Artificial Intelligence

Report: Template Matching Problem

Instructor: Dr. Junaid Akhtar

Group Members:

1. **Muhammad Hammad Sani**
2. **Sandeep Kumar**

Evolutionary theory to the evolutionary algorithm:

As Human beings and having the ability to observe and think ,we (consciously /unconsciously) observe the natural phenomena in our day to day life and some of us not only just observe them but spend their lives finding an explanation of that natural phenomena (we refer this explanation as theory or interpretation). Then some models are made on the basis of theory and these models have different applications. In this report, we will be exploring one of the interpretations of Diversity that Why there is so much diversity on this planet and will see how this theory has led us to a computational model which helped us to solve a template matching problem.

Diversity and Its Explanation by Darwin:

Why Darwin theory?

There are a number of theories giving explanations of Diversity but we have chosen Darwin theory because he spent a handsome period of his life finding answers to his questions like, Can it be if species were not fixed for all kinds but in fact slowly changed? He tried to be more rational and focused on being more evident by putting aside traditional beliefs.

What is Darwin Theory!

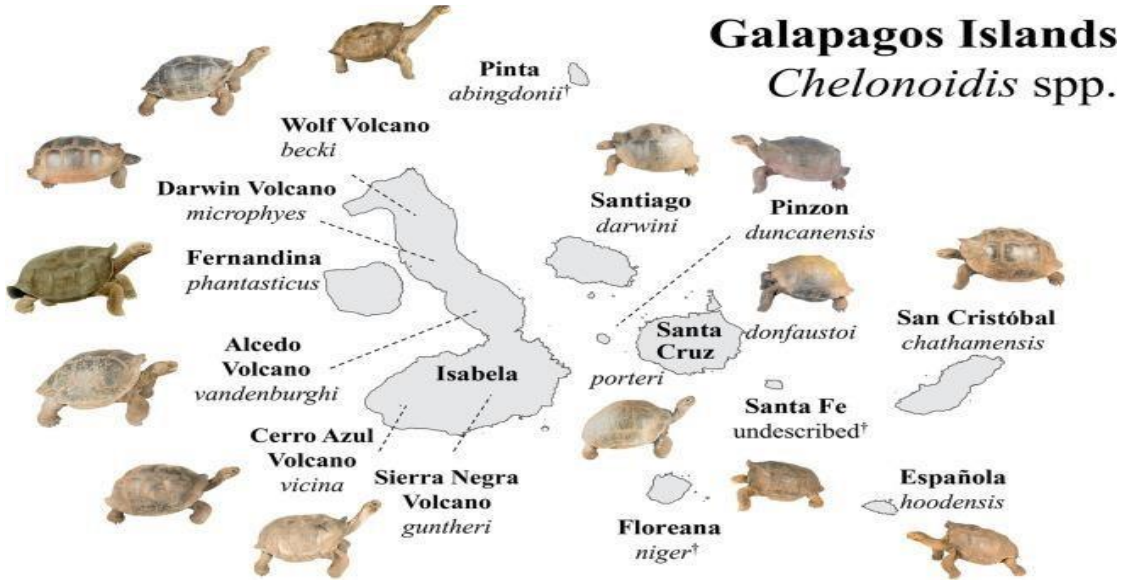
Darwin's theory of evolution is known as Darwinism, which is the detailed explanation of biological evolution. Darwin stated that natural selection of small and hereditary differences that enhance the individual's capacity for survival and growth causes all species of organisms to grow and diversify. After Darwin's book "On the origin of species," which describes how species evolve over generations through the inheritance of physical structure or behavioral features, the general concepts of mutation of species or evolution gained acceptance."[1]

But a simpler definition is,

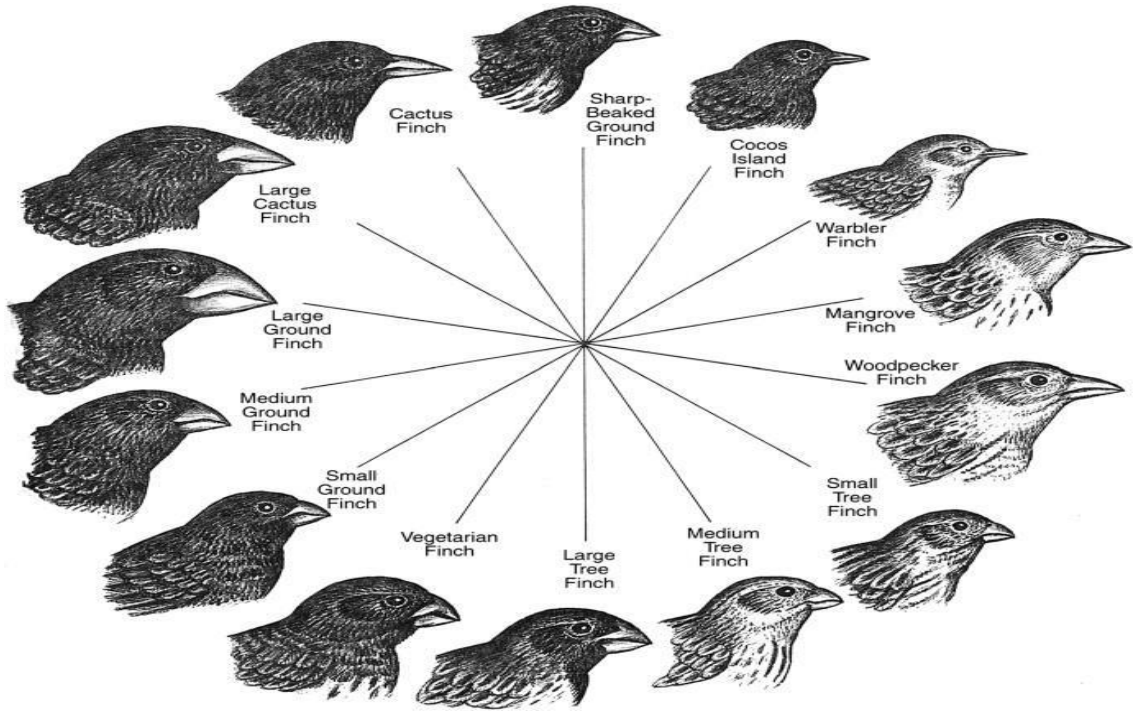
"Species are evolving from simpler form to complex depending upon their environments and taking time to evolve"

A Simple Explanation of the Definition:

Darwin saw tortoises in several locations and noted that they differed in their appearance in each location and argued that the environments to which they are living can cause differences between their appearance.



He also observed the variations in beaks of finches, some finches have long beaks which are used for cracking nuts and he claimed that the finches with the long beak are not able to service in some places. Additionally, he studied a variety of species and looked for connections among them. From the above observations we can define that over a long period of time, one species may gradually change into something different, and we could even force it to be classified as a different species.



Some discoveries like

- **Discovery of Intermediary Species**
  - Part mammal and part reptile
  - Part Bird and part reptile
- **Fossilized remains**

were proved helpful in answering many of the questions raised on his theory like the question Darwin asked, “There should be a connection between not only the same but different species.” was answered when the **discovery of intermediary species** was made and as species are evolving from simpler form to complex form, there should be a lot of time required for this process, was answered by **fossilized remains** discovery.

Diversity is a Natural Reality!

According to our understanding, Diversity is basically the existence of so many species which are different in termsof their

- Appearance
- Ability to communicate
- Ability to survive in different environments

Natural Reality can be refer to a phenomenon that is happening around us continuously and it is connected with nature.

A General Model from Darwin Theory

As we mentioned earlier, species are evolving from simple to complex and their survival depends on different factors like environment.

So, a general model from this theory can be

1. Existence of Species Initially which can be termed as **Population**. The idea here is that these will be the species on which further process by will happen (some of them will evolve and some of them will die)

2. Then from our population, some of them will survive and this survival is done by Nature. The fittest of these species with respect to their population will survive. This is called **Natural Selection**

3. From Natural Selected, a contribution towards new generation will be made. This new generation will be result of combination of pairs from selected population. This phenomena is called **Crossover**. The new generation will have similarities to pairs, they are made of. Environment maybe result in altering the properties of child from parent pair. This is called as **Mutation**

This Model is the basis of Evolutionary Algorithms.

Evolutionary Algorithm:

Evolutionary algorithm is based on the model of Darwin theory of Evolution. In evolutionary algorithm, there are certain things we do like

- Population Initialization
- Fitness Evaluation:
  - Determining the good ones
- Selection
  - Selecting individuals which will contribute to the future population
- After selection, recombining them, how they will evolve to future population
- Mutation

```
populationInitialization() -> # to initialize the population

fitnessEvaluation()        -> # checking their fitnesses
selection()                -> # selecting the fittest from them
newGeneration()            -> # making new generation by crossover and mutation
    crossover()
    mutation()

# then doing fitnessEvaluation of newGeneration, doing selection and
# checking whether a certain criteria is met and according to criteria
# checking whether our population has evolved to criteria we want or not.
# this process goes again and again until criteria is met or our set limit is not exceeded.
```

Application

One of the benefits of this model is that it helps us in solving search problems in a more optimized form. We solved a template matching problem using an evolutionary computational model. In this case we take images which are represented as 2D points which is basically a 2D list. We give a small template taken from large template and then we need to search from a large template.

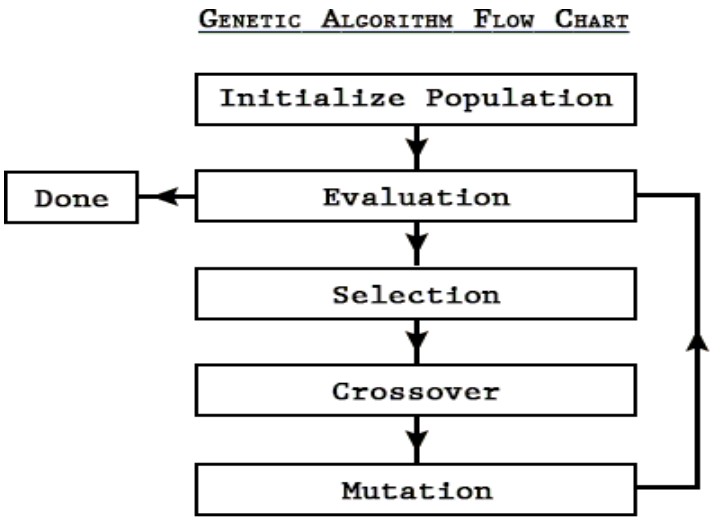


Figure 2

Picture taken from google images

1. Population Initialization

First of all, we made a function which takes the size of total population we want to generate and then creates a list of (x, y) tuples where x is less than big image rows and y is smaller than big image cols.

```
def giveRandomNumber(maxRange):
    return int(random() * maxRange + 1)


def populationInitialization(groupCols, groupRows, populationSize):
    currentGeneration = []
    while (len(currentGeneration) != populationSize):
        x, y = (giveRandomNumber(groupRows), giveRandomNumber(groupCols))
        if (x + boothiRows ) < groupRows and (y + boothiCols) < groupRows:
            currentGeneration.append((x, y))

    return currentGeneration
```

2. Fitness Evaluation:

Then fitness values of each point were calculated. It takes values of the current generation and compares it with a small image and returns the values between -1 to 1.

```
def findCorelationOfTwoMatrixes(slicedMatrixFromBarriImage, boothi):
    x = slicedMatrixFromBarriImage
    y = boothi
    return ((x - x.mean()) * (y - y.mean())).mean()/ (np.std(x) * np.std(y))


def fitnessEvaluation(currentGeneration, barriImage, boothi):
    fitnessValues = []
    for points in currentGeneration:
        x, y = points
        slicedImageFromBarriImage = barriImage[x:x +boothiRows, y: y+boothiCols]
        fitnessValues.append(findCorelationOfTwoMatrixes(slicedImageFromBarriImage, boothi))
    return fitnessValues
```

3. Selection

This function returns a list of sorted values with respect to fitness values.

```
def selection(currentGeneration, fitnessValues):
    print("I am length of Gen", len(currentGeneration))

    tempArray = []
    for i in range(len(currentGeneration)):
        tempArray.append([fitnessValues[i], currentGeneration[i]])
    tempArray.sort(key=lambda row: (row[0]), reverse=True)
    return tempArray
```

4. New Generation

We take a crossover in multiple ways to create a new population.

1. we cross the x of parent1 with y of parent2 and then y of parent1 with x ofparent2 without converting into binary.

Secondly, we convert the x1,y1 and x2,y2 into binary and then initialize a random number and take a random cut. After doing this we merged the left side of x1,y1 with the right of x2,y2. And left of x2y2 with right of x1y1. Then converted back them to decimal points

```
def newGeneration(selectedGeneration, groupRows, groupCols):
    newGeneration = []
    newGeneration.append(selectedGeneration[0][1])

    lastPoint = selectedGeneration[-1][1]
    selectedGeneration = giveMeCoordinates(selectedGeneration)
    for i in range(1, len(selectedGeneration) - 1, 2):
        newPoints = crossOver(selectedGeneration[i], selectedGeneration[i+1])
        newGeneration.append(newPoints[0])
        newGeneration.append(newPoints[1])

    newGeneration.append(mutation(lastPoint))
    newGeneration = mutationOfPoints(newGeneration)
    return newGeneration
```

Experimentation we did

Experiment 1:

First we thought why not to just try increasing the population size and try to achieve the maximum threshold value which we kept at **0.85. No cross over and mutation was done.** So, we tried this way.

Hypothesis:

Thought in terms of probability way and find that (x, y) combination whose fitness value is greater than threshold (kept 0.85) was almost impossible to achieve.

i) Result when

- Initial Population was set to 100.
- Generation was set to 100
- Threshold value to 0.85.

Unsuccessful to achieve threshold value.

ii) Result when

- Initial Population was set to 1000.
- Generation was set to 100
- Threshold value to 0.6.

When initial population size was increased and threshold was decreased. Sometimes, I was able to achieve a threshold value and sometimes not.

Observation:
I observed that whatever the generation size be, as there is no crossover and mutation. Same point will be repeated and the maximum fitness value achieved will remain the same for every number of generations.

Experiment 2:

As we were writing code step by step and experimenting on the basis of code written, we though let's try crossover in different ways.

Hypothesis 1 :

First we tried a simple way of doing crossover. We just did the replacement.

Let's take (x1, y1) and (x2, y2). We just replaced x1 with x2 and rest is same. So, (x1, y1) and (x2, y2) will become

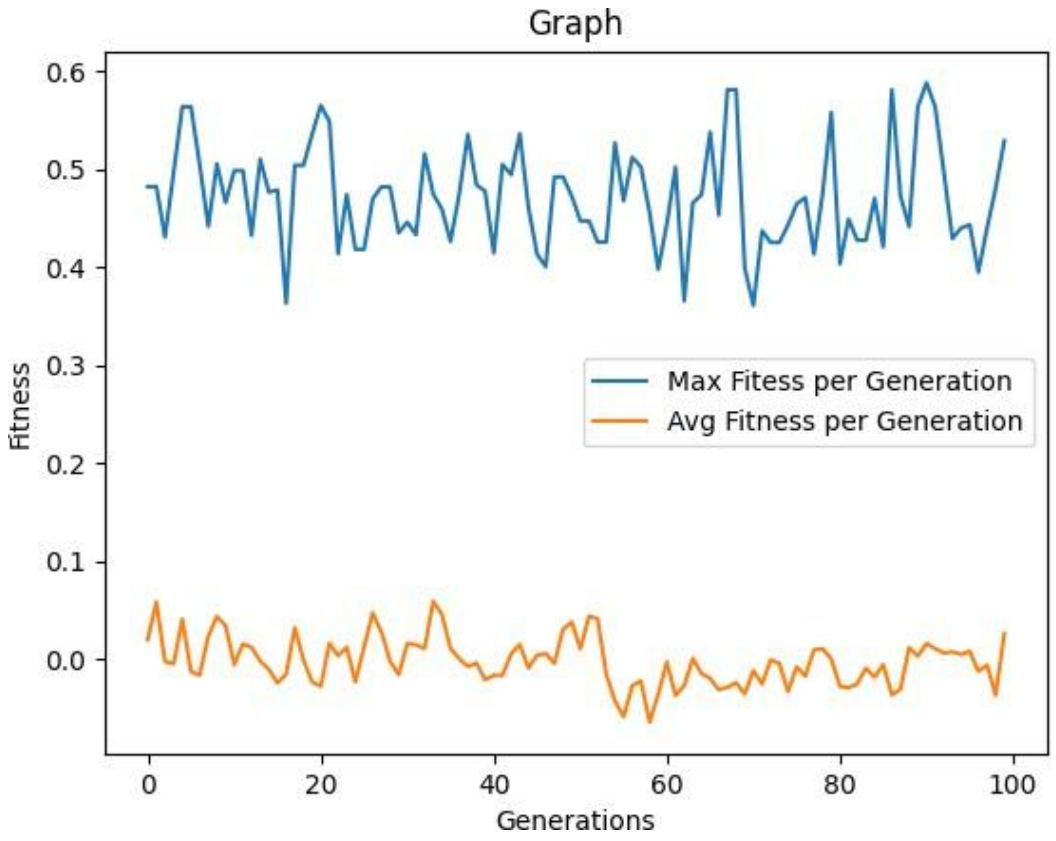
(x1, y2) and (x2, y1).

We thought as new points are generating, we will achieve better results.

Results

i) Result when

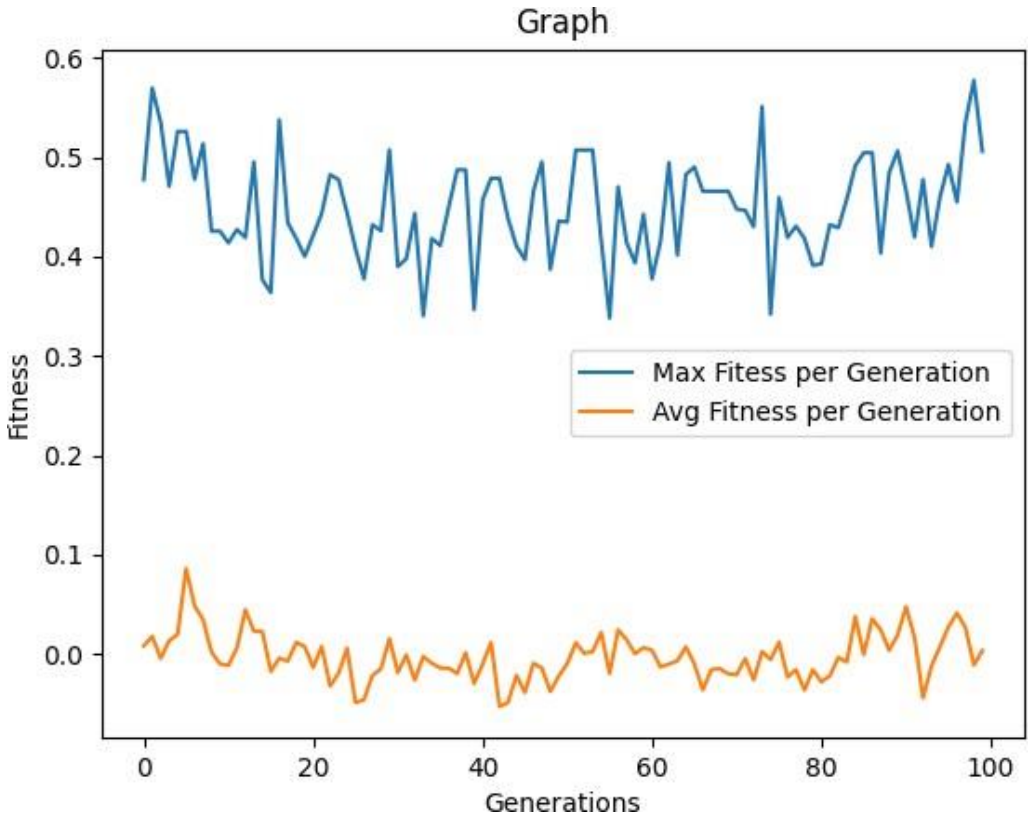
- Initial Population was set to 100.
- Generation was set to 100
- Threshold value to 0.85.



Maximum threshold achieved = 0.58

ii) Result when

- Initial Population was set to 1000.
- Generation was set to 100
- Threshold value to 0.85.



Max threshold achieved = 0.577

Observation:

As crossover is happening in such a way that only the points are just interchanging, therefore, no actual variation is happening and we are unable to achieve successful results

Hypothesis 2:

Now, we did single point crossover with no mutation.

Chromosome1	11011|00100110110
Chromosome2	11011|11000011110
Offspring1	11011|11000011110
Offspring2	11011|00100110110

Single Point Crossover

Convert coordinates into binary, made a random cut, and then make offspring from them.

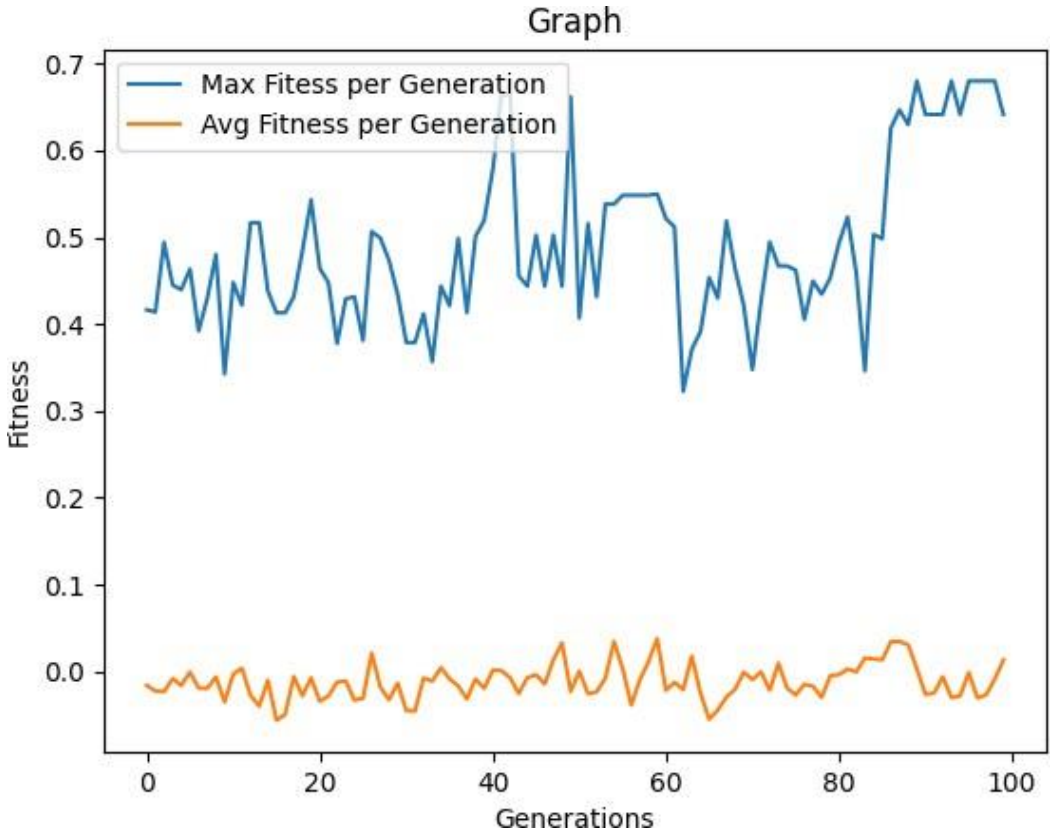
Observation 2:

The results were better but were not able to meet the threshold value of 0.8. The reason we think is again same kind of points repeating during crossover.

Threshold achieved = 0.67

i) Result when

- Initial Population was set to 100.
- Generation was set to 100
- Threshold value to 0.85.



Experiment 3:

Hypothesis:

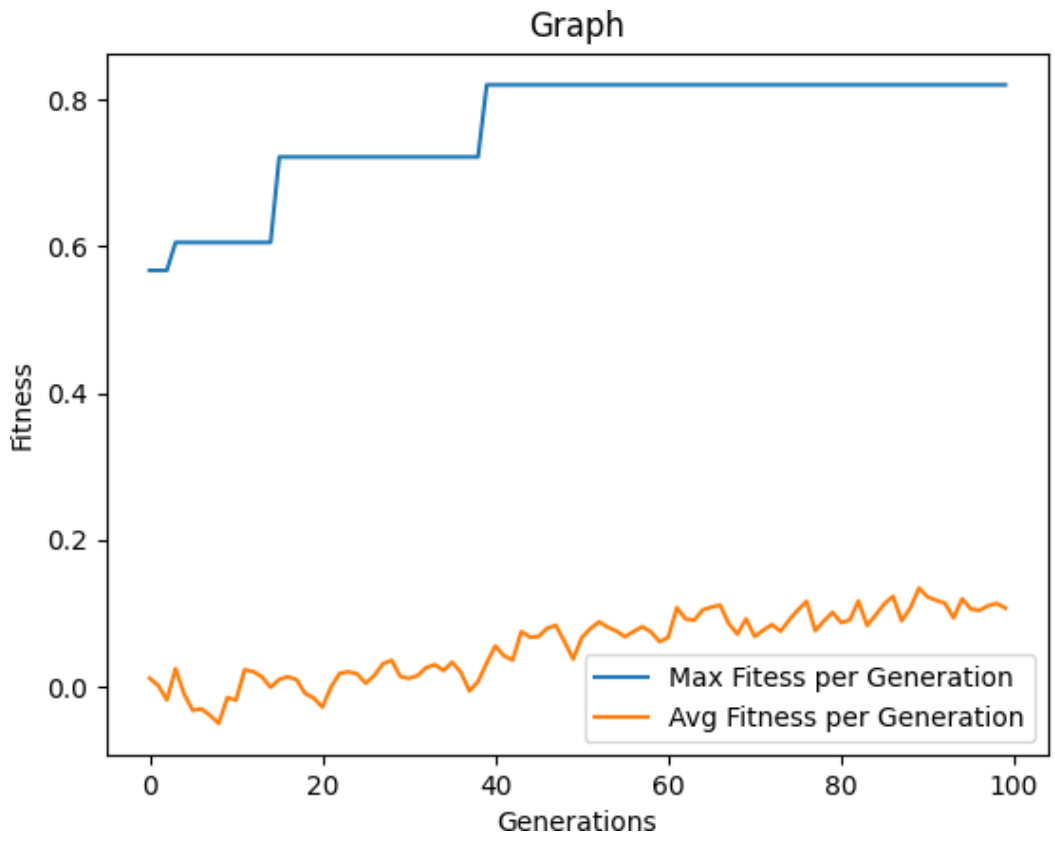
As we have tried crossover, now we tried mutation with 2% rate. 2 out of 100 were mutated when crossover was done.

We observed that our best point with best correlation was not preserved and we preserved that too and hoped to achieve better results.

i) Result when

- Initial Population was set to 100.
- Generation was set to 100
- Threshold value to 0.85.

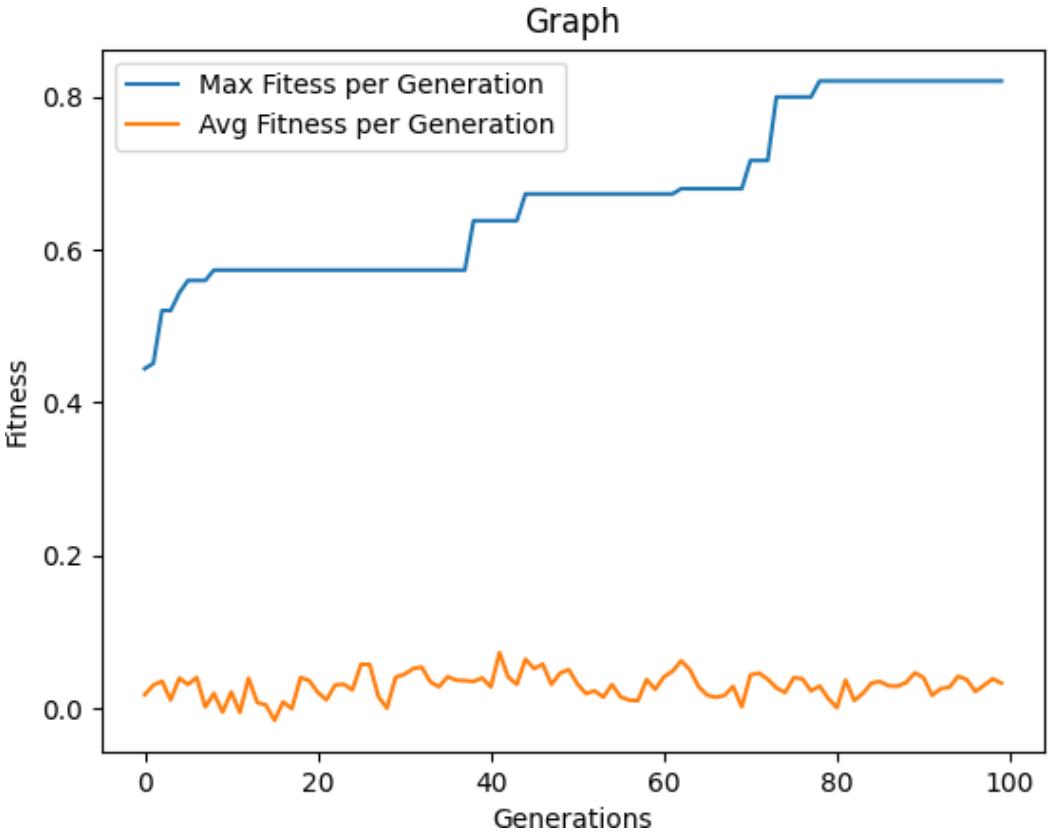
Threshold achieved = 0.82



ii) Result when

- Initial Population was set to 1000.
- Generation was set to 100
- Threshold value to 0.85.

Threshold achieved = 0.82



iii) Result when

- Initial Population was set to 1000.
- Generation was set to 500
- Threshold value to 0.85.

Threshold achieved = 0.93 and boothi found on 493th generation

points = (104, 639)

Experiment 4:

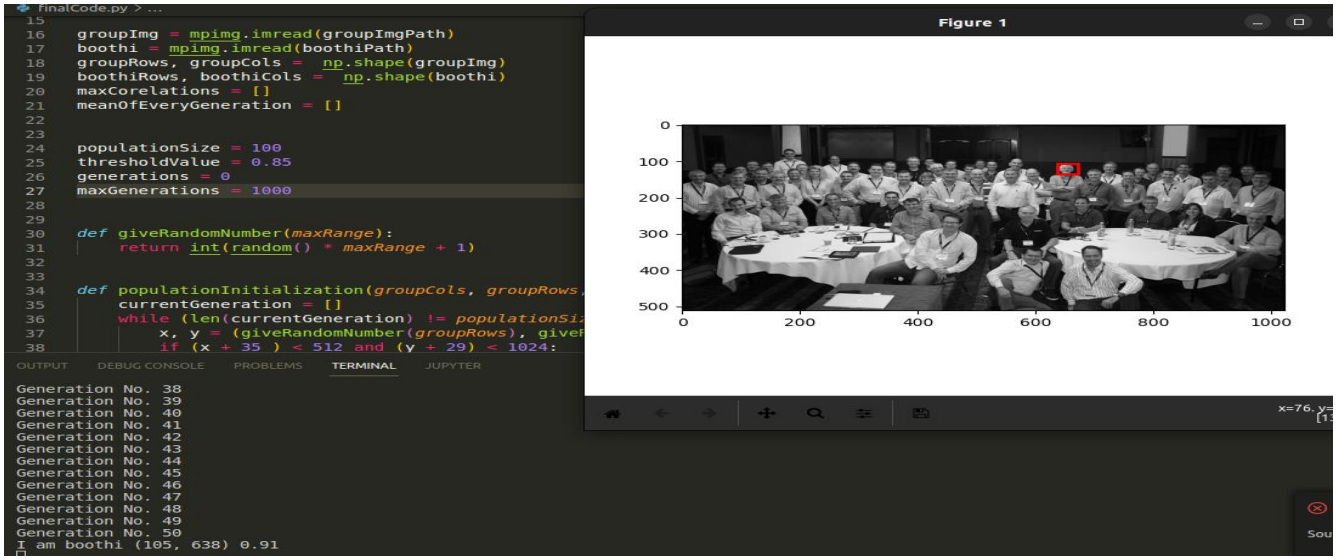
Hypothesis:

As we were able to find boothi, but were not able to achieve desired output of graph in terms of plotting of mean. So, we tried to do two point of crossover excepting the graph of mean of population will behave well.

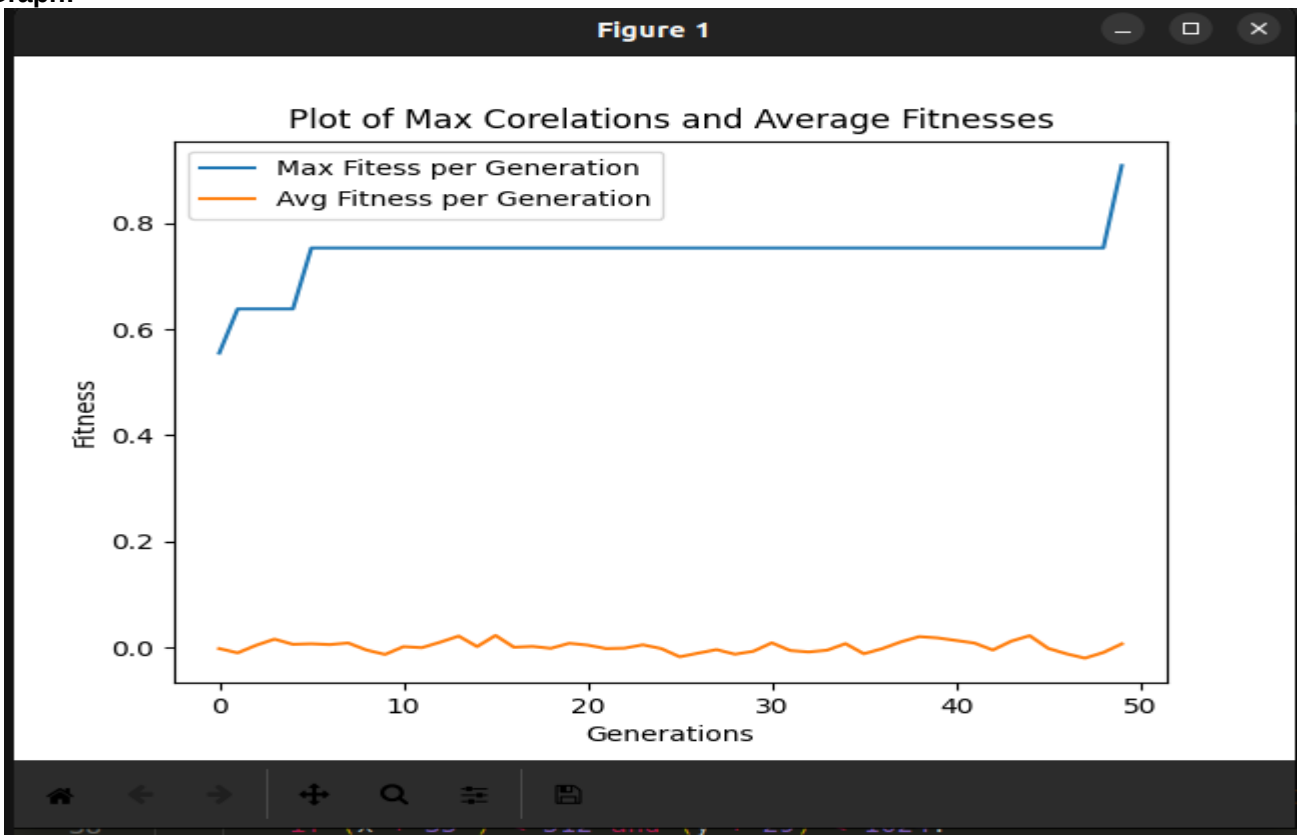
i) Result when

- Initial Population was set to 100.
- Generation was set to 100
- Threshold value to 0.85.

Threshold achieved = 0.91 on 50th generation



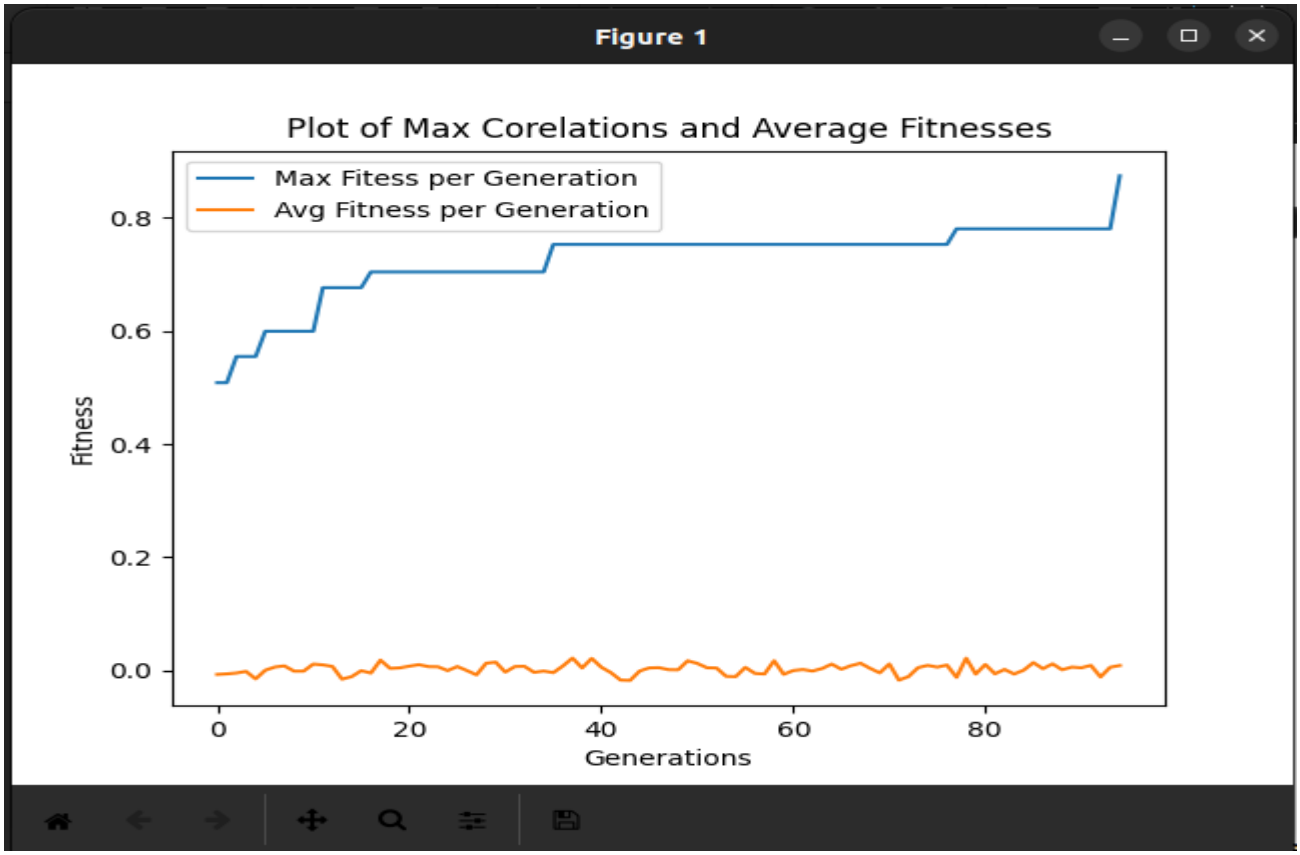
Graph:



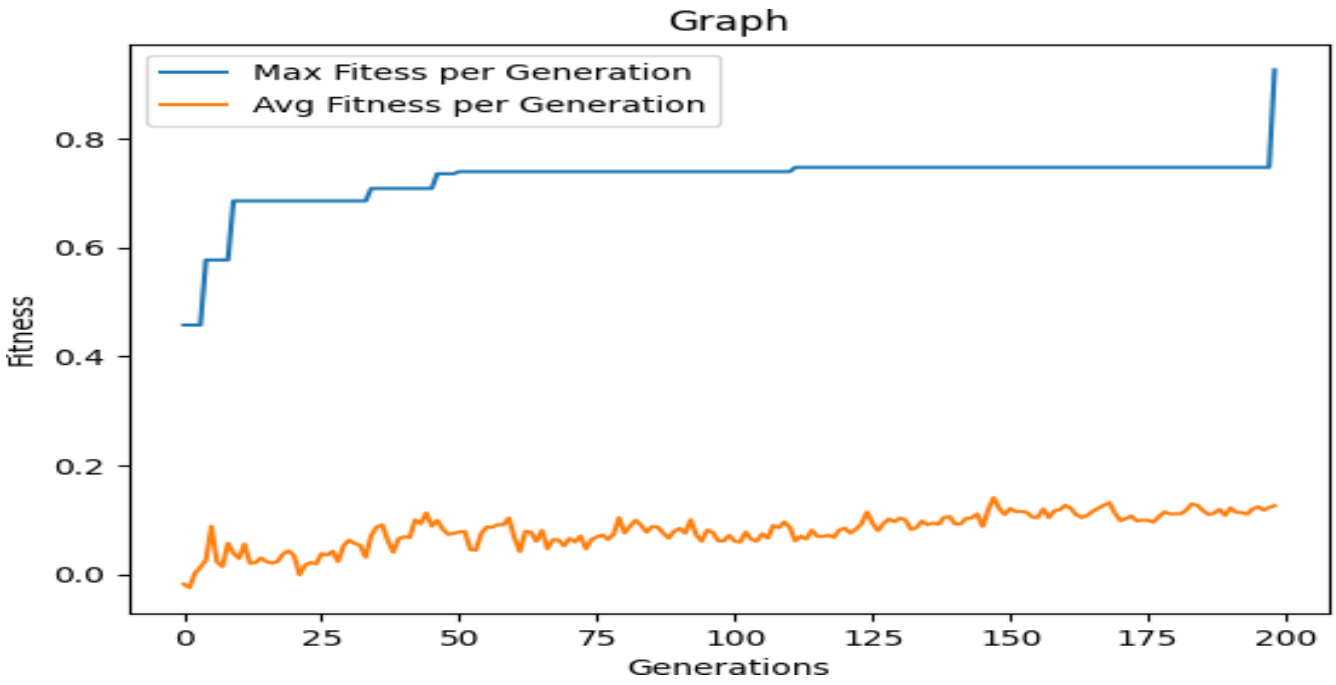
ii) Result when

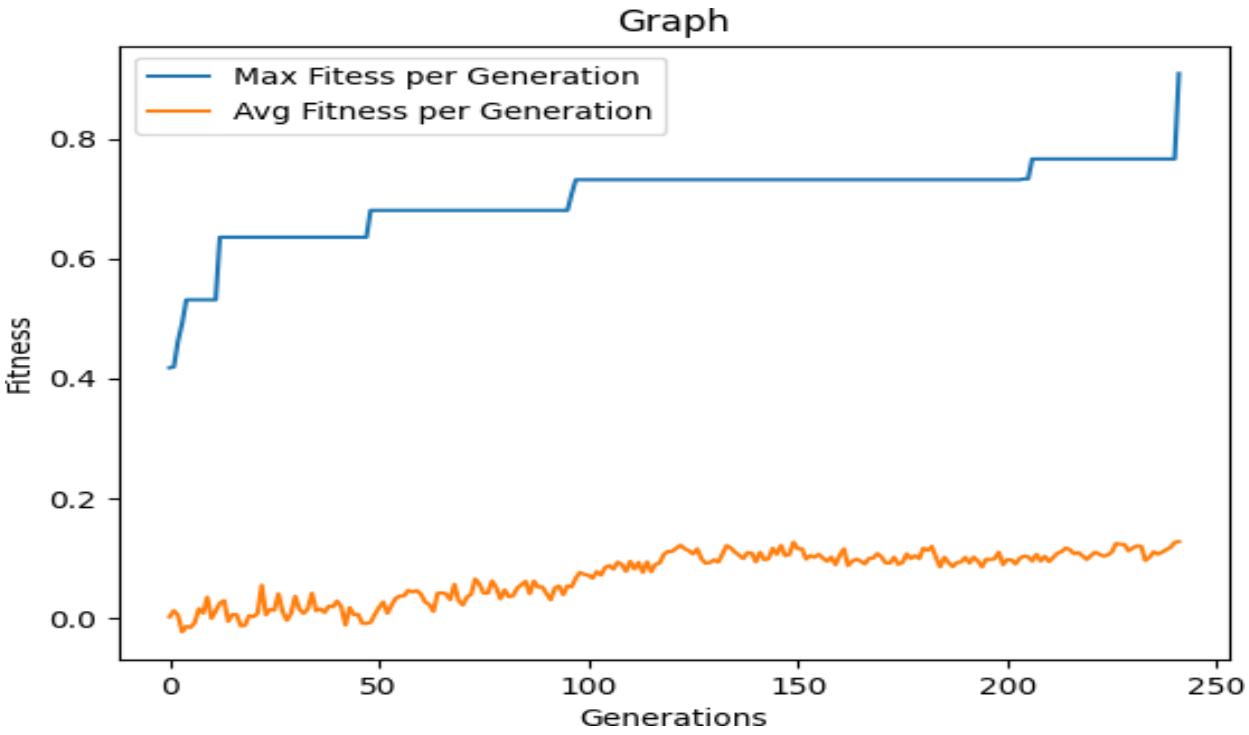
- Initial Population was set to 1000
- Generation was set to 100
- Threshold value to 0.85.

Threshold achieved = 0.87 on 95th generation



Some graph when boothi were found





Probabilistic Analysis with respect to Population Size and Generation Number

The probability of finding the boothi when
populationSize = 100
maxNumberOfGenerations = 100
is close to 50%.

The probability of finding the boothi when
populationSize = 1000
maxNumberOfGenerations = 100
is close to 70%.

The probability of finding the boothi when
populationSize = 1000
maxNumberOfGenerations = 1000
is close to 90%.

Probability is calculated by running program 10 times for each values. Then result is compiled on how many times the boothi is found out of 10 times.

Issues we faced

Main issues we faced some

1. When finding correlations, matrix rows and cols size were not equal sometimes
2. Our population size was being reduced

Why they were happening and how we solved them?

1) Some points were close to max row and column size and when finding corelation we have to add the small image row size and column size. Therefore, those points were going outside of big image. We used while loop and introduced a check before appending a point to initial population.
2) We were using dictionary to store to the fitness value and coordinates with respect to their fitness value (key was fitness value and coordinate were the values). When two or multiple fitness values are same, only one coordinate (x, y) will be stored as value. We used a two dimensional array and were able to solve this issue.