

***AI LAB ASSIGNMENT:***

***GENETIC ALGORITH:***

***SUBMITTED TO: SIR QAZI WASQA KHAN***

***SUBMITTED BY: RABIA ZEB***

***REGISTRATION NO: SP20-BCS-037***

***DATED:27TH MAY,2022***

PART 1:

import random

import copy

POPULATION\_SIZE = 10

GENERATION =10

LEN\_CHROMOSOME = 12

split1 = 5

split2=8

**Create Chrom**

def create\_chrom():

chrom = []

for i in range(LEN\_CHROMOSOME):

if random.random()>0.5:

chrom.append(1)

else:

chrom.append(0)

return chrom

**Create Population:**

def create\_pop():

pop = []

for i in range(POPULATION\_SIZE):

a = create\_chrom()

pop.append(a)

return pop

print(create\_pop())

**Create fitness:**

def my(x):

return x[1]

def fitness(chrom):

return chrom.count(0)

def pop\_with\_fit(population):

pop\_fit =[]

for i in population:

a=(i,fitness(i))

pop\_fit.append(a)

return sorted(pop\_fit, key=my, reverse=True )

print(pop\_with\_fit(create\_pop()))

**print sum:**

for i in create\_pop():

print(sum(i))

**Create selection:**

def selection(pop):

print(len(pop[:5]))

return pop[:8]

print(selection(pop\_with\_fit(create\_pop())))

**Create Crossover:**

def crossover(sel):

cros\_lis=[]

for i in range(0,len(sel),2):

p1=sel[i][0]

p2=sel[i+1][0]

c1=p1[:split1]+p2[split1:split2]+p1[split2:]

c2=p2[:split2]+p1[split2:split1]+p2[split1:]

cros\_lis.append(p1)

cros\_lis.append(p2)

cros\_lis.append(c1)

cros\_lis.append(c2)

return cros\_lis

print(crossover(selection(pop\_with\_fit(create\_pop()))))

**Create Mutation:**

def mutation(crs):

mut=[]

for a in crs:

ind1=random.randint(0,len(a)-1)

ind2=random.randint(0,len(a)-1)

if a[ind1]<a[ind2]:

a[ind1]=0

else:

a[ind2]==1

mut.append(a)

return mut

print(mutation(crossover(selection(pop\_with\_fit(create\_pop())))))

**Show Result:**

def show(pop,generation):

print()

print('Generation no [',generation,']','Best Chromosome: ',pop[0][0],'Fitness:',pop[0][1])

print(60\*'-')

for no, i in enumerate(pop):

print('chromosome#',no+1,'<<<',i[0],'>>>','Fitness',i[1])

print(show(create\_pop(),mutation(crossover(selection(pop\_with\_fit(create\_pop()))))))

**PART B:**

import numpy as np

import random

def countThreat (population):

threat = 0

for currentGene in range(0,len(population)):

currentQueen = [currentGene, population[currentGene]]

for previousGene in range(0,currentGene):

previousQueen = [previousGene, population[previousGene]]

slope = (currentQueen[1]-previousQueen[1]/(curreneQueen[0]-previousQueen[0]))

if slope == 0:

threat +=1

break

elif slope == 1 or slope == -1:

threat +=1

break

return threat

def initilizeRandomPopulation(populationSize, ChromosomeSize):

population = []

for i in range(populationSize):

population.append(np.random.random\_integers(low=0,high=ChromosomeSize-1, size=(ChromosomeSize)))

return population

def sortPopulation(population):

population.sort(key=countThreat)

return population

def crossover(population,crossoverCount):

chromosomeLength = len(poplation[0])

for i in range(0, crossoverCount):

crossoverParent1 = random.choice(population)

crossoverParent2 = random.choice(population)

crossoverPoint = random.randint(1, chromosomeLength-1)

child1 = []

chil1.extend(crossoverParent1[: crossoverPoint])

chil1.extend(crossoverParent2[: crossoverPoint])

child2 = []

chil2.extend(crossoverParent2[: crossoverPoint])

chil2.extend(crossoverParent1[: crossoverPoint])

population.append(child1)

population.append(child2)

return population

def mutation(population, mutationCount):

chormosomeLength = len(population[0])

for i in range(0, mutationCount):

mutationParent = random.randint(0, chromosomeLength-1)

mutationGene = random.randint(0, chromosomeLength-1)

child = mutationParent

child[mutationPoint] = mutationGene

population.append(child)

return population

print(mutation(crossover((population,crossoverCount()))))