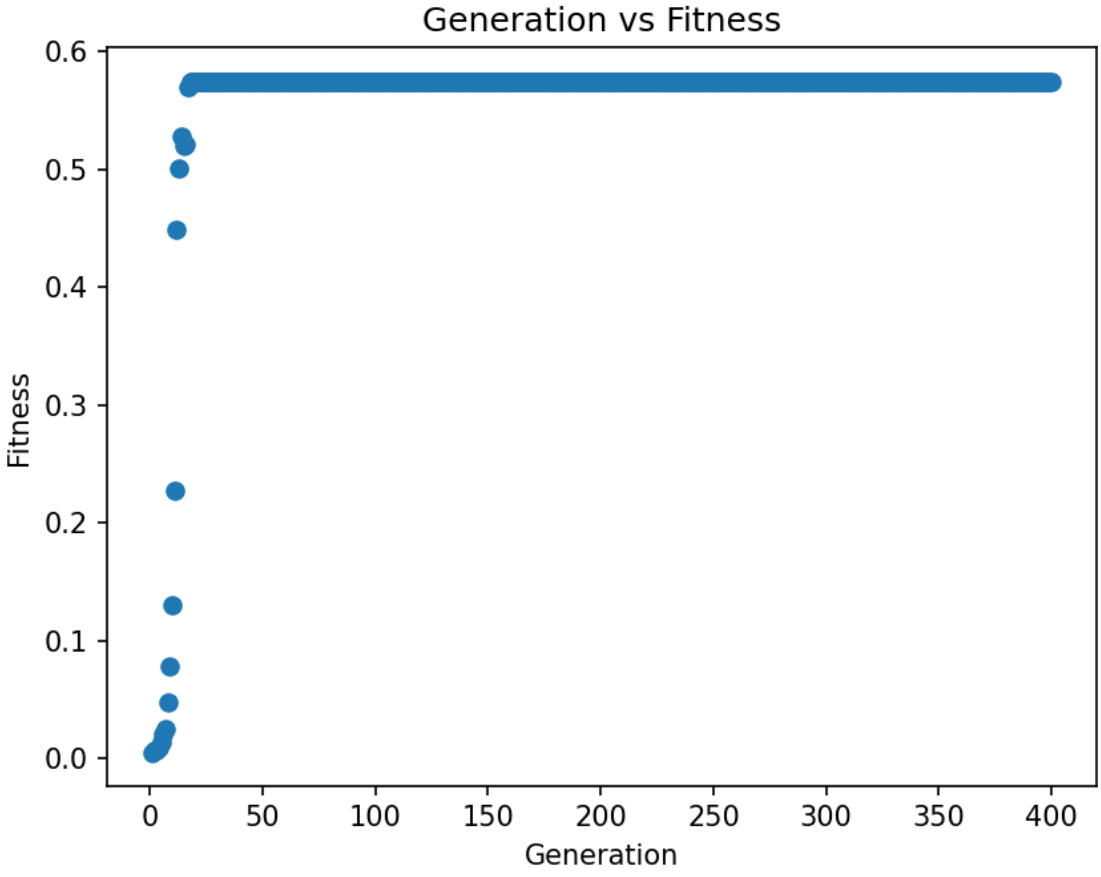


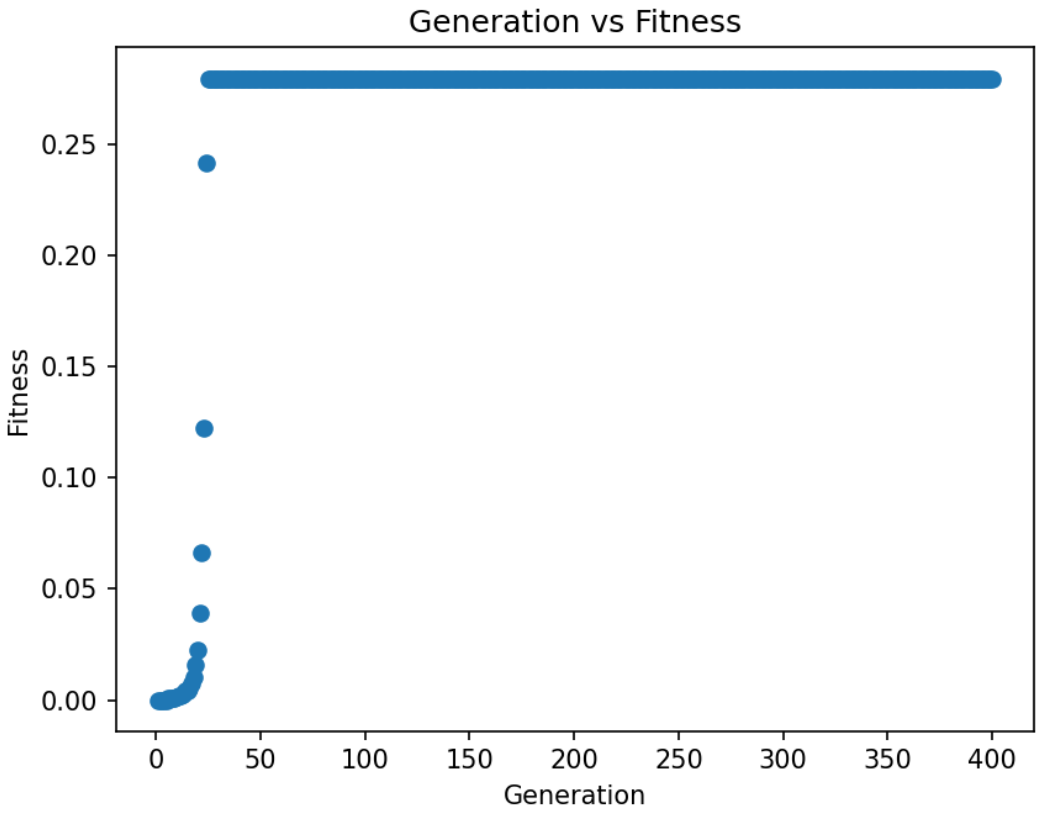




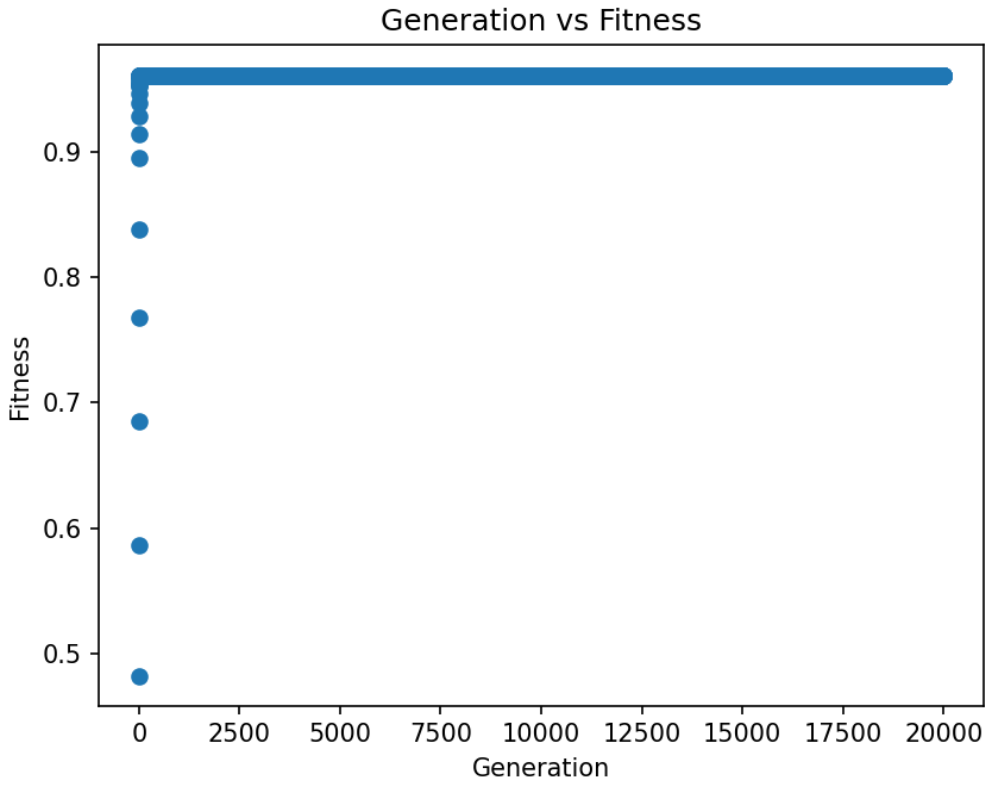
Generations=400


Generation vs Fitness
Fitness
0.2
0.4
0.6
0.8
0
50
100
150
200
250
300
350
400
Generation


Generation vs Fitness
Fitness
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
50
100
150
200
250
300
350
400
Generation



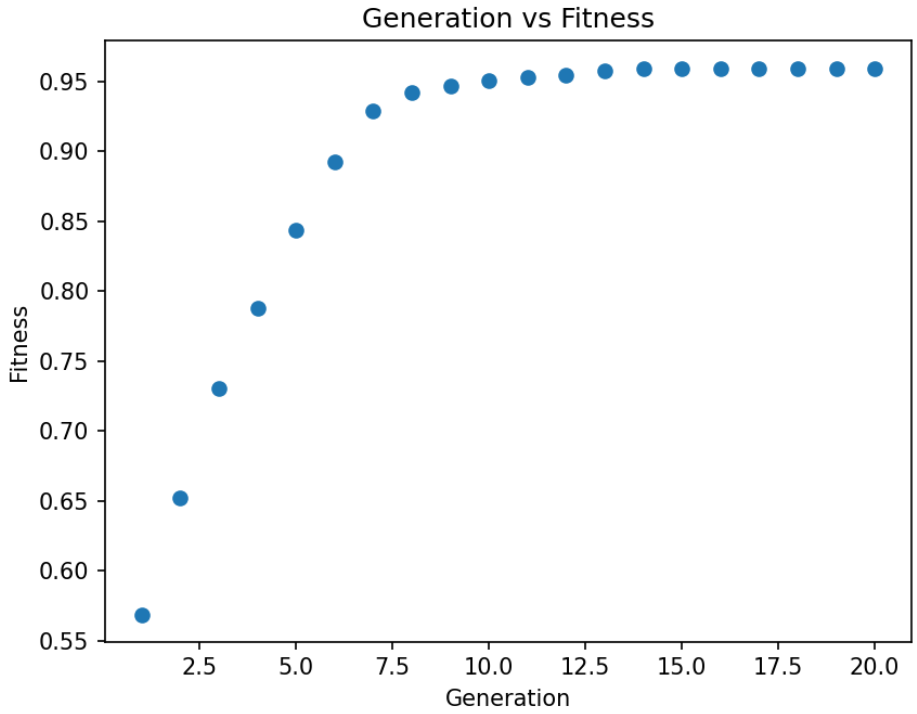
Generations=20000

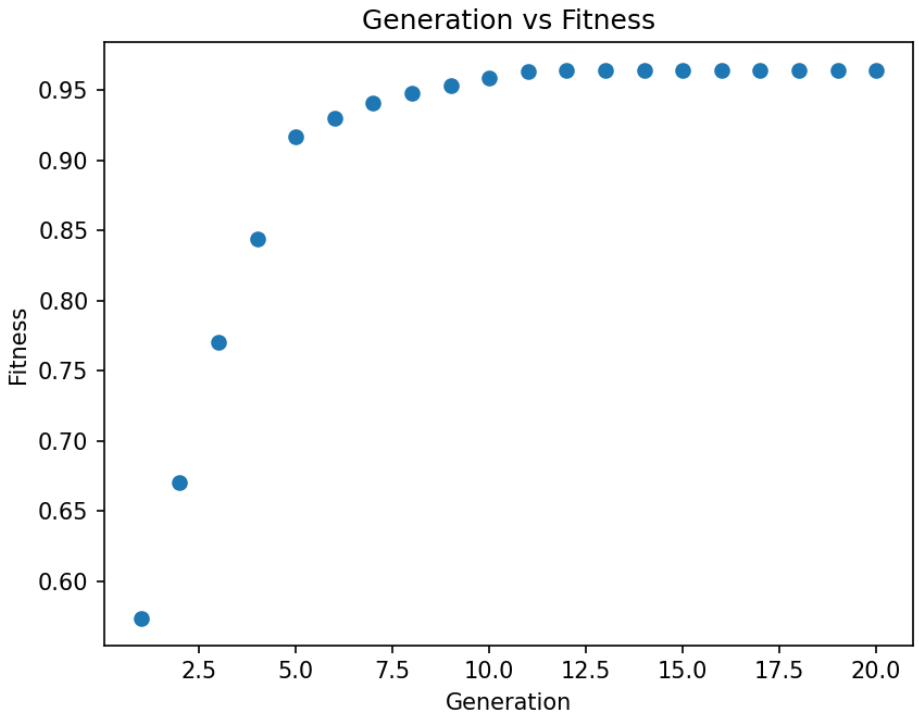


6. Analysis

We noticed in every generation we tried, graph represents the evolution of the fitness function over time and shows how the fitness improves as the generations increase.

But we also noticed that in every generation, that fitness start to become constant after almost 20 generations, so we tried to put our generation as 20 to see the result.





We noticed that the fitness start becoming constant and generation 20, and we thought if the GA continue running even though the fitness is constant it is going to be waste of time, so we concluded that the best number of generations to run our GA is 50, in case there are better solutions that might appear after the 20^{th} generation.

The GA in our code is used to find the optimal tour itinerary based on the budget, experience type, and the number of activities. The code uses a combination of genetic operators and fitness functions to find the best solution.

The result shows that the GA is able to find an optimal solution after a few generations, and the fitness function improves as the number of generations increases. This shows that the GA is an effective method for solving this type of problem.

In conclusion, the GA is an effective tool for helping tourists plan their tours. The code presented here is a simple implementation of a GA that solves a complex problem by using optimization techniques to find the best solution.We think that there are many ways that we can improve this problem. The GA can be improved by incorporating additional criteria into the fitness function, such as distance between activities, time of day, and weather conditions, number of people in the trip and wither there are children or not, these information could lead us to a better solution and a better planned tour and will make the GA even more effective in helping tourists plan their tours and have an enjoyable experience.

7. Experimental settings

First we did run our GA 20 times to see the average fitness.

The Average after running the GA 20 times = 0.8465

We Initialized our first generation randomly in the search range by using the method initialize_population. This method takes population size as a parameter (n), then creates n chromosomes by randomly selecting genes (activities) from the predefined list of activities.

```
def initialize_population(population_size):
    # Initialize a population of chromosomes with random genes
    population = []
    for i in range(population_size):
        genes = random.sample(ACTIVITIES, NUM_ACTIVITIES)
        chromosome = Chromosome(genes)
        population.append(chromosome)
    return population
```