LAPORAN TUGAS PEMROGRAMAN GENETIC ALGORITHM
PENGANTAR KECERDASAN BUATAN

Disusun oleh:
Alvian Daniswara Adhipramana Putra (1301200059)
Dhanu Satrio Darjanto (1301204174)

Program Studi S1 Informatika
Fakultas Informatika
2022

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa atas berkat dan rahmatnya penyusunan laporan tugas “Searching” dapat deselesaikan dengan baik. Penulis menyadari bahwa dalam proses penulisan laporan ini banyak mengalami kendala, namun kendala-kendala tersebut dapat diatasi.

Laporan tugas ini dibuat dalam rangka memenuhi tugas yang diberikan oleh bapak Agus Hartoyo, ST dan memahami algoritma dan pemrograman khusunya tentang Genetic Algorithm (GA).

Penulis menyadari betul sepenuhnya bahwa laporan ini masih jauh dari sempurna. Oleh karena itu, dengan segala kerendahan hati Penulis berharap saran dan kritik demi perbaikan lebih lanjut. Penulis berharap semoga laporan ini dapat memberikan manfaat bagi pembaca

BAB I
PENDAHULUAN

1.1. Latar Belakang

Kecerdasan buatan atau artificial intelligence merupakan bagian dari ilmu komputer yang membuat agar mesin (komputer) dapat melakukan pekerjaan seperti dan sebaik yang dilakukan oleh manusia. Sistem cerdas (intelligent system) adalah sistem yang dibangun dengan menggunakan teknik-teknik artificial intelligence.

Dalam penggunaan AI terdapat penerapan beberapa algoritma, salah satunya adalah algoritma genetika. Menurut Goldberg (1989), algoritma genetika merupakan sebuah algoritma pencarian yang berdasarkan dengan mekanisme seleksi alam dan genetika alam. Algoritma genetika digunakan dan dikembangkan sebagai algoritma yang khusus dalam mencari suatu solusi yang optimal terhadap masalah yang bersesuaian dengan proses evolusi seperti contohnya adalah pencarian nilai maksimum pada suatu fungsi.

1.2. Tujuan

Tujuan dari analisa permasalahan tugas pemrograman ini adalah untuk menentukan nilai minimum dari suatu fungsi yang diberikan.

1.3. Rumusan Masalah

bagaimana cara mengimplementasikannya ke dalam suatu program komputer untuk mencari nilai x dan y sehingga diperoleh nilai minimum dari fungsi

BAB II
LANDASAN TEORI

2.1 Genetika Algorithm

Algoritma Genetika dikembangkan pertama kali oleh John Holand dari New York, Amerika Serikat yang dipublikasikan dalam bukunya yang berjudul “Adaption in Natural and Artificial Systems” tahun 1975. Algoritma Genetika merupakan Teknik untuk menemukan solusi optimal dari permasalahan yang mempunyai banyak solusi. Teknik ini akan melakukan pencarian dari beberapa solusi yang diperoleh sampai mendapatkan solusi terbaik sesuai dengan kriteria yang telah ditentukan atau yang disebut sebagai fungsi fitness.

Algoritma ini masuk dalam kelompok algoritma evolusioner dengan menggunakan pendekatan evolusi Darwin di bidang Biologi seperti pewarisan sifat, seleksi alam, mutasi gen dan kombinasi (crossover). Karena merupakan Teknik pencarian optimal dalam bidang ilmu komputer, maka algoritma ini juga termasuk dalam kelompok algoritma metaheuristik.

2.2 Python

Bahasa pemrograman yang dikembangkan oleh Guido van Rossum ini terus mengalami pembaruan hingga saat ini. Nama Python sendiri diambil dari program televisi favoritnya yang bernama “Monty Python Flying Circus”. Python adalah bahasa pemrograman yang populer digunakan di seluruh dunia untuk mengembangkan situs web, algoritma dan menyederhanakan proses otomatisasi.

Melalui bahasa pemrograman Python, setiap program akan menjadi lebih ringkas jika dibandingkan bahasa pemrograman lain. Tak hanya itu, Python bertujuan untuk menghasilkan kode yang lebih jelas dan lebih logis untuk berbagai keperluan. Proses pengkodean Python sangat sederhana sehingga memberikan keleluasaan bagi developer untuk mengembangkan fitur baru dari suatu situs atau aplikasi.

BAB III
ANALISIS DAN PERANCANGAN SISTEM

3.1 Operator Evolusi

- Representasi : Integer
  Pada representasi kami menggunakan model integer dengan x (-5,5) dan y (-5,5).

- Model Populasi : Generational
  Pada model populasi kami menggunakan model generational dimana program akan membuat generasi baru. Ukuran populasi pada program ini sesuai dengan inputan yang diminta atau bebas, dengan jumlah individu sebanyak 3 individu

- Seleksi orang tua : Roulette Wheel
  Pada seleksi orang tua menggunakan metode roulette wheel yaitu semakin besar nilai fitness suatu individu maka semakin besar peluang untuk terpilih sebagai orang tua.

- Rekombinasi : satu titik potong
  Pada rekombinasi kami menggunakan metode satu titik potong dengan probability crossover adalah 0.5.

- Mutasi : Nilai secara acak
  Pada mutasi kami menggunakan metode mutasi secara acak dimana nilai diganti dengan gen baru yang dibangkitkan dengan metode random.radint. Probability mutasi adalah 0.5.

- Seleksi : Steady State model (elitisme)
  Pada metode seleksi kami menggunakan steady state model dengan elitisme dimana pergantian kromosom ditentukan berdasarkan nilai minimum individu terbaik dari 3 individu.

3.2 Coding (Python)

- Library

```
import random
import math
```

- Dekode kromosom

```
def generate_population(size, batasX, batasY):
    batasBawahX, batasAtasX = batasX
    batasBawahY, batasAtasY = batasY

    population = []
    for i in range(size):
        individu = {
            #dekode kromosomnya
            "x": random.randint(batasBawahX, batasAtasX),
            "y": random.randint(batasBawahY, batasAtasY),
        }
        population.append(individu)

    return population
```

- Perhitungan Fitness

```
def calculate_fitness(individu):
    #rumus fitness
    x = individu["x"]
    y = individu["y"]
    return ((math.cos(x) + math.sin(y))**2)/((x**2 + y**2))
```

- Elitism

```
def sort_by_fitness(population):
    return sorted(population, key=calculate_fitness, reverse=True) # mengurutkan berdasarkan nilai minimal
```

- Pemilihan Orangtua

```
def roulette_wheel(sorted_population, fitness_sum):
    offset = 0
    normalized_fitness_sum = fitness_sum

    lowest_fitness = calculate_fitness(sorted_population[0])
    if lowest_fitness < 0:
        offset = -lowest_fitness
        normalized_fitness_sum += offset * len(sorted_population)

    accumulation = 0
    draw = random.randint(0, 1)
    for individu in sorted_population:
```

```
        fitness = calculate_fitness(individu) + offset
        probability = fitness / fitness_sum
        accumulation += probability

        if draw <= accumulation:
            return individu
```

- Crossover (pindah silang)

```
def crossover(individu_a, individu_b):
    xa = individu_a["x"]
    ya = individu_a["y"]
    xb = individu_b["x"]
    yb = individu_b["y"]

    return {"x": xb, "y": ya}
```

- Mutasi

```
def mutation(individu):

    next_x = random.randint(batasBawahX, batasAtasX)
    next_y = random.randint(batasBawahY, batasAtasY)

    return {"x": next_x, "y": next_y}
```

- Pergantian Generasi

```
def generation_new(population_before):
    generation_after = []
    sorted_by_fitness_population = sort_by_fitness(population_before)
    #elitism dipotong disini
    population_size = len(population_before)
    fitness_sum = sum(calculate_fitness(individu) for individu in population)

    i = 0
    while i <(population_size):
        ortu_1 = roulette_wheel(sorted_by_fitness_population, fitness_sum)
        ortu_2 = roulette_wheel(sorted_by_fitness_population, fitness_sum)

        if random.randint(-5,5) < 0.5 :
            individu = crossover(ortu_1, ortu_2)
            individu = mutation(individu)
            generation_after.append(individu)
        else :
```

```
            generation_after.append(ortu_1)
        i+=1
    return generation_after
```

- Program Utama

```
batasBawahX, batasAtasX = -5,5
batasBawahY, batasAtasY = -5,5

generations = int(input("generasi yang dibutuhkan : ", ))
population = generate_population(size=3, batasX=(-5,5), batasY=(-5,5))

i = 1
while True:
    print("generasi ke :", i)
    for individu in population:
        print(individu)
        print("Fitnessnya :,", calculate_fitness(individu))
        fitness_sum = sum(calculate_fitness(individu) for individu in population)
        probability = calculate_fitness(individu)/fitness_sum
        print("probabilitas terpilih :", probability)
        print("-----------------------------------------")

    if i == generations:
        break

    print("=====================================================")
    print("individu terminimum di gen ini: ", sort_by_fitness(population)[-1])
    print("=====================================================")
    print(" ")
    temp = sort_by_fitness(population)[-1]

    i += 1
    population = generation_new(population)
    population.append(temp)

individu_terbaik = sort_by_fitness(population)[-1]
print("=====================================================")
print ("Individu terminimum yang didapatkan : ")
print(individu_terbaik)
print("Fitnessnya: ", calculate_fitness(individu_terbaik))
print("Probabilitas individu terminimum :", probability)
print("=====================================================")
```

3.3 Output

- Generasi ke - 1

```
generasi yang dibutuhkan : 3
generasi ke : 1
{'x': 3, 'y': 2}
Fitnessnya :, 0.0005008995604579987
probabilitas terpilih : 0.0028110457800735646
--------------------------------------------------
{'x': -3, 'y': 0}
Fitnessnya :, 0.10889834925835366
probabilitas terpilih : 0.6111369809543701
--------------------------------------------------
{'x': -2, 'y': 4}
Fitnessnya :, 0.06879050675496282
probabilitas terpilih : 0.3860519732655564
--------------------------------------------------
==================================================
individu terminimum di gen ini:  {'x': 3, 'y': 2}
==================================================
```

- Generasi ke - 2

```
generasi ke : 2
{'x': 2, 'y': 0}
Fitnessnya :, 0.043294547392048514
probabilitas terpilih : 0.927751106798232
--------------------------------------------------
{'x': 5, 'y': -3}
Fitnessnya :, 0.0005975962452614905
probabilitas terpilih : 0.012805782976302475
--------------------------------------------------
{'x': 4, 'y': -5}
Fitnessnya :, 0.0022730799410796677
probabilitas terpilih : 0.04870942320013337
--------------------------------------------------
{'x': 3, 'y': 2}
Fitnessnya :, 0.0005008995604579987
probabilitas terpilih : 0.010733687025332084
--------------------------------------------------
==================================================
individu terminimum di gen ini:  {'x': 3, 'y': 2}
==================================================
```

- Generasi ke - 3

```
generasi ke : 3
{'x': 0, 'y': 4}
Fitnessnya :, 0.0036965641430281426
probabilitas terpilih : 0.009441299397134357
------------------------------------------------
{'x': 5, 'y': 1}
Fitnessnya :, 0.048689409647859505
probabilitas terpilih : 0.12435636882486092
------------------------------------------------
{'x': -3, 'y': -1}
Fitnessnya :, 0.3354258483732364
probabilitas terpilih : 0.8567025317286391
------------------------------------------------
{'x': -5, 'y': 0}
Fitnessnya :, 0.0032185694184709507
probabilitas terpilih : 0.008220465365806454
------------------------------------------------
{'x': 3, 'y': 2}
Fitnessnya :, 0.0005008995604579987
probabilitas terpilih : 0.0012793346835591386
------------------------------------------------
================================================
Individu terminimum yang didapatkan :
{'x': 3, 'y': 2}
Fitnessnya:  0.0005008995604579987
Probabilitas individu terminimum : 0.0012793346835591386
================================================
```

3.4 Hasil Analisa

Dari output dapat dilihat bahwa hasil telah mencapai tujuan yaitu mencari nilai minimum sesuai dengan fungsi yang telah diberikan. Dimana nilai minimum berdasarkan hasil fitnes nya yang memenuhi kriteria kromosom x dan y berupa mendekati batas atas dari x dan mendekati batas atas dari y. Setelah kami mencoba menjalankan program, kami mendapatkan hasil fitnes terminimum yaitu 0.0005008995604579987 dengan nilai x = 3 dan y = 2

BAB IV
PENUTUP

4.1 Kesimpulan

Dari hasil observasi di atas, dapat disimpulkan bahwa Algoritma Genetika dapat menghasilkan hasil yang optimal untuk mencari nilai minimum dan individu terbaik pada setiap generasi. Melalui proses seleksi orang tua, rekombinasi, mutasi dan seleksi survivor. Strategi yang digunakan untuk seleksi orang tua kami menggunakan metode roulette wheel, untuk rekombinasi atau pindah silang kami menggunakan metode satu titik potong, dan untuk mutasi kami menggunakan metode nilai acak dengan representasi integer. Strategi - strategi tersebut sudah dipertimbangkan untuk melakukan observasi.

4.2 Link Video dan Colab

Video :

Colab : https://bit.ly/CollabKEL20