Penerapan Genetic Algoritma untuk Menemukan Nilai Minimum Suatu Fungsi

Nur Fuad Azizi

Fakultas Informatika, Telkom University

Jalan Telekomunikasi no. 1, Bojongsoang, Kabupaten Bandung

A. Analisis Masalah

Permasalahan yang diteliti dalam penerapan algoritma genetik ini adalah untuk mencari nilai minimum dari fungsi berikut.

$$h(x_1, x_2) = \cos(x_1)\sin(x_2) - \frac{x_1}{(x_2^2 + 1)}$$

Dengan batasan $-1 \leq x_1 \leq 2$ dan $-1 \leq x_2 \leq 1$. Output yang diharapkan dari program yang dibangun adalah **kromosom terbaik** dan juga nilai $\boldsymbol{x_1}$ dan $\boldsymbol{x_2}$ dari hasil dekode kromosom terbaik tersebut.

Algoritma genetik dapat mencari nilai kromosom terbaik, karena algoritma genetik akan menyeleksi setiap individu dengan iterasi. Dalam setiap iterasi dilakukan pemilihan orang tua, proses mutasi, dan juga proses crossover untuk membentuk individu yang lebih baik lagi. Semakin banyak iterasi, maka hasil akan semakin akurat. Dengan menerapkan konsep algoritma genetik, diharapkan suatu sistem dapat mengetahui sendiri hasil yang paling baik saat memecahkan suatu masalah.

B. Strategi Penyelesaian Masalah

1. Membangun kelas-kelas yang dibutuhkan

Disini saya menggunakan 2 kelas, yaitu kelas Individu dan kelas Populasi.

```
class Individu:
    # Class yang akan menampung suatu individu
    # Attribut class Individu adalah kromosom,
    # nilai x1, x2, h, dan fitness
    a = 3
    arr_chrom = []
    x1, x2 = 0, 0
    h = 0
    fitness = 0
```

```
class Populasi():
    # Class yang akan menampung sebuah populasi
    # dimana populasi terdiri dari banyak individu
    # Attribut utama class Populasi adalah array of Individu,
    # total fitness, dan total populasi
    arr_individu = []
    bestIndividu = 0
    sum_pop = 20
    sum_fitness = 0
    NKromosom = 10
```

2. Melakukan encoding dan decoding kromosom

Encoding dan decoding kromosom yang dibangun dengan cara dirandom. Bentuk kromosom yang dibuat adalah dengan representasi biner.

3. Menghitung fungsi h dan fungsi fitness untuk tiap-tiap individu

```
def fungsiH(self):
    x1 = self.x1
    x2 = self.x2
    cosx1 = np.cos(x1)
    sinx2 = np.sin(x2)
    self.h = cosx1 * sinx2 - (x1/(x2*x2+1))

def hitungFitness(self):
    h = self.h
    a = self.a
    self.fitness = 1/(h+a)
```

Penghitungan fungsi h menggunakan rumus diatas. Sedangkan untuk menghitung fitness, karena ini kasus minimasi, maka rumusnya adalah $\frac{1}{h+a}$ dengan a adalah bilangan positif untuk menghindari perolehan nilai h = 0.

4. Menyeleksi orang tua

Untuk menentukan orang tua, menggunakan dua cara, yaitu elite dan roullette wheel.

5. Membentuk individu baru dari orang tua dengan mutasi dan rekombinasi

```
# LAKUKAN REKOMBINASI
while (size < sizeInduk*2):
    rand1 = random.randint(0,sizeInduk)
    rand2 = random.randint(0,sizeInduk)
    if (rand1 != rand2):
        induk1 = self.arr_individu[rand1]
        induk2 = self.arr_individu[rand2]
        anak1 = self.Copy(self.arr_individu[rand1])
        anak2 = self.Copy(self.arr_individu[rand2])

        anak1.crossover(induk1,induk2,1)
        anak2.crossover(induk1,induk2,2)
        nextGen.append(anak1)
        nextGen.append(anak2)
        size += 2
```

```
# LAKUKAN MUTASI
while(size < self.sum_pop):
    randind = random.randint(0,sizeInduk)
    if (random.random() <= doMutate):
        anak = self.Copy(nextGen[randind])
        anak.mutate(pm)
        nextGen.append(anak)
        size+=1

nextPop = Populasi(nextGen)
return nextPop
```

Selanjutnya adalah menambahkan individu anak yang akan dijadikan sebagai penerus generasi selanjutnya. Dua metode yang dipakai adalah mutasi dan rekombinasi biner (*crossover*). Saya menerapkan *uniform crossover* dalam melakukan rekombinasi.

6. Melakukan proses pergantian generasi

Hasil dari proses-proses sebelumnya ditampung ke dalam suatu array of Individu untuk selanjutnya digunakan sebagai penerus generasi selanjutnya.

7. Menentukan kondisi berhenti

Program ini dibangun dengan populasi sebanyak 20 dan akan berhenti saat mencapai generasi ke-50.

```
# ============================== MAIN ======

sum_gen = 50
Gen = []
population = Populasi(None)
AllBest=[]
Indeks=[]
i=0

for i in range(sum_gen):
    Gen.append(population)
    best = Gen[i].bestIndividu
    print("=====Gen ke-{}=====".format(i+1))
    print("Nilai x1:",round(best.x1,3))
    print("Nilai x2:",round(best.x2,3))
    print("Fitness :",round(best.fitness,3))
    population = Gen[i].Regenerasi()

print()
print("Kesimpulan Individu terbaik")
print("kromosom       : ", end="")
best.printIndividu()
print()
print("Nilai x1       :",best.x1)
print("Nilai x2       :",best.x2)
print("Nilai h(x1,x2):",best.h)
```

```
=====Gen ke-1=====
Nilai x1: 1.903
Nilai x2: -0.032
Fitness : 0.901
=====Gen ke-2=====
Nilai x1: 1.903
Nilai x2: -0.032
Fitness : 0.901
=====Gen ke-3=====
Nilai x1: 1.903
Nilai x2: 0.097
Fitness : 0.923
=====Gen ke-4=====
Nilai x1: 1.903
Nilai x2: 0.097
Fitness : 0.923
=====Gen ke-5=====
Nilai x1: 1.903
Nilai x2: 0.097
Fitness : 0.923
=====Gen ke-6=====
Nilai x1: 1.903
Nilai x2: 0.097
Fitness : 0.923
=====Gen ke-7=====
Nilai x1: 1.903
Nilai x2: 0.097
Fitness : 0.923
```

```
=====Gen ke-45=====
Nilai x1: 2.0
Nilai x2: 0.097
Fitness : 1.022
=====Gen ke-46=====
Nilai x1: 2.0
Nilai x2: 0.097
Fitness : 1.022
=====Gen ke-47=====
Nilai x1: 2.0
Nilai x2: 0.097
Fitness : 1.022
=====Gen ke-48=====
Nilai x1: 2.0
Nilai x2: 0.097
Fitness : 1.022
=====Gen ke-49=====
Nilai x1: 2.0
Nilai x2: 0.097
Fitness : 1.022
=====Gen ke-50=====
Nilai x1: 2.0
Nilai x2: 0.097
Fitness : 1.022

Kesimpulan Individu terbaik
kromosom      : [1, 1, 1, 1, 1, 1, 0, 0, 0, 1]
Nilai x1      : 2.0
Nilai x2      : 0.09677419354838701
Nilai h(x1,x2): -2.0216527429522126
```

C. Parameter GA

- Total populasi tiap generasi: 20
- Total generasi: 50
- Probabilitas mutasi: 0.4
- Panjang kromosom: 10
- Nilai a: 3

D. KESIMPULAN

Dari percobaan diatas, nilai paling minimum h(x1,x2) yang bisa dicapai adalah -2,02. Untuk nilai x1-nya adalah 2 dan nilai untuk x2-nya adalah 0,09.

REFERENSI

[1] The MathWorks, Inc., *How The Genetic Algorithm Works* Retrieved from https://www.mathworks.com/help/gads/how-the-genetic-algorithm-works.html