LAPORAN OBSERVASI
TUGAS PEMROGRAMAN 01 – GENETIC ALGORITHM

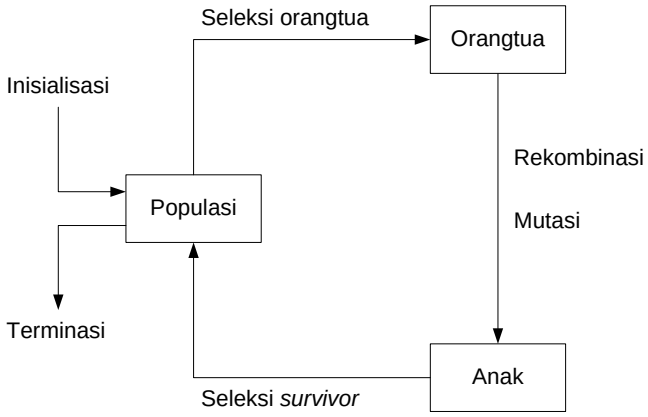
diajukan untuk memenuhi tugas mata kuliah (CII – 2M3) Pengantar Kecerdasan Buatan

oleh Kelompok 18:
Otniel Abiezer (NIM 1301180469)
Muhammad Haidir Ali (NIM 1301180205)

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
BANDUNG
2021

DAFTAR ISI

1. Skema Umum Genetic Algorithm



Atau jika dituliskan per langkah adalah sebagai berikut:
1.1.Inisialisasi
1.2.Menghasilkan Populasi
1.3.Seleksi Orang Tua (*Parent Selection*)
1.4.Rekombinasi (*Crossover*)
1.5.Mutasi (*Mutation*)
1.6 Seleksi Survivor (*Survivor Selection*)
1.7 Diulang sampai terminasi

2. Macam-macam Strategi

Dari langkah-langkah di atas, ada beberapa strategi yang dapat diterapkan pada genetic algorithm:

2.1.Inisialisasi

Pada inisialisasi, yaitu langkah untuk menentukan representasi individu (*encode* kromosom), strategi-strategi yang dapat dipakai adalah:

a. Representasi Biner
b. Representasi Integer
c. Representasi Real
d. Representasi Permutasi

Selain itu di inisialisasi juga menentukan perhitungan nilai fitness. Nilai fitness adalah nilai kecocokan suatu individu kromosom terhadap fungsi objektif yang dicari. Perhitungan nilai fitness terbagi 2, yaitu :

a. Maksimasi

$$f = h$$

b. Minimasi

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

2.2. Populasi

Populasi adalah kumpulan individu yang terbentuk. Untuk jumlahnya dapat ditentukan bebas, tetapi idealnya adalah di atas 10.

2.3. Seleksi Orang Tua (*Parent Selection*)

Seleksi dari 2 individu untuk dijadikan orang tua agar menghasilkan individu baru lagi untuk generasi berikutnya, metode pemilihan orang tua:

a. Roulette Wheel
b. Baker's SUS (Stochastic Universal Sampling)
c. Tournament Selection

2.4. Rekombinasi (*Crossover*)

Proses untuk menukar isi gen dari kromosom yang dimiliki oleh orang tua kepada anaknya. Metodenya dibagi berdasarkan representasi yang dipilih di nomor 1:

a. Representasi Biner
   - Rekombinasi satu titik (*1-point crossover*)
   - Rekombinasi banyak titik (*Multipoint crossover*)
   - Rekombinasi seragam (*Uniform crossover)*
b. Representasi Integer
   - Rekombinasi satu titik (*1-point crossover*)
   - Rekombinasi banyak titik (*Multipoint crossover*)
   - Rekombinasi seragam (*Uniform crossover)*
c. Representasi Real
   - *Discrete Crossover*
   - *Single Arithmetic Crossover*
   - *Simple Arithmetic Crossover*
   - *Whole Arithmetic Crossover*
d. Representasi Permutasi
   - *Order Crossover*
   - *Partially Mapped Crossover*
   - *Cycle Crossover*
   - *Edge Recombination*

2.5. Mutasi

Perubahan yang bersifat kecil, acak, berbahaya, dan jarang terjadi pada gen. Mutasi pada genetic algorithm diperlukan untuk mendapatkan data yang tidak terjebak pada optimum local. Metodenya dibagi berdasarkan representasi di nomor 1 antara lain:

a. Representasi Biner
   Membalikkan gen (1 menjadi 0 dan sebaliknya)
b. Representasi Integer
   - Membalik nilai integer
   - Memilih nilai secara acak
   - Mutasi *Creep* (Perlahan)
c. Representasi Real
   - Mutasi *Uniform*
   - Mutasi *Non-Uniform* dengan distribusi tetap
d. Representasi Permutasi
   - Pertukaran (*Swap*)
   - Penyisipan *(Insert)*

- Pengacakan (*Scramble*)
- Pembalikan (*Inversion*)

2.6. Seleksi Survivor (*Survivor Selection*)

Menyeleksi individu mana yang layak untuk dilanjutkan ke generasi berikutnya. Metodenya sebagai berikut:

a. Generational Model
b. Steady State Model

2.7. Terminasi

Cara agar memberhentikan genetic algorithm agar tidak jalan terus (*infinite loop*). Metodenya sebagai berikut :

a. Max iteration (max generation)
b. Time limit
c. Fitness plateau
d. Fitness threshold
e. Population and generational diversity

3. Hal yang Diobservasi

3.1. Desain Kromosom dan Metode Pengkodean

Desain kromosom adalah sepanjang 16 kotak yang berisi gen di mana gen ke-1 sampai ke-8 merupakan gen dari x dan gen ke-9 sampai ke-16 merupakan gen dari y. Metode pengkodean adalah menggunakan representasi biner, karena lebih optimal dan mudah untuk diimplementasikan.

```
def createRandomIndividu():
    #Membuat kromosom individu dengan 16 gen yang masing-masing berupa biner
    indv = []
    for i in range(0,16):
        indv.append(random.randint(0,1))
    return indv
```

3.2. Ukuran Populasi

Ukuran populasi adalah 20

```
def createPopulasi():
    #Menciptakan populasi awal sebanyak 20
    populasi_kromosom = []
    populasi_fitness = []
    for i in range (0,20):
        while True: #Menjamin nilai fitness tidak negatif
            indv = createRandomIndividu()

            x = binaryDecodeToReal_X(indv)
            y = binaryDecodeToReal_Y(indv)

            nilai_fitness = hitungFitness(x,y)
            if (nilai_fitness >= 0):
                populasi_fitness.append(nilai_fitness)
                populasi_kromosom.append(indv)
                break

    return populasi_kromosom, populasi_fitness
```

3.3.Pemilihan Orang Tua

Dengan Roulette Wheel

```python
#3. Pemilihan Orang Tua
def parentSelection(populasi_fitness):
    #Dengan Roullete Wheel
    total_fitness = 0
    for i in range (0, len(populasi_fitness)):
        total_fitness += populasi_fitness[i]

    roullete = random.uniform(0, total_fitness)

    for i in range (0, len(populasi_fitness)):
        roullete -= populasi_fitness[i]
        if (roullete <= 0):
            return i
```

3.4.Pemilihan dan Teknik Operasi Genetik

a. *Crossover*

Representasi Biner di 2 titik, yaitu titik 5 dan 11

```python
#4. CrossOver
def crossOver(kromosom1, kromosom2):
    #Rekombinasi Biner di 2 titik, yaitu di 5 dan 11
    rekombinasi_1 = []
    rekombinasi_2 = []
    if (mesinGacha(70)):    #Peluang CrossOver 70%
        for i in range(0, 16):
            if (5 <= i <= 11):
                rekombinasi_1.append(kromosom2[i])
                rekombinasi_2.append(kromosom1[i])
            else:
                rekombinasi_1.append(kromosom1[i])
                rekombinasi_2.append(kromosom2[i])
    else:
        for i in range(0, 16):
            rekombinasi_1.append(kromosom1[i])
            rekombinasi_2.append(kromosom2[i])

    return rekombinasi_1, rekombinasi_2
```

b. Mutasi

Representasi Biner

```python
#5. Mutasi
def mutasi(anak):
    #Mutasi Biner 0 --> 1 dan 1 --> 0
    hasil_mutan = []
    if (mesinGacha(2)):     #Peluang Mutasi 2%
        gen = random.randint(0, len(anak))
        for i in range(0, len(anak)):
            if (i == gen):
                if (anak[i] == 1):
                    hasil_mutan.append(0)
                else:
                    hasil_mutan.append(1)
            else:
                hasil_mutan.append(anak[i])
    else:
        for i in range(0, len(anak)):
            hasil_mutan.append(anak[i])

    return hasil_mutan
```

3.5.Probabilitas

a. *Crossover*
   Peluang = 70%
b. Mutasi
   Peluang = 2%

3.6.Metode Pergantian Generasi (*Survivor Selection*)

Generational Model dengan elitism 2 individu dengan nilai fitness terbaik

```python
#6. Pergantian Generasi
def survivorSelection(populasi_fitness):
    #Ambil 2 individu dengan nilai fitness terbaik (elitism)
    maks = 0
    for i in range(0, len(populasi_fitness)):
        if (populasi_fitness[i] >= maks):
            maks = populasi_fitness[i]
            elit1 = i

    maks = 0
    for i in range(0, len(populasi_fitness)):
        if ((populasi_fitness[i] >= maks) and (i != elit1)):
            maks = populasi_fitness[i]
            elit2 = i

    return elit1, elit2
```

3.7.Kriteria Penghentian Evolusi

Max Iteration sampai generasi ke-100

```python
for i in range(1, 100): #Dihentikan sampai generasi ke-100
```

4. Proses yang Harus Dibangun

4.1.Dekode Kromosom

```python
#1. Decode Kromosom
def binaryDecodeToReal_X(indv):
    #Decode 8 kotak pertama dari biner menjadi nilai X berupa bil. real
    yang_dibagi = 0
    pembagi = 0
    for i in range (0, 8):
        dua_pangkat = 2 ** (-1 * (i+1))
        yang_dibagi += indv[i] * dua_pangkat
        pembagi += dua_pangkat

    return -1 + (((2 - (-1))/pembagi) * yang_dibagi)

def binaryDecodeToReal_Y(indv):
    #Decode 8 kotak terakhir dari biner menjadi nilai Y berupa bil. real
    yang_dibagi = 0
    pembagi = 0
    for i in range (0, 8):
        dua_pangkat = 2 ** (-1 * (i+1))
        yang_dibagi += indv[i+8] * dua_pangkat
        pembagi += dua_pangkat

    return -1 + (((1 - (-1))/pembagi) * yang_dibagi)
```

4.2.Perhitungan Fitness

```python
#2. Perhitungan Fitness
def hitungFitness(x,y):
    #Maksimasi
    return (math.cos(x) ** 2) * (math.sin(y) ** 2) + (x + y)
```

4.3.Pemilihan Orang Tua

Sama dengan 3.3.

4.4.Crossover

Sama dengan 3.4.a.

4.5.Mutasi

Sama dengan 3.4.b.

4.6.Pergantian Generasi

Sama dengan 3.6.

5. Nilai Parameter Genetic Algorithm yang Dianggap Optimal

5.1.Desain Kromosom

Representasi Biner. Semakin banyak gen yang dapat ditampung kromosom, maka akan semakin optimal.

5.2.Ukuran Populasi

Ukuran populasi yang baik paling sedikit 10

5.3.Pemilihan Orang Tua

Roulette Wheel, karena individu dengan nilai fitness tinggi memiliki peluang dipilih menjadi orang tua juga tinggi

5.4.Crossover

Peluang biasanya 60% - 80%

5.5.Mutasi

Peluang biasanya 1% - 5%

5.6. Survivor Selection

Generational dengan elitism karena mengamankan generasi dengan nilai fitness terbaik

5.7. Terminasi

Dengan maksimal generasi karena pasti menjamin program untuk berhenti

6. Kesimpulan

Dari semua parameter di atas, dapat disimpulkan bahwa pemilihan metode-metode genetic algorithm sangat dipengaruhi oleh pemilihan representasi individu (decode kromosom) di awal. Selain itu, pemilihan strategi juga bergantung dari fungsi optimal yang ingin dicari.

Hasil output program

```
PS D:\Kuliah Online\Kodingan\Python\Genetic Algorithm> python 18_if4305_1301180469.py
Kromosom terbaik adalah :
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

Dengan hasil decode x =  2.0  dan y =  1.0
Yang menghasilkan nilai maksimasi, yaitu  3.1226228726579794
```

7. Video

Link video :

http://bit.ly/video_1301180469

(Dibuka dengan e-mail SSO)