TUGAS PEMROGRAMAN

D
I
S
U
S
U
N

OLEH:

Muhammad Fadil Maulana Akbar - 1301204297
Muhammad Rafid Mustaghfirin -1301200417

DAFTAR ISI

BAB I 2

1.1 Latar Belakang 3

1.2 Tujuan Observasi 3

BAB II 3

2.1 Desain kromosom dan metode decode 4

2.2 Ukuran populasi 4

2.3 Metode Pemilihan orangtua 4

2.4 Metode operasi genetik (pindah silang dan mutasi) 5

2.5 Probabilitas operasi genetik (Pc dan Pm) 6

2.6 Metode pergantian generasi (seleksi survivor) 6

2.7 Kriteria penghentian evolusi (loop) 7

BAB III 8

3.1 Populasi dan Kromosom 8

3.2 Dekode Kromosom 8

3.3 Fitness 9

3.4 Seleksi Orang Tua 10

3.5 Crossover dan Mutasi 11

3.6 Seleksi Survivor 12

3.7 Output program 15

BAB IV 16

4.1 Kesimpulan 16

BAB I
PENDAHULUAN

1.1 Latar Belakang

Genetic Algorithm (GA) adalah bagian dari Evolutionary Algorithm yaitu suatu algoritma yang mencontoh proses evolusi alami dimana konsep utamanya adalah individu-individu yang paling unggul akan bertahan hidup, sedangkan individu-individu yang lemah akan punah. Keunggulan individu-individu ini diuji melalui suatu fungsi yang dikenal sebagai fitness function. Fitness dalam GA didefinisikan sebagai gambaran kelayakan suatu solusi terhadap suatu permasalahan. Fitness Function akan menghasilkan suatu nilai fitness value yang akan menjadi referensi untuk proses GA selanjutnya.

1.2 Tujuan Observasi

Tujuan dari laporan obervasi ini adalah untuk menemukan nilai minimum dari fungsi

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

Dengan batasan $-5 \leq x \leq 5$ dan $-5 \leq y \leq 5$ selain itu laporan ini juga bertujuan untuk mencari kromosom terbaik dan nilai hasil decode kromosom terbaik.

BAB II
ANALISIS DAN DESAIN

2.1 Desain kromosom dan metode decode

Pembentukan genetika algoritma diawali dengan pembentukan kromosom. Kromosom di generesi selanjutnya akan mengalami evolusi lalu menghasilkan kromosom - kromosom baru dari generasi sebelumnya.

Terdapat tiga cara untuk melakukan decode kromosom

- Representasi biner
- Representasi integer
- Representasi bilangan real

Dalam laporan observasi ini kita memakai reprensentasi biner dan panjang kromosom 8 dan 16.

2.2 Ukuran populasi

Ukuran populasi merupakan parameter penting, karena semakin besar ukuran populasi maka semakin lama waktu yang dibutuhkan untuk komputasi sedangkan jika ukurannya terlalu kecil maka dapat menyebabkan konvergensi dini.

Dalam laporan observasi ini ukuran populasi di rentang antara 100 - 150 populasi.

2.3 Metode Pemilihan orangtua

Proses untuk pemilihan orang tua dari kromosom yang ada, terdapat 3 cara dalam pemilihan orang tua

- Roulette Wheel

  Roulette wheel selection adalah metode seleksi yang memilih orang tua berdasarkan nilai kecocokannya (fitness). Kromosom yang lebih baik memiliki persentasi dipilih yang lebih besar.
- Stochastic Universal Sampling (SUS)

  Stochastic universal sampling hampir sama dengan metode seleksi roulette wheel yang dimana satu individu diletakkan dalam segmen segaris sehingga memiliki nilai fitness yang sama perbedaannya pada stochastic ditambahkan pointer pada setiap individu yang ingin diseleksi.
- Tournament Selection

  Tournament Selection adalah salah satu metode seleksi dalam algoritma genetika. Dalam seleksi turnamen, n individu dipilih secara acak dari yang lebih besar nilai fitness dalam populasi, dan individu-individu yang dipilih bersaing satu sama lain

Dalam observasi ini metode pemilihan orangtua yang kita pakai adalah Tournament Selection dan Roulette Wheel.

2.4 Metode operasi genetik (pindah silang dan mutasi)

1. Crossover

   Terdapat 2 cara yaitu ;

- Single point crossover :
  Dengan acak memilih 1 point
- Two point crossover :
  Dengan acak memilih 2 point

Dalam observasi ini kita memakai Two point crossover.

2. Mutation

Mutasi dalam operasi genetik yaitu operator yang dapat mengubah gen - gen dalam kromosom.

2.5 Probabilitas operasi genetik (Pc dan Pm)

1. Pc (Probabilitas Crossover)
   Banyaknya kemungkinan suatu kromosom untuk mengalami proses crossover.
   Dalam observasi ini kita memakai Pc dengan nilai 0,8 .

2. Pm (Probabilitas Mutation)
   Banyaknya kemungkinan suatu kromosom untuk mengalami proses mutation
   Dalam observasi ini kita memakai Pm dengan nilai 0,3 .

2.6 Metode pergantian generasi (seleksi survivor)

Proses ini bertujuan untuk memilih N kromosom dari gabungan antara populasi sebelumnya dan offspring yang dihasilkan. Lalu, N kromosom yang terpilih akan digunakan sebagai populasi untuk perulangan atau iterasi berikutnya

Metode pergantian generasi ada 2 yaitu ;

1. Generational replacement
   Dalam algoritma, berevolusi dikenal dengan skema penggantian populasi yang disebut *generational replacement* yang berarti semua individu dari suatu generasi digantikan sekaligus oleh jumlah individu baru hasil pindah silang dan mutasi.
2. Steady state
   Metode ini berisi menghapus hanya satu keturunan membentuk generasi dan mereproduksi hanya satu keturunan. Jadi untuk setiap generasi hanya 1 keturunan yang dihasilkan.

Dalam laporan ini memakai Generational Replacement sebagai metode pergantian generasinya.

2.7 Kriteria penghentian evolusi (loop)

Kriteria yang digunakan untuk menghentikan proses evolusi dengan memberikan batas regenerasi pada jumlah tertentu. Batasan generasi yang dipakai pada laporan observasi ini adalah 100 dan 200 generasi.

BAB III
IMPLEMENTASI

3.1 Populasi dan Kromosom

```
def generateChromosom(chr_size):
    result = []
    for _ in range(chr_size):
        result.append(random.randint(0, 1))
    return result
```

Fungsi ***generateChromosome*** digunakan untuk membuat sebuah array berisi kromosom dengan angka 0 dan 1 secara acak sepanjang chr_size.

```
def generatePopulation(pop_size, chr_size):
    pop = []
    for _ in range(pop_size):
        pop.append(generateChromosom(chr_size))
    return pop
```

Fungsi dari ***generatePopulation*** berfungsi untuk membuat array sebuah populasi dari berbagai kromosom. Panjang dari sebuah populasi ditentukan oleh nilai pop_size.

3.2 Dekode Kromosom

```python
def decodeChromosome(chromosome, chr_size):
    xMin, xMax = (-5, 5)
    yMin, yMax = (-5, 5)

    N, x, y = 0, 0, 0
    n = (chr_size) // 2

    for i in range(0, n):
        N += 2**-(i+1)
    for i in range(0, n):
        x += chromosome[i] * 2 ** -(i+1)
        y += chromosome[n + i] * 2 ** -(i+1)
    x = xMin + ((xMax - xMin) / N) * x
    y = yMin + ((yMax - yMin) / N) * y

    return [x, y]
```

Nilai x dan y yang akan digunakan untuk menghitung fungsi didapat dari mendekode nilai kromosom menggunakan fungsi ***decodeChromosome*** dengan kromosom berbentuk biner. Pada fungsi ini kelompok kami menggunakan looping untuk perhitungan agar mempermudah jika kita ingin mengubah panjang kromosom.

3.3 Fitness

```python
    # menghitung nilai dari fungsi
def Objectivefunction(x, y):
    return ((math.cos(x)+math.sin(y))**2) / (x**2 + y**2)

    # menghitung nilai minimisasi dari fungsi
def function(x, y):
    return 1 /  (0.01 + ((math.cos(x)+math.sin(y))**2) / (x**2 + y**2))

    # fungsi untuk memasukkan semua nilai fitness ke dalam array
def fitness(population, chr_size):
    result = []
    for i in population:
        # pemanggilan fungsi function dan decode chromosome
        result.append(function(*decodeChromosome(i, chr_size)))
    return result
```

Untuk menghitung nilai fitness pertama kita membutuhkan suatu fungsi untuk mengembalikan nilai fungsi dari soal. Untuk memasukkan nilai fungsi tersebut kelompok kami membuat suatu fungsi bernama fitness yang akan mereturn sebuah array yang berisi semua kemungkinan solusi.

3.4 Seleksi Orang Tua

a. Tournament

```python
def tournament(population, pop_size, tour_size, chr_size):
    result = []
    for _ in range(tour_size):
        temp = population[random.randint(0, pop_size - 1)]
        if result == [] or function(decodeChromosome(temp, chr_size)[0],
        decodeChromosome(temp, chr_size)[0]) > function(decodeChromosome(result, chr_size)[0], decodeChromosome(result, chr_size)[0]):
            result = temp
    return result
```

PUntuk metode seleksi pertama kami menggunakan metode tournament cara kerjanya yaitu :

1. Membuat sebuah array kosong dengan nama “result”.
2. Menentukan banyaknya kromosom yang akan dipilih.
3. Memilih kromosom secara acak lalu hitung nilai fitnessnya.
4. Kemudian bandingkan nilai fitness dari result dengan nilai fitness pada langkah ke - 3.
5. Jika nilai fitness result lebih besar maka tukar nilai result dengan nilai pada langkah ke - 3.

b. Roulette wheel

```
def RouletteWheel(population ,pop_size, fitness):
    indv = 0
    val = random.uniform(0, 1)
    totalfitness = sum(fitness)
    for i in range(pop_size):
        if (fitness[i] / totalfitness) > val:
            indv = i
            break
        i += 1
    return population[indv]
```

Metode seleksi yang ke - 2 menggunakan metode Roulette wheel, cara kerjanya yaitu :

1. Membuat suatu variabel untuk menampung indeks dengan nama "indv".
2. Lalu membuat suatu variabel dengan nama "val" untuk menampung bilangan real antara 0 dan 1.
3. Kemudian jumlahkan semua fitness dan masukkan variabel total fitness.
4. Kemudian buatlah perulangan sebanyak jumlah populasi.
5. Jika index nilai fitness dibagi dengan total fitness lebih besar dari "val" maka index nilai fitness tersebut akan dimasukkan ke nilai "indv".

3.5 Crossover dan Mutasi

a. Crossover(Two Point Crossover).

```
def crossover(parentA, parentB, pc, chr_size):
    val = random.uniform(0, 1)
    if val < pc:
        pindah = random.randint(0, chr_size-1)
        for i in range(pindah):
            parentA[i], parentB[i] = parentB[i], parentA[i]
    return [parentA, parentB]
```

Kami menggunakan metode two point crossover untuk mengcrossover nilai dari ***parentA*** dan ***parentB***.

Cara kerjanya yaitu :

1. Membuat suatu variabel bernama “val” yang berisi bilangan real antara 0 dan 1.
2. Jika nilai “val” kurang dari probabilitas crossover maka crossover akan dilakukan secara acak.

b. Mutasi

```
def mutation(offsprings, pm, chr_size):
    val = random.uniform(0, 1)
    if val < pm:
        offsprings[0][random.randint(0, chr_size - 1)] = random.randint(0,1)
        offsprings[1][random.randint(0, chr_size - 1)] = random.randint(0,1)
    return offsprings
```

Fungsi mutation digunakan untuk mengubah nilai kromosom dengan probabilitas yang sangat kecil.

3.6 Seleksi Survivor

```python
def elitism(fit):
    idx1, idx2 = 0, 0
    for i in range(1, len(fit)):
        if fit[i] < fit[idx1]:
            idx2 = idx1
            idx1 = i
    return [idx1, idx2]
```

Kelompok kami menggunakan generational replacement sebagai metode seleksi survivor, oleh karena itu harus ditentukan lebih dulu nilai elitisme dari populasi tersebut. Fungsi elitism akan mengembalikan dua index dari array fitness yang memiliki dua nilai terkecil.

```python
def generationalReplacement():
    #initialization
    chr_size, pop_size, pc, pm, generation = (16, 150, 0.8, 0.2, 1000)
    tour_size = 5 # Khusus tournamentSelection
    population = generatePopulation(pop_size, chr_size)
    for _ in range(generation):
        fit = fitness(population, chr_size)
        newPopulation = []
        elite1, elite2 = elitism(fit)
        newPopulation.append(population[elite1])
        newPopulation.append(population[elite2])
        for _ in range(0, pop_size - 2, 2):
                    # Menggunakan Tournament Selection
            # parentA = tournament(population, pop_size, tour_size, chr_size)
            # parentB = tournament(population, pop_size, tour_size, chr_size)
            # while(parentA == parentB):
            #     parentB = tournament(population, pop_size, tour_size, chr_size)
                    # Menggunakan Roulette Wheel Selection
            parentA = RouletteWheel(population, pop_size, fit)
            parentB = RouletteWheel(population, pop_size, fit)
            while(parentA == parentB):
                parentB = RouletteWheel(population, pop_size, fit)
            offsprings = crossover(parentA[:], parentB[:], pc, chr_size)
            offsprings = mutation(offsprings, pm, chr_size)
            newPopulation.extend(offsprings)
    population = newPopulation
    printNilaiMin(population, chr_size, fit)
```

Setelah menentukan nilai elitisme kita bisa melakukan pergantian generasi dengan fungsi ***generationalReplacement***. Didalam fungsi ini akan dilakukan perulangan sebanyak generasi yang sudah ditentukan. Perulangan ini berisi inisialisasi dari program tersebut, seleksi orang tua, crossover dan mutasi.

Generasi terakhir akan berisi kromosom terbaik dan nilai fitness terbaik.

```
def printNilaiMin(population, chr_size, fit):
    idx = fit.index(min(fit))
    decode = decodeChromosome(population[idx], chr_size)

    print("===============Hasil Genetic Algorithm===============")
    print("Kromosom terbaik\t: ", population[idx])
    print("Nilai fitness\t\t: ", fit[idx])
    print("Nilai X\t\t\t: ", decode[0])
    print("Nilai y\t\t\t: ", decode[1])
```

Pada fungsi ***printNilaiMin*** akan mengoutputkan kromosom terbaik, niali fitness terbaik, nilai x dan y yang sudah didecode dari kromosom terbaik.

3.7 Output program

a. Test menggunakan Roulette Wheel

```
# ======================Roulette Wheel Selection======================

# Test dengan chr_size = 8, pop_size = 100, generation = 100:
# ===============Hasil Genetic Algorithm===============
# Kromosom terbaik        :  [1, 1, 0, 1, 1, 0, 0, 1]
# Nilai fitness           :  99.97510572941917
# Nilai fungsi            :  3.92673529654062 54e-05
# Nilai X                 :  3.666666666666666
# Nilai y                 :  1.0
# Process finished --- 1.950829267501831 seconds ---

# Test dengan chr_size = 10, pop_size = 150, generation = 200:
# ===============Hasil Genetic Algorithm===============
# Kromosom terbaik        :  [0, 0, 0, 0, 0, 1, 1, 0, 0, 0]
# Nilai fitness           :  99.998222567881
# Nilai fungsi            :  0.013918736270974261
# Nilai X                 :  -5.0
# Nilai y                 :  2.741935483870968
# Process finished --- 17.95911931991577 seconds ---
```

b. Test menggunakan Tournament

```
# ========================Tournament Selection========================

# Test dengan chr_size = 8, pop_size = 100, tour_size = 4, generation = 100:
# ===============Hasil Genetic Algorithm===============
# Kromosom terbaik        :  [1, 0, 1, 1, 0, 0, 1, 0]
# Nilai fitness           :  99.92237109023498
# Nilai fungsi            :  0.0019007515009992928
# Nilai X                 :  2.333333333333333
# Nilai y                 :  -3.666666666666667
# Process finished --- 0.5887758731842041 seconds ---


# Test dengan chr_size = 10, pop_size = 150, tour_size = 5, generation = 200:
# ===============Hasil Genetic Algorithm===============
# Kromosom terbaik        :  [0, 1, 1, 0, 0, 0, 0, 0, 1, 0]
# Nilai fitness           :  99.93997043687797
# Nilai fungsi            :  0.09196410381771221
# Nilai X                 :  -1.129032258064516
# Nilai y                 :  -4.354838709677419
# Process finished --- 2.4650518894195557 seconds ----
```

BAB IV
PENUTUP

4.1 Kesimpulan

Berdasarkan laporan pada program genetic algorithm, program tersebut akan terus berevolusi sesuai dengan parameter yang sudah ditentukan hingga ditemukan nilai evolusi terbaik.

Dengan menggunakan dua metode seleksi orang tua yaitu metode Roulette wheel dan metode Tournament dengan parameter yang berbeda dapat dilihat bahwa hasil dari kedua seleksi tersebut menghasilkan nilai dan waktu komputasi yang berbeda.

Dari laporan yang telah kami kerjakan dapat disimpulkan bahwa seleksi orang tua dengan metode Tournament memiliki waktu komputasi lebih cepat dibanding metode Roulette wheel, tetapi metode Roulette wheel memiliki nilai fitness terbaik yang lebih akurat dibanding metode Tournament

Source Code: https://github.com/fadilmr/TuPro1-GeneticAlgorithm