Laporan Analisis Mencari Nilai Minimum
dari Sebuah Rumus dengan Gentic Algorithm

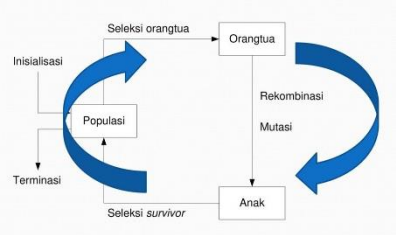
Diketahui sebuah rumus sebagai berikut :

$$f(x_1, x_2) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$$

Dengan batasan dengan batasan $-3 \leq x_1 \leq 3$ dan $-2 \leq x_2 \leq 2$, dicari nilai minimum dari fungsi tersebut. Bahasa yang saya gunakan untuk memecahkan persoalan tersebut adalah dengan bahasa pemrogramman python, dengan versi Python 3.6 dan untuk memvisualisasikannya menggunakan library matplotlib python.

A. Genetic Algorithm

Genetic Algorithm (GA) adalah bagian dari Evolutionary Algorithm yaitu suatu algoritma searching yang mencontoh proses evolusi alami dimana konsep utamanya adalah individu-individu yang paling unggul akan bertahan hidup, sedangkan individu-individu yang lemah akan punah.



Skema Genetic Algorithm

Genetic algorithm merupakan salah satu algoritma yang sangat tepat digunakan dalam menyelesaikan masalah optimasi mencari kemungkinan dari calon solusi untuk mendapatkan solusi yang optimal dalam penyelesaian masalah. Dalam program ini, genetic algorithm digunakan untuk memecahkan persoalan dengan cara membandingkan nilai f(x_1,x_2) yang sebelumnya dengan nilai baru f(x_1,x_2).

B. **Analisis dan penjelasan strategi penyelesaian masalah**

Pertama-tama, kita lakukan import library dari python sebagai berikut :

```
1  import random
2  import numpy as np
3  import matplotlib.pyplot as plt
```

Pada baris ke-1 digunakan untuk pembangkit bilangan acak, Numpy memiliki kegunaan untuk operasi vektor dan matriks, pada baris3 digunakan untuk visualisasi plot 3D.

Setelah itu lakukan inisialisasi variabel sebagai berikut :

```
13  Rax1          = 3     #Batas Atas
14  Rbx1          = -3    #Batas Bawah
15  Rax2          = 2     #Batas Atas
16  Rbx2          = -2    #Batas Bawah
17  mutationRate  = 0.06  #Probability Mutasi
18  ProbCrossOver = 0.67  #Probability CrossOver
19  UkPop         = 6     #Ukuran Populasi
20  JumGen        = 10    #Jumlah Generasi
```

Variabel Rax1 dan Rbx1 digunakan untuk batasan nilai dari x1 sedangkan Rax2 dan Rbx2 digunakan untuk membatasi nilai dari x2,variable mutationRate digunakan untuk probabilitas dari mutasi ,variable ProbCross Over digunakan untuk probabilitas dari CrossOver, variable UkPop digunakan untuk mengeset berapa banyak populasi yang akan digenerate,variable JumGen digunakan untuk membangkitkan sebanyak apa Gen dan juga digunakan sebai kondisi berhenti .

```
24  def rumus(x1,x2):
25      return(4-2.1*x1**2+x1**4/3)*x1**2+x1*x2+(-4+4*x2**2)*x2**2
```

Pada baris ke 24 digunakan untuk memgenerate fungsi rumus untuk mencari nilai dari x1 dan x2 dengan batasan yang telah dibuat

```
44  def Kromosomx1(Populasi):
45      MaxSum = 0
46      Penampung1 =(JumGen/2)-1
47      LenghtGen1 = int (JumGen)
48      x1 = np.zeros([UkPop],  dtype = float)
49      for kk in range(1,LenghtGen1):
50          MaxSum = MaxSum + 2**(-kk)
51      for ii in range(0,UkPop):
52          for jj in range (1,LenghtGen1):
53              x1[ii] = x1[ii] + Populasi[ii][jj-1]*(2**-jj)
54          x1[ii] = Rbx1 + ((Rax1-Rbx1)/MaxSum) * x1[ii]
55      return x1
```

```
59  def Kromosomx2(Populasi):
60      JumGen2=(JumGen/2)-1
61      LenghtGen2 = int (JumGen)
62      MaxSum = 0
63      x2 = np.zeros([UkPop],  dtype = float)
64      for kk in range(1,LenghtGen2):
65          MaxSum = MaxSum + 2**(-kk)
66      for ii in range(0,UkPop):
67          for jj in range (1,LenghtGen2):
68              x2[ii] = x2[ii] + Populasi[ii][jj-1]*(2**-jj)
69          x2[ii] = Rbx2 + ((Rax2-Rbx2)/MaxSum) * x2[ii]
70      return x2
```

Selanjutnya buat fungsi decode Kromosomx1 dan Kromosomx2 digunakan untuk mendecode individu kedalam kromosom dalam bentuk biner untuk Kromosomx1 menggunakan batasan $-3 \leq x_1 \leq 3$ dan Kromosomx2 dengan batasan $-2 \leq x_2 \leq 2$

```
def Fitnees(x1,x2):
    BilKecil = 0.001
    h = 1/(4-2.1*x1**2+x1**4/3)*x1**2+x1*x2+(-4+4*x2**2)*x2**2
    Fitnees = 1 / (h + BilKecil)
    return Fitnees
```

Membuat fungsi Fitnees untuk mencari individu dengan berkualitas tinggi,jika untuk meminimalisasi fungsi kita bisa menggunakan rumus 1/(h+BilKecil),dimana BilKecil merupakan konstanta yang digunakan untuk menghindari pembagian nol ketika h=0

```
def Parent(Populasi, Fitnees):
    Pindex=0
    JumlahFitness = sum(Fitnees)
    for i in range(len(Populasi)):
        if (Fitnees[i]/JumlahFitness) > np.random.uniform(0.1):
            Pindex = i
            break
        i = i + 1
    return Populasi[Pindex]
```

Membuat fungsi Roulette Wheel untuk mencari parent dari sebuah individu dengan menggunakan nilai Fitnees .Proses pemilihan dilakukan sebanyak N kali.Pada setiap putaran,individu yang ditunjuk oleh jaruk dinyatakan terpilih sebagai parent.

```
def CrossOver(Parent1,Parent2,JumGen):
    if(np.random.rand()<ProbCrossOver):
        limit   = int(1+(np.ceil(np.random.randint(JumGen-1))))
        Child1  = np.concatenate([Parent1[0:limit], Parent2[limit:JumGen]])
        Child2  = np.concatenate([Parent2[0:limit], Parent1[limit:JumGen]])
        return Child1
```

Membuat fungsi CrossOver yaitu dengan membuat kondisi dimana jika hasil dari pembangkitan bilangan acak secara random < dari probabilitasCrossOver dimana Probabilitas CrossOver diantara 60%-70%.CrossOver merupakan salah satu komponen paling penting dalam GA. Hal ini disebabkan dengan adanya perkainan silang, solusi yang dihasilkan akan menuju konvergen pada suatu titik tertentu secara acak.

```
def MutasiKromosom(Kromosom,JumGen, mutationRate):
    MutKrom = Kromosom
    for i in range(JumGen):
        if (np.random.uniform(0.1) < mutationRate):
            if (Kromosom[i]==0):
                MutKrom[i] = 1
            else:
                MutKrom[i] = 0
    return np.concatenate(MutKrom)
```

Membuat fungsi Mutasi untuk mengembalikan informasi bit yang hilang akibat CrossOver. Mutasi dilakukan parameter mutationRate yaitu probabilitas yang sanget kecil.Dengan cara membangkitkan bilangan acak secara random < mutationRate

3. Parameter GA yang Optimum

Saat Ukuran Populasi 10 dan JumGen 100

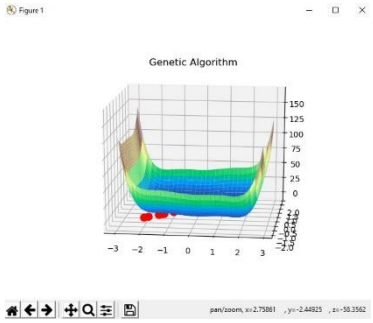
```
Rax1          = 3
Rbx1          = -3
Rax2          = 2
Rbx2          = -2
mutationRate  = 0.06
ProbCrossOver = 0.67
UkPop         = 10
JumGen        = 100
```

```
Kromosom Terbaik
-2.0159321675241695

Hasil Dekode Kromosom Terbaik untuk X1
-2.0159321675241695

Hasil Dekode Kromosom Terbaik untuk X2
-1.3439547783494463

Hasil dari nilai x1 dan x2 adalah
0.005664723753437614
```



Nilai dari x1adalah -2,0159 dan nilai dari x2 adalah -1.3439. Maka nilai minimum dari adalah 0.005 dan Kromosom terbaik -2.0159