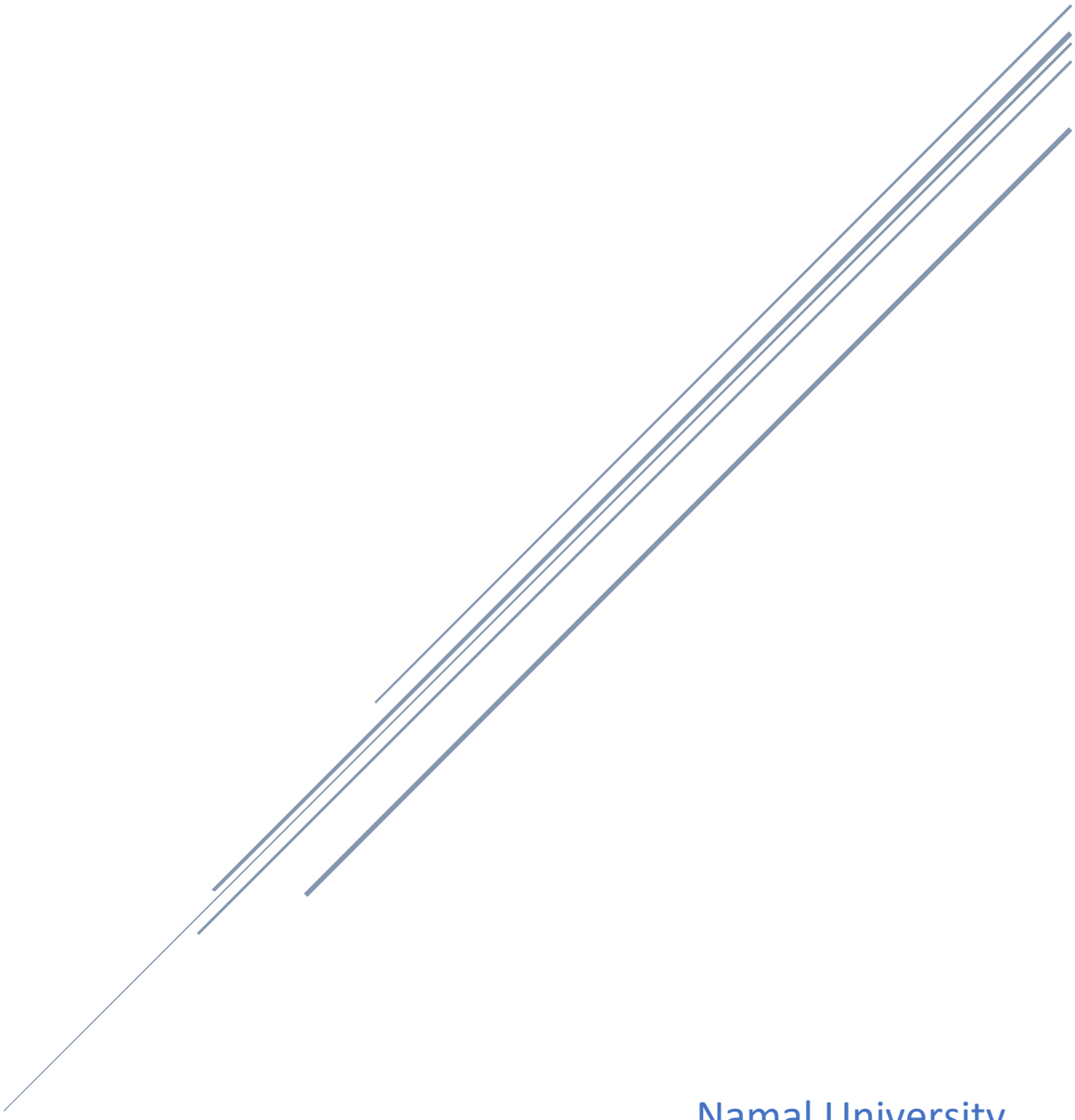
AI- EVOLUTIONARY ALGORITHMS

Template Matching Problem

Instructor: Dr. Junaid Akhter

Submitter: Ali Murtaza

Reg # NIM-BSCS-2019-39



Namal University
Artificial Intelligence

Contents

Background

Charles Darwin was 12 February 1809-19 April 1882 was English Biologist, Naturalist and Geologist. At 16 he studied medicine at Edinburg University. He is best known for his contribution of his theory of evolution.

From 1831 to 1836, he went on a scientific voyage towards the HMS Beagle to study the various facts about science and natural world.

In 1859, he published his work done on theory of evolution in the book “On the Origin of Species” which includes detailed evidence about his great theory.

Now, I will discuss some detail about Darwin Evolutionary Theory.

1.Natural Reality

On his journey to HMS Beagle Darwin observed the nature closely. He observed that there is a relation between different species in the nature. He also observed that there is extensive diversity in nature. He ponders what is the reason behind this diversity.

He observed that different species of animals adopted according to their environment. He noticed that birds of same species have different types of beaks in different regions. He stated that reason of different sizes of beaks is because this requirement of their survival. He observed that most of the species in the nature are closely related. He also noticed that same kind of species change their traits according to their environmental conditions.

2. THEORY

After spending the great time on his observation Darwin deduced a theory which he published in his book “On the Origin of Species”. His theory answers the question of why there is great diversity in nature based on the idea of Natural Selection. In other words, we can also call it as the survival of the fittest.

According to the theory, if we have a population of individuals having certain environment. Then natures select those individuals which suits best to that environment according to its fitness criteria and they reproduce and evolve. In evolution of next generation, the traits which are necessary for survival are inherited to next generations. In this way after many generations and due to transfer of good traits of survival they diversify and change into different species.

3. Model

Since in evolution nature selects the fittest individuals through this way, we get the best and optimized individuals after evolution of generations. This theory is well suited for problems that need optimization. In problems like template matching problem, we are required to reach at optimized point and this theory can help us to get an optimized point. The computational model is as follows.

1. We have population of individual (coordinates). We will select them randomly.
2. Then will select the best individuals inspired from natural selection using fitness function criteria.
3. Then we will cross the best individuals and reproduce the off-springs from them.
4. Then we also do mutations to few individuals by changing their bits.
5. Then we this process is repeated until we have optimized individuals.

Sudo CODE

START

INITIALIZE POP

WHILE BEST FIT GET OR GENERATIONS EXHAUSTED

FITNESS EVALUATION OF POPULATION

RANKING POPULATION BASED ON FITNESS

CROSSOVER THE POPULATION

MUTATE

REPEAT UNTIL LOOP ENDS

STOP

4. Application

We have template matching problem. We have two images one is large and other is small. We have to find small image from large image.

INITIALIZE POPULATION

We first initialize population by choosing random points from large picture. Then, using this population we will get other generations. And this initialization function will run only once in program.

FITNESS EVALUATION

Fitness Evaluation function will get population and evaluate them and assign a fitness value between 0 and 1 based on small picture.

RANKED POPULATION

This function will sort the population based on correlation value and we will get the best fit value of that population on the top. This population will be sent for crossing.

CROSSOVER FUNCTION

Crossover function will get the ranked population and cross the first two coordinates sequence wise and generates a new population. Crossover is done by first converting the coordinates to binary and converging them into one and then flipping the bits by choosing a random number and then converting them into integers. This Crossover function will then send to mutation function.

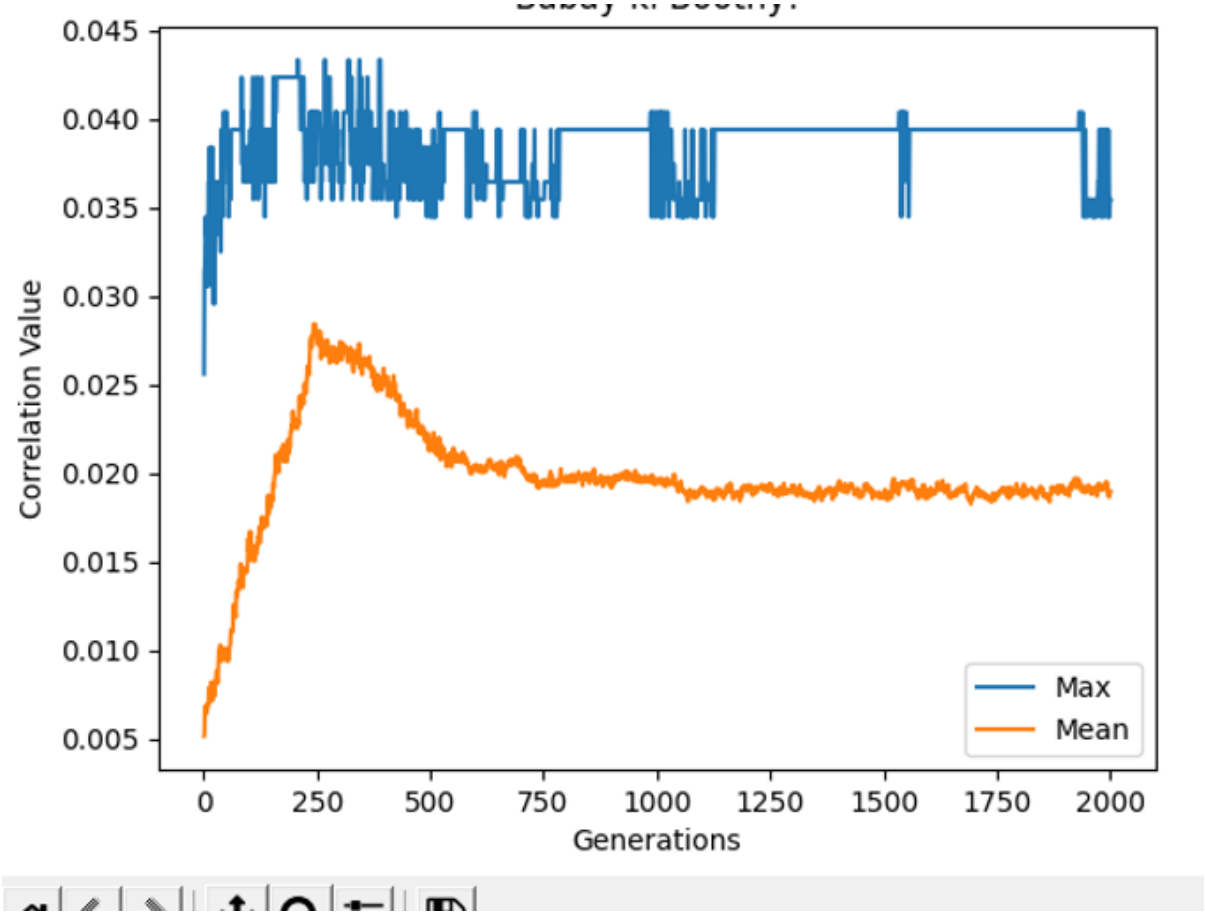
MUTATION

Mutation will be done on all coordinates or few depending upon the experiment. Mutation can done by changing a bit from 0 to 1 or 1 to 0 by choosing a bit randomly. Then, population generated from this will again be sent for fitness evaluation until the best fit coordinate is obtained.

5. EXPERIMENTS

Experiment # 1.

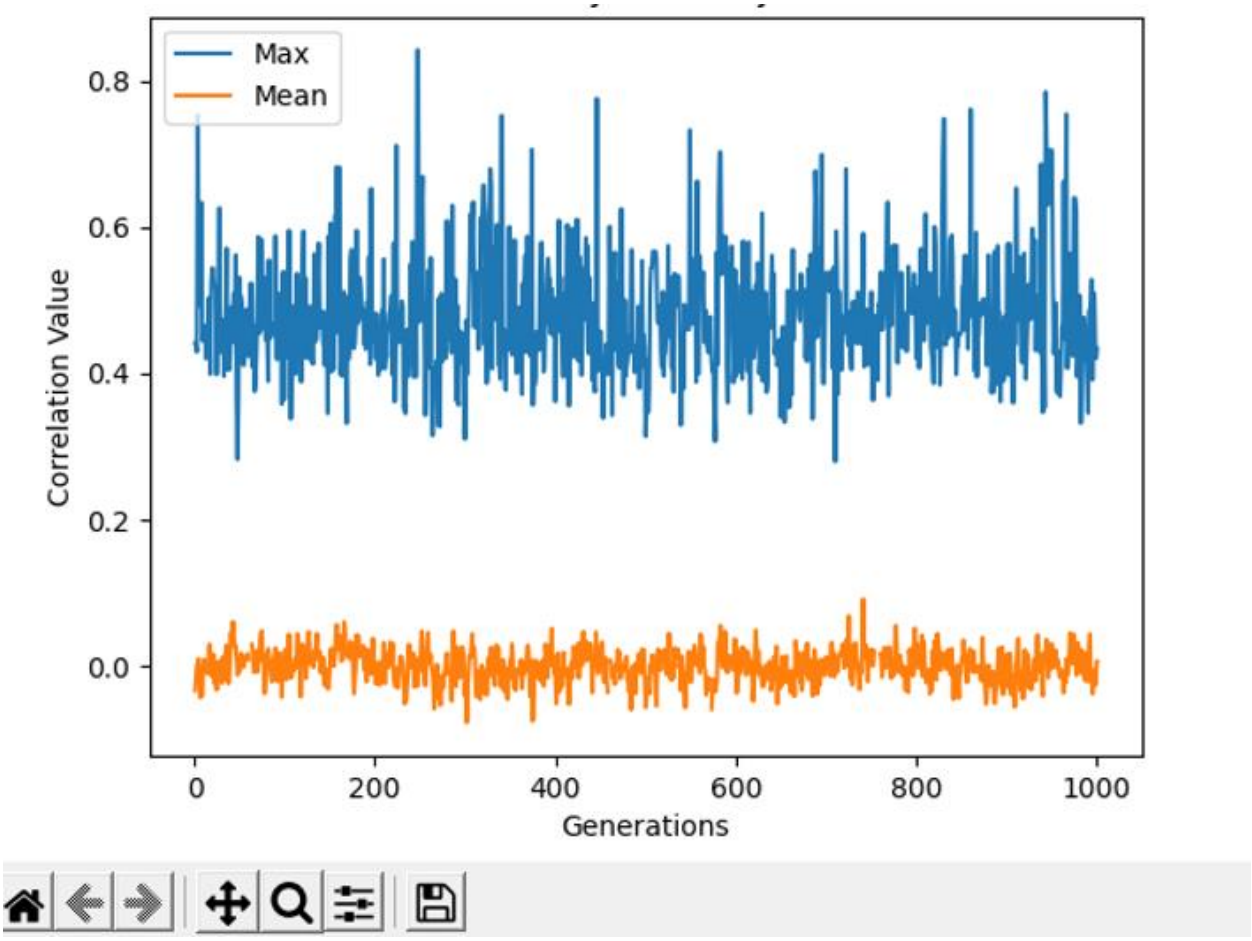
In this experiment I choose the correlation function as a counter with which counts how many points of template matches with the sliced image sliced from main image with only crossover.



Result: The correlation value was not growing from 0.045 even after many generations so strategy needs to be changed.

Experiment #2

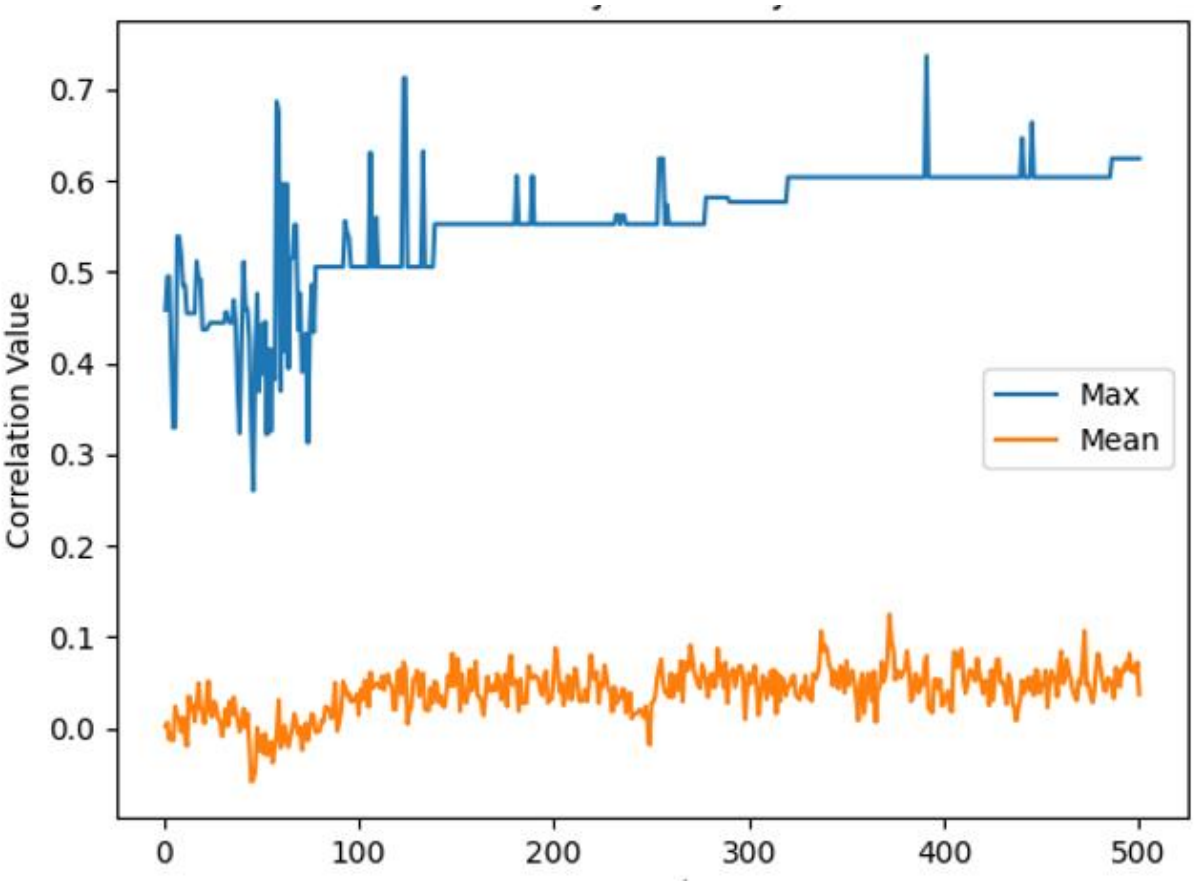
Now Correlation function is changed to matrix correlation function which we studied in Probability and Statistics course after pondering over what is the problem, also crossover and mutation is done on complete population.

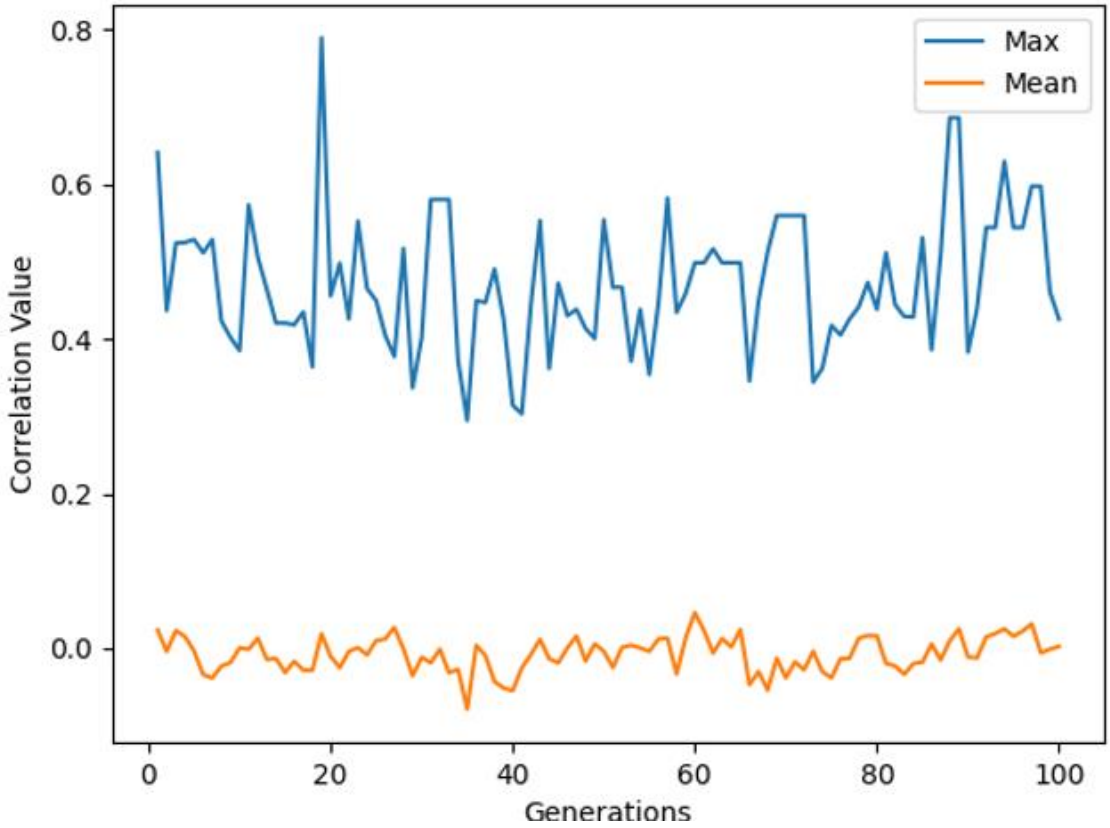


Result: The result was good. Correlation function changing strategy worked but and BABA JEE face was also getting at greater than 0.8 correlation value but still the problem was graph pattern was not according to expectation. So now the strategy again needs to changed.

Some Failed Experiments:

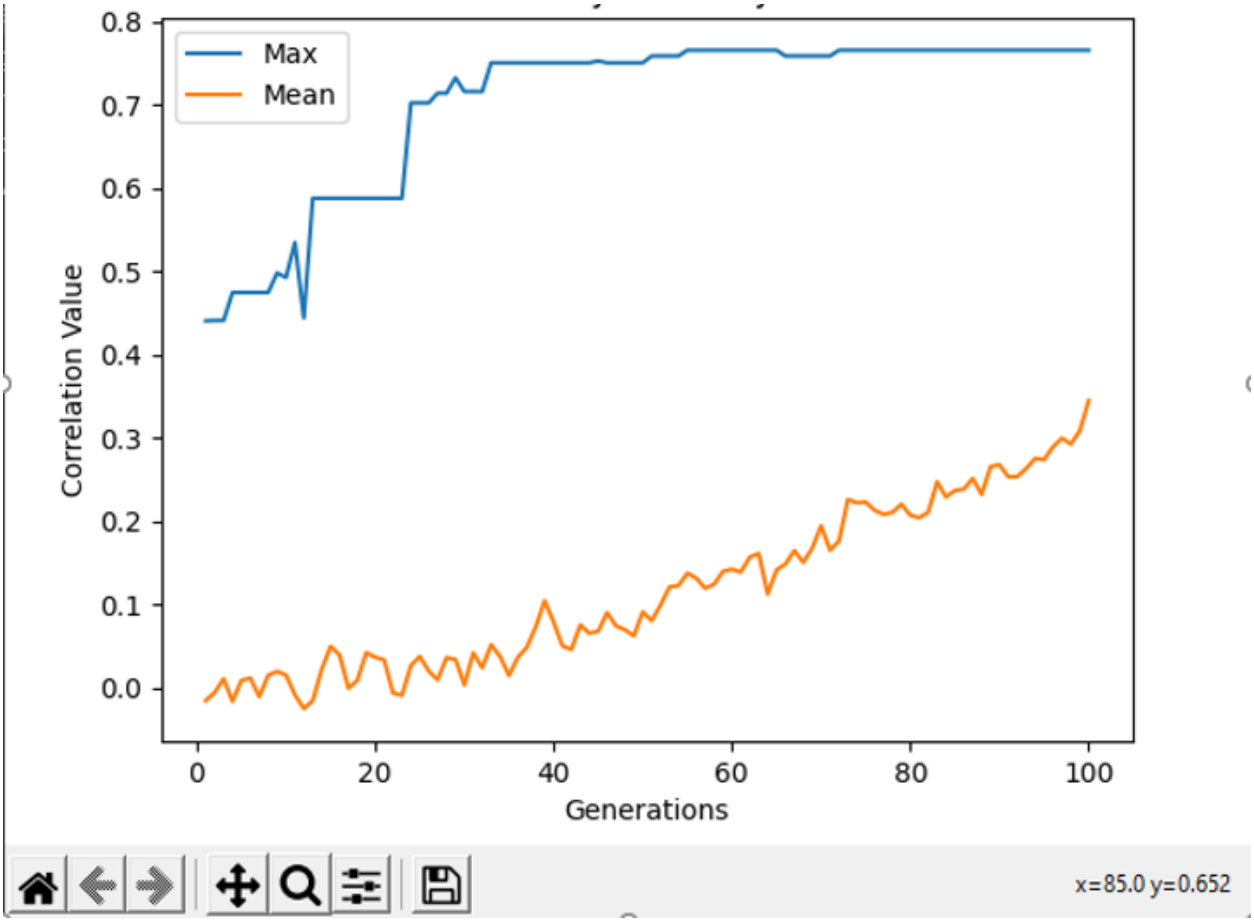
After changing many strategies it is found that the problem is with mutations. Since by changing the patterns of mutation the graph pattern was changing. So, it is found that mutation need to changed to get the right pattern of graph.


Correlation Value
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
100
200
300
400
500
Max
Mean


Correlation Value
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100
Generations
Max
Mean

Experiment # 3:

I tried to swap the worse of population with the best of population and slowing the rate of mutation by mutating randomly only one point.



Result: After this experiment the result was some how satisfying because BABA JEE FACE was getting and generations also evolved. But this is also not the end. This can also be improved so get the ideal solution.

```python
import random

import matplotlib.pyplot as plt
import matplotlib.image as img
from matplotlib.patches import Rectangle
from PIL import Image
import numpy as np
import math


bigimg = img.imread("groupGray.jpg")
template = img.imread("boothiGray.jpg")


threshold = 0.85




# print(c)

b_shape = bigimg.shape

rows = b_shape[0]
```

```
b_shape = bigimg.shape

rows = b_shape[0]
r = rows -1
cols = b_shape[1]
c= cols -1


size = bigimg.size


sizeofpop = 100
gen_max = []
gen_mean = []
gens = []
max_coords = []
max_cor = 0

def initialize_pop(rows,columns,sizeOfpop):
    population = []
    for i in range(sizeOfpop):
        row = random.randint(0,r)

        col = random.randint(0,c)
        point = (row,col)
        population.append(point)

    #
    return population

population = initialize_pop(rows,cols,sizeofpop)
```

```
55
56  def fitness_eval(bigimg,template,population):
57      correlation_list = []
58      for i in range(len(population)):
59          point = population[i]
60          if point[0]+35 < (bigimg.shape)[0] and point[1]+29 < (bigimg.shape)[1]:
61              prow,pcol = point[0],point[1]
62              sliced_img = bigimg[point[0]:point[0]+35,point[1]:point[1]+29]
63              fitness = correlation(template,sliced_img)
64              corr = (fitness,prow,pcol)
65              correlation_list.append(corr)
66          else:
67              prow,pcol = point[0],point[1]
68              corr = (0,prow,pcol)
69              correlation_list.append(corr)
70
71      return correlation_list
72
73
74
75
76  def correlation(template,sliced_img):
77
78      temp_mean = np.mean(template)
79
80      slice_mean = np.mean(sliced_img)
81
82      num = 0
83      d1 = 0
84      d2 = 0
85      slice_col = 0
86      for i in range(len(sliced_img)):
```

```
87          temp =template[i]
88          sliced = sliced_img[i]
89
90
91          for j in range(len(temp)):
92              if j < len(sliced):
93                  # if sliced[j] != []:
94                  slice_value = sliced[j]
95                  # print(temp[i])
96                  num +=(temp[j] - temp_mean)*(slice_value - slice_mean)
97                  d1 += (temp[j] - temp_mean)** 2
98                  d2 += (slice_value - slice_mean)**2
99                  a = d1 * d2
100
101     denum = math.sqrt(a)
102     corr = num / denum
103     return corr
104
105
106 def rankedPop(pop_correlation):
107     pop_correlation.sort(key = lambda x: x[0],reverse = True)
108
109
110     corr_array = []
111     points_array = []
112     # print(len(pop_correlation))
113     for i in range(len(pop_correlation)):
114         point = pop_correlation[i]
115         corr_array.append(point[0])
116         coordinate = (point[1],point[2])
117         points_array.append(coordinate)
118     b= corr_array[0]
```

```
        max_cor = b


        gen_max.append(b)
        mean = sum(corr_array) /len(corr_array)

        max_point = points_array[0]
        max_coords.append(max_point)


        points_array[(len(points_array)-1)] = points_array[0]



        gen_mean.append(mean)
        return points_array, max_cor



def crossover(sorted_pop):
        # print(sorted_pop)
        bimg_row_size = len(bin(b_shape[0]-1).replace("0b",""))

        bimg_col_size = len(bin(b_shape[1]-1).replace("0b",""))


        new_pop = []
        n1_ltemp = 0
        n1_rtemp = 0
        n2_ltemp = 0
        n2_rtemp = 0
```

```
    n2_rtemp = 0
    for i in range(len(sorted_pop)):
        if i % 2 == 0:
            i_point = sorted_pop[i]
            n1_row = bin(i_point[0]).replace("0b","")
            n1_col = bin(i_point[1]).replace("0b","")

            n1_row = n1_row.zfill(bimg_row_size)


            n1_col = n1_col.zfill(bimg_col_size)

            n1  = n1_row + n1_col



            i_point2 = sorted_pop[i+1]
            n2_row = bin(i_point2[0]).replace("0b","")
            n2_col = bin(i_point2[1]).replace("0b","")

            n2_row = n2_row.zfill(bimg_row_size)

            n2_col = n2_col.zfill(bimg_col_size)


            n2 = n2_row + n2_col



            slicer = random.randint(0,len(n1))
            # slicer = 5
```

```
            n1_ltemp = n1[:slicer]
            n1_rtemp = n1[slicer:]

            n2_ltemp = n2[:slicer]
            n2_rtemp = n2[slicer:]


            c1_bin = n1_ltemp +n2_rtemp
            c2_bin = n2_ltemp + n1_rtemp
            # print(n1, c1_bin,"\n", n2,c2_bin )




            c1_row = c1_bin[:(len(c1_bin)) // 2]
            c1_col = c1_bin[(len(c1_bin) // 2):]

            c2_row = c2_bin[:(len(c2_bin)) // 2]
            c2_col = c2_bin[(len(c2_bin) // 2):]

            c1 = (int(c1_row,2), int(c1_col, 2))
            c2 = (int(c2_row,2), int(c2_col, 2))
            new_pop.extend([c1,c2])

    return new_pop

#
def mutation(crossed_pop,max_cor):
    # print(len(crossed_pop))
    mutated_pop = []
    rand_indexes = []
```

```
219     for i in (rand_indexes):
220
221
222         cross_point = crossed_pop[i]
223         row = cross_point[0]
224         col = cross_point[1]
225         # print(row,col)
226         row_bin = bin(row).replace("0b","")
227         col_bin = bin(col).replace("0b","")
228
229
230         mutated_row_bin = ''
231         mutated_col_bin = ''
232         mutated_col_bin = col_bin
233
234
235
236         for i in range(len(row_bin)):
237             if i != 0:
238                 # print("bbbbbbbbb")
239                 mutated_row_bin += row_bin[i]
240             else:
241                 if row_bin[i] == '0':
242                     # print('aaaaaaaaaaaaaaaaa')
243                     mutated_row_bin += str(1)
244                 else:
245                     mutated_row_bin += str(0)
246
247
248
249
250         mutated_point = mutated_row_bin + mutated_col_bin
```

```
            mutated_point = mutated_row_bin + mutated_col_bin
            mutated_row = mutated_point[:len(mutated_point) // 2]
            mutated_col = mutated_point[(len(mutated_point)) //2 :]

            mutated_p = (int(mutated_row,2),int(mutated_col,2))
            crossed_pop[i]=mutated_p

            # print("sadadsadasd")
        return crossed_pop
    # population = mutation(crossed_pop)


    # print(gen_mean)

    max_counter = 0
    g = 0
    gen = 0
    pre_max = 0
    while gen < 100 and threshold > max_cor and max_counter != 600  :
        print(max_cor)
        g +=1
        pop_correlation = fitness_eval(bigimg,template,population)
        # print(pop_correlation)
        sorted_pop,max_cor = rankedPop(pop_correlation)
        if pre_max == max_cor:
            max_counter +=1
        else:
            max_counter =0
        crossed_pop= crossover(sorted_pop)
        # population= crossover(sorted_pop)
        # print(crossed_pop)
        population = mutation(crossed_pop,max_cor)
```

```
283         gens.append(g)
284         # print("ccccccccccc")
285         gen +=1
286         pre_max = max_cor
287         print(gen,max_counter)
288     # np.corrcoef(template)
289
290     maxi =max(gen_max)
291     mp= max_coords[len(max_coords)-1]
292
293
294
295
296
297     im = Image.open('groupGray.jpg')
298
299
300     plt.imshow(im)
301
302     ax = plt.gca()
303
304
305     rect = Rectangle((mp[1],mp[0]), 29, 35, linewidth=1, edgecolor='r', facecolor='none')
306
307     ax.add_patch(rect)
308
309     plt.show()
310
311
312
```