Nama : Afiyatar Asyer
NIM : 672019061
Matkul : TEKNIK OPTIMALISASI C

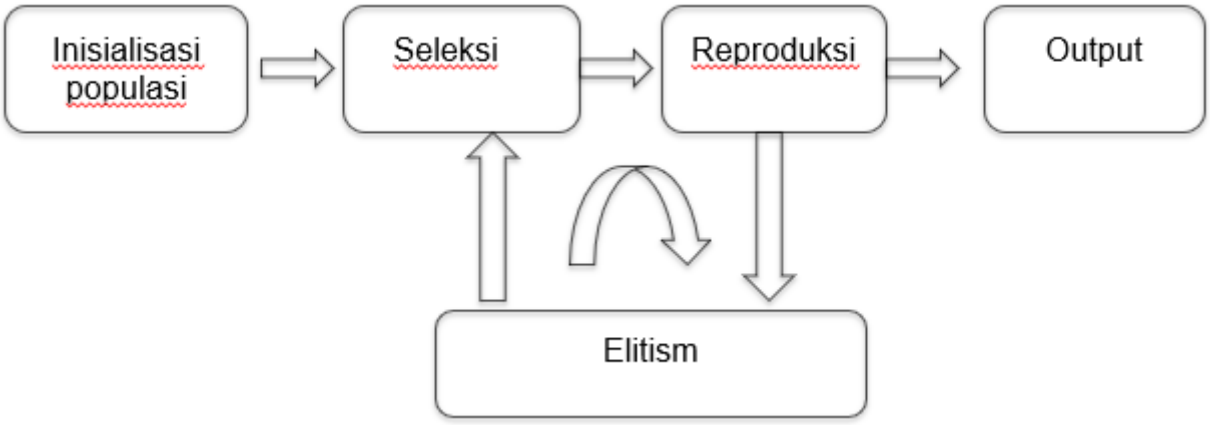
TUGAS 1
GENETIC ALGORITHM - TRAVELING SALESMAN PROBLEM

Diketahui:

- Ada 5 Kota yang akan dikunjungi **: A B C D E**
- Kordinat x dan y **: A (0,0), B (5,6), C (1,4), D (3,4), E (7,7)**

Ditanya **: Tujuan Terpendek untuk mengunjungi 5 Kota?**

Penerapan AG dalam menyelesaikan permasalahan TSM:



Codingan

Library yang digunakan:

```
#Library yang digunakan
import numpy as np #Menyimpan nilai dalam bentuk array
import random #Digunakan untuk mencari nilai random
from array import *
```

Penentuan daftar Kota:

```
daftarKota = [[0,0], [5,6], [1, 4], [3,4],[7,7]] #Kordinat dari kota
for i in daftarKota:
    for j in i:
        print(j,end = " ")
    print()

jumlahKota =  len(daftarKota)
print("Jumlah Kota: ", jumlahKota)
```

```
0 0
5 6
1 4
3 4
7 7
Jumlah Kota:  5
```

Daftar kota yang digunakan dimasukan dalam bentuk array 2 Dimensi x dan y

Sehingga menghasilkan nilai:

```
0 0
5 6
1 4
3 4
7 7
```

Inisialisasi Populasi:

```
def initPopulasi(ukuranPopulasi,jumlahKota): #Fungsi initialisasi Populasi dengan parameter ukuranPopulasi dan jumlahKota
  Pop = np.empty((ukuranPopulasi,jumlahKota)) # Variabel yang menyimpan dari nilai inisialisasi populasi dalam bentuk matriks Baris dan Kolom
  Pop = Pop.astype('int') # Konversi dalam bentuk integer
  for i in range(ukuranPopulasi):
    Indiv = np.random.permutation(jumlahKota) # Membangkitkan nilai acak (Random) melalu perulangan dengan range jumlah kota
    for j in range(jumlahKota):
      Pop[i][j]=Indiv[j] # Inisialisasi nilai tiap kolom
  return Pop #Nilai kembalian dari Pop


ukuranPopulasi = 4 # Ukuran dari Populasi
Panggil = initPopulasi (ukuranPopulasi,jumlahKota) # Panggil Fungsi initPopulasi
print(Panggil) # cetak nilai Panggil
```

```
[[2 3 1 4 0]
 [0 3 2 1 4]
 [3 4 0 2 1]
 [4 3 1 0 2]]
```

Populasi dari pop akan terdiri dari empat populasi dimana jumlah kolom akan menyesuaikan dengan jumlah kota (bersifat acak dapat berubah setiap di jalankan):

```
[2 3 1 4 0] Solusi 1
[0 3 2 1 4] Solusi 2
[3 4 0 2 1] Solusi 3
[4 3 1 0 2] Solusi 4
```

Fungsi Objektif untuk menentukan nilai euclidian:

```
#Fungsi Objektif dalam mengukur nilai euclidian
def funcObj(populasi,daftarKota): #Fungsi untuk menghitung nilai objektif dengan parameter populasi dan daftarKota

  ukuran = populasi.shape # variabel ukuran dalam menentukan populasi dalam bentuk matriks
  ukuranPopulasi = ukuran[0] #variabel kolom
  jumlahKota = ukuran[1] #variabel baris

  matrikJarak = np.empty((ukuranPopulasi)) # variabel yang akan menghasilkan ukuran array
  matrikJarak  = matrikJarak.astype('float')
  n = len(daftarKota) # variabel dengan nilai banyak kota

  for i in range(ukuranPopulasi):
    jarak = 0
    for j in range(n-1): #perulangan dengan batas daftar kota dikurang satu hingga menyesuaikan ukuran populasi
      jarakX = daftarKota[populasi[i][j]][0]-daftarKota[populasi[i][j+1]][0] #membandingkan jarak kota satu dengan kota lain dengan kordinat x
      jarakY = daftarKota[populasi[i][j]][1]-daftarKota[populasi[i][j+1]][1] #membandingkan jarak kota satu dengan kota lain dengan kordinat y
      # i : Rute, j : Kota
      Jarak = np.sqrt(jarakX**2 + jarakY**2 )  # menghitung jarak euclidian
      jarak = jarak  + Jarak # nilai jarak

    jarakTotal =  np.sqrt((daftarKota[populasi[i][n-1]][0]-daftarKota[populasi[i][0]][0])**2+
                  (daftarKota[populasi[i][n-1]][1]-daftarKota[populasi[i][0]][1])**2 )
    jarak = jarak  + jarakTotal # menghasilkan jarak rute
    matrikJarak[i] = jarak # Nikau jarak (nilai rute) akan disimpan di matriks jarak

  return matrikJarak # mengembalikan nilai jarak

fitP = funcObj(Panggil,daftarKota) # variabel untuk fungsi objek
print(fitP) # panggil fungsi objek
```

```
[29.24637567 26.46998636 26.32316364 20.83130956]
```

Menghitung kualitas solusi (fitness)

A (0,0), B (5,6), C (1,4), D (3,4), E (7,7)

I_1 = A B D C E

$$f(I_1) = d_{AB} + d_{BD} + d_{DC} + d_{CE} + d_{EA}$$

$$d_{AB} = \sqrt{(0-5)^2 + (0-6)^2} = \sqrt{25+36} = \sqrt{61} = 7.81$$

$d_{BD} = 2.83$

$d_{DC} = 2.00$

$d_{CE} = 6.71$

$d_{EA} = 9.90$

$f(I_1) = 29.25$

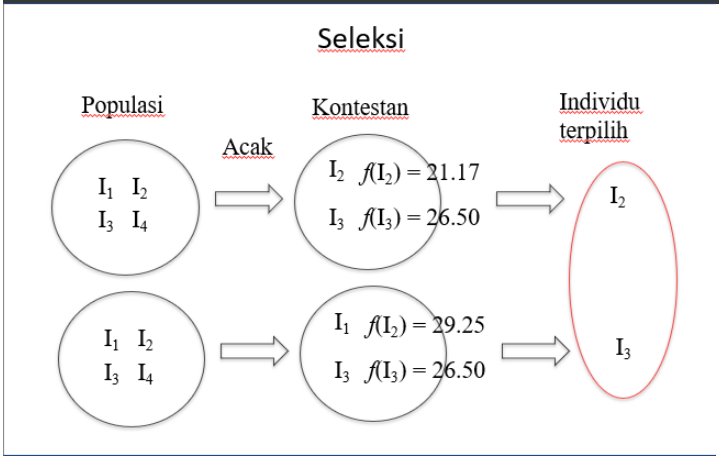
Sehingga didapat kualitas solusi:

```
[29.24637567 26.46998636 26.32316364
20.83130956]

Dari nilai rute
```

Fungsi seleksi untuk menentukan induk terbaik:

```
# Fungsi seleksi menggunakan turnament
def funcSelection (population,funcObj):
  jmlInduk = 2
  ukuran = population.shape
  ukuranPopulasi = ukuran[0]#variabel kolom
  jumlahKota = ukuran[1]#variabel baris

  Induk = np.empty((jmlInduk,jumlahKota)) # inisialisasi array induk terpilih (rute)
  Induk = Induk.astype('int') # konvert ke integer

  for i in range (2): # perulangan sebanyak induk
    # memilih dua individu secara acak
    r1 = random.randint(0,ukuranPopulasi-1) # individu 1 yang dipilih secara acak
    r2 = random.randint(0,ukuranPopulasi-1) # individu 2 yang dipilih secara acak

  if  funcObj[r1] < funcObj [r2]: # kondisi jika individu satu lebih besar dari individu dua maka individu satu yang terpilih
                                  # kondisi jika individu dua lebih besar dari individu satu maka individu dua yang terpilih
    for j in range(jumlahKota):
      Induk [i][j] = population[r1][j] # Individu pertama yang dipilih sebagai induk
  else:
    for j in range(jumlahKota):
      Induk [i][j] = population[r2][j] # Individu pertama yang dipilih sebagai induk
  return Induk

Induk = funcSelection(Panggil,fitP)
print(Induk)
```

```
[[4 2 0 1 3]
 [3 4 0 2 1]]
```



Sehingga menghasilkan individu terpilih dengan jalur kota

[4 2 0 1 3]
[3 4 0 2 1]

Fungsi Crossover:

```
#crossover

def funcPMX(Induk):
  ukuran = Induk.shape
  baris = ukuran[0]#variabel baris
  kolom = ukuran[1]#variabel kolom

  Anak = np.empty((2,kolom)) #variabel anak dalam bentuk array dengan parameter dua dan kolom
  Anak = Anak.astype('int') # konversi ke integer

  r1 = random.randint(1,kolom-2) # variabel yang akan dibangkitkan nilai random

  for i0 in range(r1):

    Anak [0][i0] = Induk[0][i0] # Anak baris permition akan mengambil induk baris kolom prmition
    Anak [1][i0] = Induk[1][i0] # Anak kolom permition akan mengambil induk baris kolom prmition

  batasCek1 = r1 #variabel batas cek agar tidak ada individu yang sama
  batasCek2 = r1
```

```
  #segment 2 Permutasi
  for i in range(kolom):
    m = (i+r1)%kolom

    n1 = Induk[0][i]
    n2 = Induk[1][i]

    idxAda1 = 0
    idxAda2 = 0

    for j1 in range(batasCek1): #perulangan batas cek 1
      if batasCek1>(kolom-1):# kondisi jika individu sudah ada
        idxAda1 = 1
      if Anak [0][j1] ==n2: #kondisi jika anak sudah ada di n2
        idxAda1 = 1
    for j2 in range(batasCek2):
      if batasCek2>(kolom-1):
        idxAda2 = 1
      if Anak [1][i0]  == n1:
        idxAda2 = 1
```

```
    if idxAda1 == 0:
      Anak[0][batasCek1] = n2
      batasCek1 = batasCek1 +1
    if idxAda2 == 0:
      Anak[1][batasCek2] = n1
      batasCek2 = batasCek2 +1

  return Anak # mengembalikan nilai anak

AnakCX = funcPMX(Induk) #variabel nilai anak crossover
print("Anak :",AnakCX) #panggil nilai AnakCX
```

```
Anak : [[3 4 0 2 1]
 [0 2 1 3 4]]
```

Fungsi Mutasi Swap:

```
def MutasiSwap(Induk):
  ukuran = Induk.shape
  baris = ukuran[0] #Baris
  kolom = ukuran[1] #Kolom

  Anak = np.empty((2,kolom)) #Inisialisasi variable anak
  Anak = Anak.astype('int') #Konversi ke integer

  Anak = Induk

  r1 = random.randint(0,kolom-1) #Memilih bilangan random untuk kota yang akan disilangkan
  r2 = random.randint(0,kolom-1) #Memilih bilangan random untuk kota yang akan disilangkan

  while r1 == r2: #Jika a kota satu dan kota dua sama maka kota dua akan dibangkitkan kembali
    r2 = random.randint(0,kolom-1)
```

```
      #Mutasi yang dilakukan
      I1rs1 = Induk [0][r1]
      I1rs2 = Induk [0][r2]
      I2rs1 = Induk [1][r1]
      I2rs2 = Induk [1][r2]
      Anak [0][r1] = I1rs2
      Anak [0][r2] = I1rs1
      Anak [1][r1] = I2rs2
      Anak [1][r2] = I2rs1

      return Anak #mengembalikan nilai anak

    print("AnakCX :",AnakCX) #cetak nilai anak crossover
    AnakM = MutasiSwap(AnakCX)
    print("AnakM :",AnakM)#cetak nilai anak mutasi
```

```
AnakCX : [[3 4 1 2 0]
 [3 4 3 1 2]]
AnakM : [[3 2 1 4 0]
 [3 1 3 4 2]]
```

Fungsi Eliminasi:

```
def Elitism(P,Anak,fitP,fitAnak):
  ukuranP = P.shape#variabel ukuranp dalam bentuk array
  barisP = ukuranP[0] #baris
  kolomP = ukuranP[1] #kolom
  ukuranA = Anak.shape #variabel ukurana dalam bentuk array
  barisA = ukuranA[0]#baris
  kolomA = ukuranA[1]#kolom

  for i in range(barisA):
    iJelek = fitP.max()
    idxJelek = fitP.argmax()

  #Mengeliminasi anak yang memiliki jalan dengan rute terpanjang adalah individu paling jelek
    if fitAnak[i]<iJelek:
      fitP[idxJelek] = fitAnak[i]
      for j in range (kolomP):
        P[idxJelek][j] = Anak[i][j]
```

```
        return [P,fitP]

    fitAnak= funcObj(AnakM,daftarKota)
    #Mencetak semua yang perlu dilihat
    print(Panggil)
    print(AnakM)
    print(fitP)
    print(fitAnak)
    P = Elitism (Panggil,AnakM,fitP,fitAnak)
    print(P)
    print(fitP)
```

```
[[2 4 0 3 1]
 [3 4 1 2 0]
 [4 1 3 0 2]
 [1 0 4 3 2]]
[[3 2 1 4 0]
 [3 1 3 4 2]]
[28.90826195 20.83130956 20.89580466 29.18188057]
[23.60769887 19.36505818]
[array([[3, 1, 3, 4, 2],
       [3, 4, 1, 2, 0],
       [4, 1, 3, 0, 2],
       [3, 2, 1, 4, 0]]), array([19.36505818, 20.83130956, 20.89580466, 23.60769887])]
[19.36505818 20.83130956 20.89580466 23.60769887]
```

Fungsi Crossover:

```
# Inisialisasi paramater
PCX = 0.95 #Nilai daro Pxo
Pm = 0.01
ukuranPopulasi = 4
max_generasi = 5

# Inisialisasi populasi
P = initPopulasi (ukuranPopulasi,jumlahKota)
print("P0 :",P)

#melakukan evaluasi P
fitP = funcObj(P,daftarKota)
print("fitness populasi awal:",fitP)

print("fitness terbaik awal:",fitP.min())
print("rute awal :",P[fitP.argmin()])
for i in range(max_generasi):
  print("generasi :",i)
  Induk = funcSelection(P,fitP)
  #print("Induk :",Induk)
```

Hasil Akhir:

```
# Inisialisasi paramater
PCX = 0.95 #Nilai dari Pxo
Pm = 0.01
ukuranPopulasi = 4
max_generasi = 5

# Inisialisasi populasi
P = initPopulasi (ukuranPopulasi,jumlahKota)
print("P0 :",P)

#melakukan evaluasi P
fitP = funcObj(P,daftarKota)
print("fitness populasi awal:",fitP)

print("fitness terbaik awal:",fitP.min())
print("rute awal :",P[fitP.argmin()])
for i in range(max_generasi):
  print("generasi :",i)
  Induk = funcSelection(P,fitP)
  Anak = Induk
  rxo = random.random() #membangkitkan nilai random
  if rxo < PCX:#kondisi jika nilai random lebih kecil dari PCX maka akan masuk ke kondisi selanjutnya yaitu fungsi mutasiswap
    Anak = funcPMX(Induk)
    rm = random.random()
    if rm < Pm:
      Anak = MutasiSwap(Anak)#mutasi dengan parameter anak

  fitAnak = funcObj(Anak,daftarKota)#fitanak menggunakan fungsi funcobj dengan parameter Anak dan daftarKota

  print("fitness:",fitAnak.min())#tampilkan nilai minimal dari fit anak

  [P, fitP]= Elitism (P,Anak,fitP,fitAnak)

print("fitness populasi akhir:",fitP)
print("fitness akhir:",fitP.min())
print("rute akhir :",P[fitP.argmin()])
```

```
P0 : [[3 2 1 0 4]
 [2 4 1 3 0]
 [2 3 4 0 1]
 [2 3 4 0 1]]
fitness populasi awal: [29.18188057 20.89580466 29.18188057 29.18188057]
fitness terbaik awal: 20.895804660363012
rute awal : [2 4 1 3 0]
generasi : 0
fitness: 20.895804660363012
generasi : 1
fitness: 21.169423279024105
generasi : 2
fitness: 20.895804660363012
generasi : 3
fitness: 20.895804660363012
generasi : 4
fitness: 20.895804660363012
fitness populasi akhir: [20.89580466 20.89580466 20.89580466 20.89580466]
fitness akhir: 20.895804660363012
rute akhir : [2 4 1 3 0]
```

Mohon maaf untuk grafiknya masih Erorr pak dari saya

```
x = [i for i in range(ukuranPopulasi)]

import matplotlib.pyplot as plt
plt.plot(x, max_generasi)
```