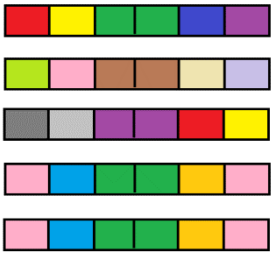
GENETIC ALGORITHM/MUHHAMAD AFFAN HASBY/1301174618/IF-41-10

kromosom adalah parameter yang menentukan solusi yang diusulkan untuk masalah yang coba dipecahkan oleh algoritma genetika. Himpunan semua solusi dikenal sebagai populasi. Kromosom sering direpresentasikan sebagai string biner, meskipun berbagai macam struktur data lain juga digunakan.



Desain chromosom yang saya buat yaitu dengan jumlah individu yang berada dalam List berjumlah 6 dan dengan populasi yaitu 5 dibawah ini adalah desainnya.

Untuk membuat codingnya saya menggunakan bahasa pyhton.

```
def individu (panjangin):
  a = []
  for i in range(panjangin):
    a.append(np.random.randint(0,2))
  return a
```

Hal pertama yang saya buat yaitu membuat individunya, codingannya seperti digambar..

Saya membuat fungsi untuk membuat sebuah individu dengan panjang 6. Saya memasukkan angka biner dengan random untuk nilai yang ada didalam list tersebut, untuk merandom nilai binernya saya menggunakan “np.random.randint(0,2)” lalu append untuk memasukkan nilainya kedalam list.

```
def populasi (ukuran_populasi):
  b = []
  for i in range(ukuran_populasi):
    b.append(individu(panjangin))
  return b
```

Setelah individu terbuat saya membuat populasi, dengan jumlah populasi yaitu 5. Untuk codingannya seperti ada digambar.

Saya membuat fungsi populasi dengan nama “populasi”, lalu saya membuat list kosong dengan parameter “b”, setelah itu saya membuat perulangan dengan menggunakan for, dengan panjang sesuai dengan populasi yang sudah saya tentukan yaitu 5. Didalam perulangan tersebut saya memanggil fungsi untuk nilainya dimasukkan kedalam list populasi. Lalu setelah itu di Return karena ini adalah sebuah fungsi.

Setelah itu saya membuat sebuah fungsi untuk mengubah dari genotype ke phenotype, karena saya menggunakan binnary untuk nilai yang ada didalam listnya, saya meggunakan rumus binary Dari rumus saya convert kedalam codingan phyton menjadi seperti digambar.

```
def gchf (ind):
  c=0
  for i in range(1,4):
    c= c+(2**(-i))
  x1 = ind[:4]
  x2 = ind[-3:]
  d=0
  e=0
  for i in range (1,4):
    d=d+(2**(-i))*x1[i-1]
    e=e+(2**(-i))*x2[i-1]
  hasil_x1 = rmin_x1 + ((rmax_x1-rmin_x1)/c)*d
  hasil_x2 = rmin_x2 + ((rmax_x2-rmin_x2)/c)*e
  return hasil_x1, hasil_x2
```

Pada codingan tersebut, individu dengan panjang 6, saya pecah menjadi 2 sesuai dengan rumus, dengan inisialisasi yaitu x1 dan x2. Setelah di pecah menjadi x1 dan x2 langsung saya hitung sesuai dengan rumusnya. Lalu karena ini adalah sebuah fungsi maka saya return.

Setelah itu saya menghitung nilai fitnesnya, saya menggunakan rumus yang ada didalam soal lalu saya convert kedalam codingan dengan menggunakan bahasa pemrograman bahasa pyhton menjadi seperti ini.

```
def fitnes (x1,x2):
  f = 1/(((((4-(2.1*(x1**2))+((x1**4)/3)))*x1**2)+(x1*x2)+((-4+4*x2**2))*x2**2)+0.01)
  return f
```

Setelah mendapatkan hasil perhitungan fitnesnya, saya membuat sebuah fungsi untuk menentukan Parent, untuk menentukan parent saya menggunakan Roulatte

```
def Roulet():
  total = 0
  for i in range (5):
    total = fit[i]

  r = np.random.uniform()
  indv = 1

  while( r > 0 ):
    r -= fit[indv]/total
    indv = indv+1
  return indv
```

Saya membuat sebuah fungsi dengan nama roulate. Didalam coding tersebut masukkan nilai fitnes yang terbaik yang nantinya akan dijadian Parent. Pemilihan parent disini bertujuan untuk mencari sebuah individu yang lebih baik yang nantinya akan dijadikan parent.

Setelah menncari parent, hasil dari parent tersebut saya crossover dengan codingan seperti berikut.

```
def cross(parent1, parent2):
  cross_poin = np.random.randint(1,panjangin-1)
  chil1 = np.hstack((parent1[0:cross_poin], parent2[cross_poin:]))
  chil2 = np.hstack((parent2[0:cross_poin], parent1[cross_poin:]))

  return chil1, chil2
```

codingan tersebut, mengambil 2 parent lalu setelah itu mengcrossover secara acak, untuk mengacaknya menggunakan variabel “cross_poin”, lalu di crossover sesuai dengan variabel “cross_poin”. Saya mesakkan nilai yang telah dicrossover tersebut kedalam variabel chil1 dan chil2. Lalu karena ini adalah sebuah fungsi saya return hasil chil1 dan chil2 tersebut.

Setelah membuat fungsi crossover saya membuat sebuah fungsi untuk memutasi nilai hasil dari crossover tersebut, fungsi tersebut saya namain dengan fungsi “mutasi”.

```
def mutasi (chil1,chil2):
  prob = np.random.uniform()
  if prob < 0.1:
    a1 = np.random.randint(0,6)
    a2 = np.random.randint(0,6)
    for i in range(6):
      if a1 == i:
        if chil1[i] == 0:
          chil1[i]= 1
        else : chil1[i]= 0
      if a2 == i:
        if chil2[i] ==0:
          chil2[i]=1
        else : chil2[i]=0

  return chil1, chil2
```

Dari fungsi tersebut saya membuat variabel prob, digunakan untuk mengacak nilai probabilitas, yang nanti probabilitasnya saya buat dengan nilai 0,1.

Sehingga jika nilai acak tersebut adalah kurang dari 0,1 maka nilai tersebut akan dimutasi, akan tetapi jika nilainya tidak kurang dari 0,1 maka tidak akan terjadi mutasi.

Setelah membuat sebuah fungsi mutasi saya membuat fungsi untuk pergantian generasi, fungsi pergantian generasi ini digunakan untuk mengganti individu sebelumnya menjadi individu yang memiliki nilai yang lebih baik.

```
population = populasi(ukuran_populasi)
for i in range(100):
  fit = []
  for i in range (ukuran_populasi):
    x1,x2 = gchf(population[i])
    fit.append(fitnes (x1,x2))
#  print(list(zip(population,fit)))
  print(fit)

  fitness = fit
  newPopulation = elitism(population, fitness, 5)
  while len(newPopulation) < 5:
    parent1, parent2 = roulate(fit)
    chil1,chil2 = cross(parent1, parent2)
    chil1,chil2 = mutasi(chil1, chil2)
    newPopulation += chil1
    newPopulation += chil2
  population = newPopulation
```

di dalam fungsi ini saya membuat populasi yaitu dengan jumlah 100, populasi yang telah dihitung dengan fungsi-fungsi diatas maka hasil populasi baru akan dihitung kembali sampai populasi tersebut memiliki nilai yang paling efisien. Perhitungan tersebut akan diulang terus menerus sampai dengan populasi yang telah ditentukan, dalam fungsi tersebut saya menentukan dengan jumlah populasi yaitu 100 populasi, dari 100 populasi terebut saya mendapat hasil nilai yang efisien.

Untuk hasil nilainya dapat dilihat pada gambar dibawah ini.

```
[0.5806312119492697, 0.1831600345486513, 0.055773970831240575, 0.0209126021085141, 0.009473384386497355]
[0.5806312119492697, 0.1831600345486513, 0.055773970831240575, 0.0209126021085141, 0.009473384386497355]
[0.5806312119492697, 0.1831600345486513, 0.055773970831240575, 0.0209126021085141, 0.009473384386497355]
[0.5806312119492697, 0.1831600345486513, 0.055773970831240575, 0.0209126021085141, 0.009473384386497355]
[0.5806312119492697, 0.1831600345486513, 0.055773970831240575, 0.0209126021085141, 0.009473384386497355]
```

Jadi nilai terbaik yang saya dapatkan karena ini adalah fungsi minimum yatu 0,00947338486497355 seperti yang ada pada gambar tersebut.