CSE422 Lab02

Genetic Algorithm

Name: Subhi Bhuiyan

ID: 17201116

import numpy as np

import random

#calculate non-attacking pairs

def fitness\_func(population, queen):

max\_fitness = int((queen\*(queen-1))/2)

clashes = 0

#calc row and column clashes

row\_col\_clashes = abs(len(population) - len(np.unique(population)))

clashes += row\_col\_clashes

#calc diagonal clashes

for i in range(len(population)):

for j in range(len(population)):

if(i != j):

dx = abs(i-j)

dy = abs(population[i]-population[j])

if(dx == dy):

clashes += 1

return int(max\_fitness - clashes)

#random selection function

def select(population):

fit = fitness\_func(population,queen)

a = [0,1,2,3,4]

size = 1

p = [.31, .29, 0.26, 0.14]

replace=True

selectA = np.random.choice(fit,size,replace,p)

return selectA

#crossover function

def crossover(parent\_x, parent\_y):

index = random.randint(0,4)

child = np.append(parent\_x[0:index],parent\_y[index:8])

return child

#mutation function

def mutation(child):

index = random.randint(0,len(child)-1)

val = random.randint(0,len(child)-1)

child[index] = val

return child

def genetic\_algo(population, queen, mutation\_threshold = 0.3):

nmax = 100000

goal\_fit = int((queen\*(queen-1))/2)

print("\ngoal fit for current population",goal\_fit)

print("\nproblem dimension: ", queen , "x" , queen)

print("population size: ",len(population))

print("max generations: ",nmax)

while nmax > 0:

new\_population = []

for i in range(len(population)):

parent\_x = select(population)

parent\_y = select(population)

child = crossover(parent\_x,parent\_y)

if random.uniform(0,1) < mutation\_threshold:

child = mutation(child)

if fitness(child, goal\_fit) == goal\_fit:

print("result: ",child," found in ", nmax-queen," generations.\n")

return child

new\_population.append(child)

population = new\_population

nmax -= 1

print("\nNo solution found in ",nmax," generations. Try again.\n")

return None

#number of queens

queen = 8

#number of start\_population / value of k

start\_population = 10

mutation\_threshold = 0.3

population = np.random.randint(0, queen, (start\_population, queen))

#print(population)

genetic\_algo(population, queen, mutation\_threshold)