LAPORAN TUGAS PEMROGRAMAN GENETIC ALGORITHM
PENGANTAR KECERDASAN BUATAN

Disusun oleh :
1. Muhammad Nur Iqbal Wariesky (1301204044)
2. Muhammad Thoriq Akhdan (1301204031)
Program Studi S1 Informatika
Fakultas Informatika
2021

ANALISIS MASALAH

A. Desain Kromosom dan Metode Pendekodean

Diberikan sebuah fungsi $h(x,y) = (\cos x + \sin y)^2 / x^2 + y^2$. Kemudian tentukan nilai minimum dari fungsi tersebut menggunakan algoritma Genetic Algorithm dengan batasan $-5 \leq x \leq 5$ dan $-5 \leq y \leq 5$. Pada kasus diatas ini kami menerapkan representasi integer untuk merepresentasikan individu dalam tugas pemograman ini. Berikut adalah algoritma fungsi kromosom dan nilai fungsi :

```
batas_x = [-5, 5]
batas_y = [-5, 5]
```

```
def fungsi(x, y):  # Menggunakan rumus fungsi dari soal
    return ((math.cos(x) + math.sin(y))**2) / (x**2 + y**2)
```

```
# Populasi


def kromosom(gen):  # Membuat kromosom dengan representasi individual menggunakan nilai integer random
    kromosomtable = []
    for i in range(gen):
        kromosomtable.append(random.randint(0, 9))
    return kromosomtable
```

B. Ukuran Populasi

Ukuran populasi disesuaikan dengan jumlah yang dimasukkan oleh user. Berikut adalah algoritma fungsi populasi :

```
def populasi(pop, gen):  # Menambahkan kromosom ke tiap populasi
    populasitables = []
    for i in range(pop):
        populasitables.append(kromosom(gen))
    return populasitables
```

C. Fungsi Fitness

Fungsi fitness yang kita gunakan adalah fungsi minimasi dengan rumus 1/(h+a), karena kita akan mencari nilai minimal dari sebuah fungsi h(x,y). Berikut adalah algoritma fungsi nilai fitness dan hitung fitness :

```
def fit(h):  # Nilai minimum maka digunakan 1/(h+a), a = nilai kecil agar pembagian dengan 0 tidak terjadi
    return 1//(h+0.01)
```

```
def fitness(populasi):  # Meenghitung fitness dari tiap populasi
    fitness_populasi = []
    for i in range(len(populasi)):
        x, y = split(populasi[i])
        gamet_x = dekode(x, batas_x)
        gamet_y = dekode(y, batas_y)
        f = fungsi(gamet_x, gamet_y)
        fitnes = fit(f)
        fitness_populasi.append(fitnes)
    return fitness_populasi
```

```
def split(krom):  # Membelah kromosom menjadi 2
    return (krom[:len(krom)//2], krom[len(krom)//2:])

# Dekode kromosom


def dekode(gamet, batas):
    pengali = 0
    pembagi = 0

    # Menggunakan rumus representasi integer
    for i in range(len(gamet)):
        n = gamet[i]
        # Dengan membagi perhitungan menjadi pengali dan pembagi untuk mempermudah perhitungan
        pengali = pengali + (n*(10**-(i+1)))
        pembagi = pembagi + (9*(10**-(i+1)))
    # Menggabungkan kembali pengali dan pembagi yang sudah dihitung dan memasukkan kedalam rumus untuk menentukan fenotype
    x = batas[0] + (((batas[1]-batas[0]) / pembagi) * pengali)
    return x
```

D. Pemilihan Orangtua

Pada parent selection kelompok kami menggunakan tournament selection. Cara kerja tournament selection yaitu memilih sejumlah x kromosom secara acak dari sebuah populasi berjumlah N kromosom. Kemudian kromosom tersebut memilih satu dari kromosom terbaik untuk dijadikan parent. Hal tersebut dilakukan secara terus menerus hingga dihasilkan N kromosom sebagai parent. Berikut adalah algoritma fungsi tournament selection :

```python
# Pemilihan orang tua


def select(populasi):  # Menggunakan seleksi orang tua Turnamen
    calon = []
    fitness_calon = []
    for i in range(5):  # turnamen berukuran k = 5
        # dipilih kromosom secara acak dari populasi
        n = random.randint(0, len(populasi)-1)
        calon.append(populasi[n])
    fitness_calon = fitness(calon)
    parent_1 = max(fitness_calon)
    idx_parent_1 = fitness_calon.index(parent_1)
    fitness_calon.remove(parent_1)
    parent_2 = max(fitness_calon)
    idx_parent_2 = fitness_calon.index(parent_2)
    return (calon[idx_parent_1], calon[idx_parent_2])
```

E. Crossover dan Mutasi

Pada tugas kali ini kelompok kami menggunakan rekombinasi satu titik yaitu dengan cara memilih satu titik pada kromosom. Cara kerja dari cara ini yaitu memilih titik potong dengan posisi X, lalu anak pertama akan mewarisi gen parent pertama dari titik awal hingga titik X, dan gen parent kedua dari titik X hingga akhir. Berkebalikan dengan anak kedua, anak kedua akan mewarisi gen parent kedua dari titik awal hingga X dan gen parent pertama dari titik X hingga titik akhir

Pada tugas ini juga kami memilih mutasi dengan memilih nilai secara acak dengan cara memilih satu gen secara acak dalam kromosom, lalu nilainya akan diganti dengan nilai yang baru secara acak dalam interval [0,9]. Proses terjadinya mutasi dipengaruhi dari probabilitas terjadinya mutasi. Berikut adalah algooritma dari rekombinasi dan mutasi :

```python
def mutation(krom, prob):  # mutasi terjadi dengan mengganti nilai secara acak
    if random.random() <= prob:
        # posisi mutasi dipilih dengan acak
        n = random.randint(0, len(krom)-1)
        print("titik mutasi:", n)
        # nilai mutasi
        m = random.randint(0, 9)
        krom[n] = m
    return krom
```

```python
def cross(gen1, gen2, prob):  # menggunakan rekombinasi satu titik
    anak1 = []
    anak2 = []
    # menentukan titik potong secara random
    T = random.randint(1, len(gen1)-1)
    if random.random() <= prob:
        print("titik potong:", T)
        # Dibuat anak 1 dengan gen 1 - T dari gen orang tua 1
        anak1[:T] = gen1[:T]
        # Dibuat anak 1 dengan gen T+1 - G dari gen orang tua 2
        anak1[T:] = gen2[T:]
        anak2[:T] = gen2[:T]
        anak2[T:] = gen1[T:]
    else:
        anak1 = gen1
        anak2 = gen2
    return (anak1, anak2)
```

F. Probabilitas Operasi Genetik

Semakin tinggi angka probabilitas mutasi, maka semakin besar kemungkinan sebuah gen dalam kromosom terjadi mutasi. Sebaliknya semakin kecil angka probabilitas mutasi, maka semakin kecil kemungkinan sebuah gen dalam kromosom untuk bermutasi.

```python
mut_prob = 0.1
```

G. Seleksi Survivor

Pada proses seleksi survivor kita menggunakan metode Generational Model dimana suatu populasi berukuran N kromosom akan diganti dengan N kromosom baru pada generasi berikutnya. Namun untuk menjaga kromosom terbaik sehingga bisa menghasilkan solusi yang optimal kita menggunakan elitisme yaitu memilih sebuah kromosom terbaik dari generasi sebelumnya dan memindahkannya ke generasi selanjutnya. Dengan cara tersebut minimal kita akan mempunyai satu solusi terbaik dari generasi sebelumnya. Berikut algoritma dari proses seleksi survivor :

```python
# Pergantian generasi


def elitism(populasi):  # Populasi baru dipilih dengan nilai fitness terbaik dan menggantikan populasi dari generasi sebelumnya
    populasibaru = []
    fitness_populasi = fitness(populasi)
    best = max(fitness_populasi)
    idx_best = fitness_populasi.index(best)
    populasibaru.append(populasi[idx_best])
    return populasibaru
```

H. Evolusi

1. Model populasi : Generational Model
2. Seleksi orangtua : Tournament Selection
3. Rekombinasi : Satu Titik Potong
4. Mutasi : Memilih Nilai Secara Acak
5. Seleksi Survivor : Elitisme

FAKTOR YANG MEMPENGARUHI HASIL

Beberapa faktor yang dapat mempengaruhi hasil akhir yaitu metode pemilihan orang tua, rekombinasi, mutasi, seleksi survivor. Ada juga faktor lain yaitu panjang gen yang dibuat, banyaknya kromosom terhadap suatu populasi, angka probabilitas rekombinasi, angka probabilitas mutasi, dan banyaknya generasi yang dijalankan.

HASIL DAN KESIMPULAN

Dari hasil percobaan dengan panjang gen 6, banyak populasi 4, dan sebanyak 3 generasi didapatkan
----- generasi ke- 1 -----
[[4, 7, 8, 1, 2, 5], [6, 4, 5, 3, 8, 0], [7, 3, 4, 8, 1, 2], [5, 0, 3, 1, 8, 6]]

Dan didapatkan solusi [7, 0, 3, 8, 1, 2]
nilai x: 2.037037037037038
nilai y: 3.128128128128129
nilai fungsi: 0.013646122626305178
dengan nilai fitness 42.0

```
----- generasi  3 -----
[[7, 0, 3, 8, 1, 2], [7, 0, 3, 8, 8, 6], [7, 0, 3, 9, 1, 2], [7, 0, 8, 8,
1, 2]]

maka didapatkan solusi [7, 0, 3, 8, 1, 2]
dengan nilai fitness 42.0
nilai x: 2.037037037037038
nilai y: 3.128128128128129
nilai fungsi: 0.013646122626305178
```

Kesimpulan yang dapat diperoleh dari tugas pemrograman genetic algorithm adalah dengan menggunakan metode pemilihan orang tua, metode mutasi, metode seleksi survivor yang tepat dan panjang gen dan jumlah kromosom yang banyak dan angka probabilitas mutase yang semakin kecil, maka semakin besar kemungkinan untuk kita mendapatkan hasil yang optimal.

Terlampir :
Link collab :