Tugas Pemrograman 01 Searching : Genetic Algorithm (GA)

CII – 2M3 Pengantar Kecerdasan Buata

Semester Genap 2021/2022

Dosen Pembimbing :

Izzatul Ummah, S.T., M.T.

Disusun Oleh :

Kelompok 12 - IF4411

1301201472 – Fadhly Al-farizi

1301204335 – Muhammad Rieza Fachrezi

S1 Informatika

Telkom University

BAB I

Pendahuluan

- Definisi Tugas

  Melakukan analisis dan juga desain ***Genetic Algorithm (GA)*** serta implementasinya dalam program komputer menggunakan Bahasa pemrograman ***python*** untuk mencari nilai x dan y sehingga diperoleh ***nilai minimum*** dari fungsi berikut :

  $$h(x,y) = \frac{(\cos x + \sin y)^2}{x^2 + y^2}$$

  Dengan Batasan nilai x dan y sebagai berikut :

  $$-5 \leq x \leq 5 \text{ dan } -5 \leq y \leq 5$$

  Hal yang dianalisis dan desain :

  - Desain kromosom dan metode decode-nya.
  - Ukuran populasi.
  - Metode pemilihan orangtua
  - Metode operasi genetic (pindah silang dan mutase)
  - Probabilitas operasi genetik (Pc dan Pm)
  - Metode pergantian generasi (seleksi survivor)
  - Kriteria penghentian evolusi (loop)

  Dengan proses yang dibangun dan implementasikan pada program komputer :

  - Dekode Kromosom
  - Perhitungan fitness
  - Pemilihan orangtua
  - Crossover (pindah silang)
  - Mutasi
  - Pergantian Generasi

- Output Program
  - ***Kromosom terbaik***
  - ***Nilai x dan y*** hasil decode kromosom terbaik

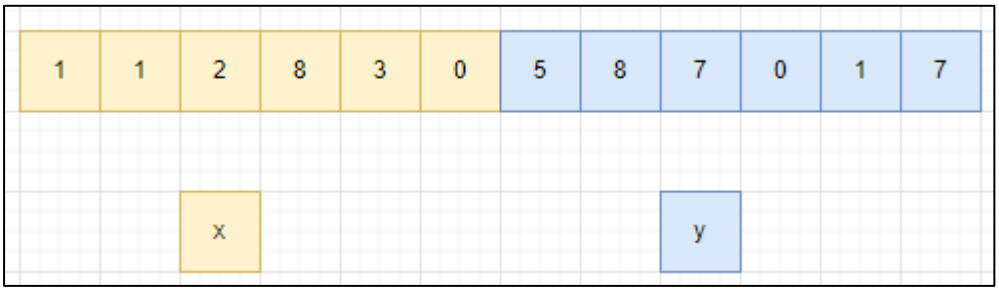
BAB II

Isi

1) Desain Kromosom dan Metode Dekode Kromosom.

   Desain Kromosom yang digunakan adalah **Representasi Integer** dengan selang bilangan antara **0 – 9.** Metode dekode yang digunakan juga merupakan metode untuk desain kromosom representasi integer, yaitu metode **Dekode Integer.**

   - Desain kromosom dengn representasi integer.



   - Perhitungan untuk dekode representasi integer.

$$x = r_{min} + \frac{r_{max} - r_{min}}{\Sigma_{i=1}^{N} 9 * 10^{-i}} (g_1 * 10^{-1} + g_1 * 10^{-2} + \cdots + g_N * 10^{-N})$$

   - Perhitungan untuk dekode representasi integer (Implementasi Program).

```
def decode_kromInt(kromosom) :
  #Dekode Kromosom
  decode_X = batas_bawahX + ((batas_atasX - batas_bawahX)/(9*sum(temp_decode))) * ((kromosom[0]*temp_decode[0]) + (kromosom[1]*temp_decode[1]) +
             (kromosom[2]*temp_decode[2]) + (kromosom[3]*temp_decode[3]) + (kromosom[4]*temp_decode[4]) + (kromosom[5]*temp_decode[5]))
  decode_Y = batas_bawahY + ((batas_atasY - batas_bawahY)/(9*sum(temp_decode))) * ((kromosom[6]*temp_decode[0]) + (kromosom[7]*temp_decode[1]) +
             (kromosom[8]*temp_decode[2]) + (kromosom[9]*temp_decode[3]) + (kromosom[10]*temp_decode[4]) + (kromosom[11]*temp_decode[5]))
  return decode_X, decode_Y
```

2) Ukuran Populasi.

   Populasi yang digunakan sebanyak **10 Generasi** dengan setiap generasinya terdapat **10 Kromosom** dan tiap-tiap kromosomnya terdapat **12 Gen.**

   - Pembuatan Populasi Awal

```
#Inisialisasi Populasi Awal
popcount = 10
populasi_awal = np.random.randint(10, size =(popcount,12))
```

   Setelah itu ada perhitungan nilai fitness yang minimum dari suatu fungsi yang diberikan dengan Batasan-batasan x dan y yang ada.

- Rumus untuk nilai fitness yang minimal

$$f = \frac{1}{h + a}$$

- Batasan x dan y

```
batas_atasX = 5
batas_bawahX = -5
batas_atasY = 5
batas_bawahY = -5
```

- Perhitungan nilai fitness

```
def fitness(x, y) :
  #Perhitungan Nilai Fitness Minimal
  return 1/(1+(((np.cos(x) + np.sin(y))**2)/(x**2 + y**2)))
```

3) Metode Pemilihan Orangtua.

Pemilihan orang tua yang akan digunakan untuk menghasilkan generasi selanjutnya pada bagian crossover menggunakan metode **Roulette Wheels**.

- Pemilihan Orangtua menggunakan Roulette Wheels (Implementasi Program)

```
def find_parent(nilai_fitness) :
  #Pencarian Index Parent menggunakan Roulette wheels
  jumlah = sum(nilai_fitness)
  r = np.random.random()
  index = 0
  while r > 0 and index < 9 :
    r = r - (nilai_fitness[index]/jumlah)
    index += 1
  return index
```

4) Metode Operasi Genetik (Pindah Silang dan Mutasi).

Crossover atau penyilangan antara orangtua kami menggunakan metode **Single Point** dengan pengambilan titik potong secara acak antara indeks ke-0 hingga ke-11. Seorang child memiliki probabilitas untuk melakukan crossover sebesar **0.8 atau 80%.**

- Implementasi Program untuk crossover

```
def crossover(new_populasi, nilai_fitness) :
  #Crossover populasi untuk menghasilkan turunan baru dari parent
  for i in range(popcount//2) :
    p1 = find_parent(nilai_fitness)
    p2 = find_parent(nilai_fitness)
    if (np.random.random() < 0.8):
      tipot = np.random.randint(12)
      child1 = np.array(list(populasi_awal[p1][tipot:]) + list(populasi_awal[p2][:tipot]))
      child2 = np.array(list(populasi_awal[p2][tipot:]) + list(populasi_awal[p1][:tipot]))
      new_populasi.append(child1)
      new_populasi.append(child2)
    else:
      new_populasi.append(populasi_awal[p1])
      new_populasi.append(populasi_awal[p2])
```

Setelah kromosom melakukan crossover terdapat probabilitas sebesar **0.05 atau 5%** untuk kromosom tersebut melakukan mutase.

- Implementasi Program untuk Mutasi.

```
def mutasi(new_populasi) :
  #Mutasi populasi representasi integer (Memilih nilai secara acak)
  for i in range(popcount):
    for j in range(12):
      if (np.random.random() < 0.05):
        new_populasi[i][j] = np.random.randint(10)
```

5) Probabilitas Operasi Genetik (P_c dan P_m).

   Seperti yang disebutkan sebelumnya untuk probabilitas crossover (P_c) sebesar **0.8 atau 80%** dan untuk probabilitas mutasi sebesar **0.05 atau 5%.**

6) Metode Pergantian Generasi (Seleksi Survivor).

   Seleksi survivor atau pergantian generasi menggunakan metode Generational model yang mana tadi memiliki probabilitas crossover (P_c) sebesar **0.8 atau 80%** dan untuk probabilitas mutasi sebesar **0.05 atau 5%.** Disertai dengan **Elitisme**, elitism merupakan suatu metode yang digunakan untuk mengganti kromosom yang memiliki nilai fitness terendah pada generasi berikutnya dengan kromosom yang memiliki nilai fitness tertinggi pada generasi sebelumnya.

- Implementasi Program untuk Elitisme.

```
#Elitisme
nilai_childfitness = []
for j in range(popcount):
  x,y = decode_kromInt(new_populasi[j])
  nilai_childfitness.append(fitness(x,y))
badkrom = new_populasi[findworst(nilai_childfitness)]
xbest, ybest = decode_kromInt(bestkrom)
xbad, ybad = decode_kromInt(badkrom)
if fitness(xbest, ybest) > fitness(xbad, ybad):
  new_populasi[findworst(nilai_childfitness)] = bestkrom
```

```
def findbest(nilai_fitness):
  #Cari index kromosom dengan fitness yang terbesar
  idx = 0
  for i in range(len(nilai_fitness)):
    if (nilai_fitness[i] > nilai_fitness[idx]):
      idx = i
  return idx

def findworst(nilai_fitness):
  #Cari index kromosom dengan fitness yang terkecil
  idx = 0
  for i in range(len(nilai_fitness)):
    if (nilai_fitness[i] < nilai_fitness[idx]):
      idx = i
  return idx
```

7) Kriteria Penghentian Evolusi (Loop).

   Generasi yang digunakan sebanyak **10 generasi**, maka program atau evolusi akan terhenti apabila generasi sudah mencapai 10.

- Implementasi Program untuk Penghentian Evolusi

```
#MAIN PROGRAM
populasi = populasi_awal
generation = 10
best_fitness = []
bad_fitness = []

#Generation Loop
for gen in range(generation):
  #Output populasi awal
  if gen == 0 :
    print("Populasi Awal/Gen 1:")
    for i in range(popcount):
      print("Kromosom",i+1,":", populasi_awal[i])
    print()

  #Output Populasi Tiap Generasi
  if gen+1 != generation :
    print("========================= Generasi ke", gen+1,"=========================")
    nilai_fitness = []
    new_populasi = []

    #Output populasi dengan dekode dan fitness
    for i in range(popcount) :
      x,y = decode_kromInt(populasi[i])
      nilai_fitness.append(fitness(x,y))
      print("Kromosom", i+1, "    :", populasi[i])
      print("Dekode Titik X :", x)
      print("Dekode Titik Y :", y)
      print("Fitness        :", fitness(x,y))
      print()
```

8) Kromosom Terbaik.

Kromosom terbaik didapatkan setelah evolusi berhenti atau setelah generasi menyentuh 10.

- Implementasi Program untuk Kromosom Terbaik.

```
#Output Kromosom Terbaik
bestkrom = populasi[findbest(nilai_fitness)]
x,y = decode_kromInt(bestkrom)
best_fitness.append(fitness(x, y))
print("========================= Kromosom Terbaik =========================")
print("Kromosom terbaik :", bestkrom)
print("Dekode Titik X   :", x)
print("Dekode Titik Y   :", y)
print("Fitness          :", fitness(x,y))
```

- Contoh Output pada Program

```
======================== Kromosom Terbaik ========================
Kromosom terbaik : [1 1 2 8 3 0 5 8 7 0 1 7]
Dekode Titik X   : -3.8716988716988716
Dekode Titik Y   : 0.8701758701758697
Fitness          : 0.9999762511669371
```

9) Nilai x dan y hasil Dekode Kromosom Terbaik.

Nilai x dan y didapatkan setelah mendapat kromosom terbaik yang terjadi apabila evolusi berhenti atau setelah generasi menyentuh 10.

- Implementasi Program untuk Nilai Dekode x dan y Terbaik.

```
#Output Kromosom Terbaik
bestkrom = populasi[findbest(nilai_fitness)]
x,y = decode_kromInt(bestkrom)
best_fitness.append(fitness(x, y))
print("======================== Kromosom Terbaik ========================")
print("Kromosom terbaik :", bestkrom)
print("Dekode Titik X   :", x)
print("Dekode Titik Y   :", y)
print("Fitness          :", fitness(x,y))
```

- Contoh Output pada Program

```
======================== Kromosom Terbaik ========================
Kromosom terbaik : [1 1 2 8 3 0 5 8 7 0 1 7]
Dekode Titik X   : -3.8716988716988716
Dekode Titik Y   : 0.8701758701758697
Fitness          : 0.9999762511669371
```

BAB III

Penutup

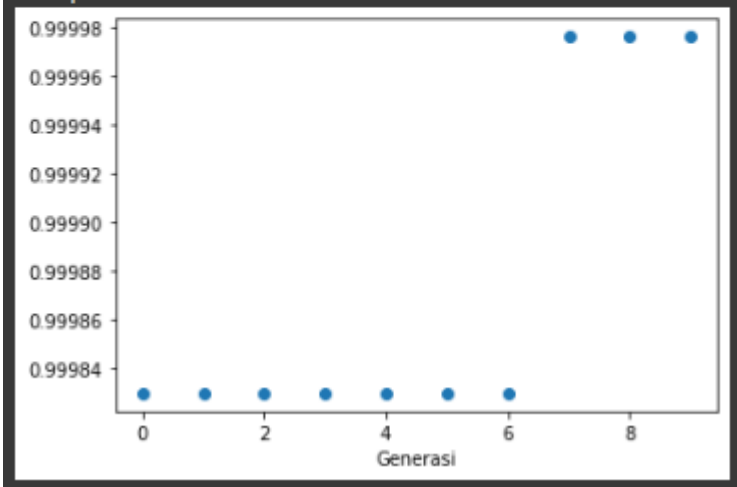
1) Kesimpulan

Algoritma GA (Genetic Algorithm) ini tidak terlalu efektif untuk mencari titik aboslut paling minimum karena semakin banyaknya generasi yang dibuat, maka titik fitness terbaik yang dapat dihasilkan tidak akan jauh berbeda nilainya dengan generasi sebelumnya atau bahkan tidak berubah sama sekali. Seperti yang digambarkan pada diagram berikut ini :

- Implementasi Program untuk Membuat Scatter Plot dari Nilai Fitness Terbaik setiap Generasi.

```
X = np.arange(len(best_fitness))
Y = best_fitness
fig, ax = plt.subplots()
ax.set_xlabel("Generasi")
ax.scatter(X, Y)
```

```
<matplotlib.collections.PathCollection at 0x7fe3e3a53a50>
```



BAB IV

Tambahan

1) Peran Anggota Kelompok

- Fadhly Al-farizi – 130120147

  Berperan dalam membangun program GA (40%), berperan dalam pembuatan video presentasi (50%), dan berperan dalam penulisan laporan (60%).

- Muhammad Rieza Fachrezi - 1301204335

  Berperan dalam membangun program GA (60%), berperan dalam pembuatan video presentasi (50%), dan berperan dalam penulisan laporan (40%).

2) Link Video Presentasi

https://www.youtube.com/watch?v=ODv4qS6XFEE