LAPORAN TUGAS PEMROGRAMAN 01 - SEARCHING
MATA KULIAH PENGANTAR KECERDASAN BUATAN

Disusun oleh :

Kelompok DI

1. **Citakamalia (1301204122)**
2. **Muhammad Reyfasha Ilhami (1301204461)**
3. **Ruh Devita Widhiana (1301204021)**

PROGRAM STUDI S1 INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

BANDUNG

2022

BAB I
PENDAHULUAN

1.1 Latar Belakang

Algoritma genetika adalah suatu algoritma pencarian yang meniru mekanisme dari genetika alam. Algoritma genetika dimulai dengan kumpulan solusi yang disebut dengan populasi. Solusi-solusi dari sebuah populasi diambil dan digunakan untuk membentuk populasi yang baru. Hal ini dimotivasi dengan harapan bahwa populasi yang baru dibentuk tersebut akan lebih baik daripada yang lama. Solusi-solusi yang dipilih untuk membentuk solusi-solusi yang baru dipilih sesuai dengan fitness mereka masing-masing (Maulik and Bandyopadhyay, 2000).

Untuk memeriksa hasil optimasi, kita membutuhkan fungsi *fitness* yang menandakan solusi yang sudah dikodekan. Selama berjalan, *parent* digunakan untuk reproduksi, pindah silang dan mutasi untuk menciptakan keturunan. Jika Algoritma Genetika didesain secara naik, populasi akan mengalami konvergensi dan akan didapatkan sebuah solusi yang optimum (Putra, 2018).

1.2 Permasalahan

Pada Permasalahan ini, kita ditugaskan untuk menganalisis dan mendesain Genetic Algorithm (GA) serta mengimplementasikannya ke dalam suatu program komputer untuk mencari nilai □ dan □ sehingga diperoleh nilai minimum dari fungsi:

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

Dengan domain (batas nilai) untuk □ dan □:

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

BAB II
PEMBAHASAN

2.1 Strategi Penyelesaian

1. Desain kromosom dan metode dekode-nya

Desain Kromosom merupakan pemilihan kandidat secara probabilistik dengan tujuan untuk direproduksi, sedangkan gen setiap orang tua akan diturunkan ke-anaknya. Kromosom dari kode solusi yang baik akan direproduksi dengan peluang yang tinggi. Metode yang digunakan pada laporan ini adalah representasi biner, dimana kromosom berbentuk himpunan bilangan bernilai 0 atau 1.

2. Ukuran populasi

Populasi adalah seluruh jumlah kromosom atau individu yang ada pada setiap generasi. Pada percobaan ini, kami menggunakan populasi sebanyak 50 kromosom per generasi.

3. Metode pemilihan orangtua

Pemilihan Orang Tua merupakan seleksi orang tua yang akan dipilih untuk proses *crossover* dan mutasi dengan tujuan untuk mendapatkan calon *parent* yang baik, sehingga dapat menghasilkan keturunan yang baik. Semakin tinggi nilai fitness orang tua maka semakin besar kemungkinan untuk terpilih. Pada pemilihan orang tua diharuskan melakukan pencarian nilai fitness, yang nantinya akan digunakan pada tahap pemilihan berikutnya. Setiap orang tuaakan mendapatkan probabilitas reproduksi, besarnya nilai tersebut tergantung pada nilai diri orang tua terhadap semua orang tua lainnya.

Metode yang digunakan pada laporan ini adalah *Roulette Wheel*, metode ini termasuk ke dalam teknik seleksi orangtua yang proporsional terhadap nilai fitness-nya. Artinya, semakin besar nilai fitness suatu individu, semakin besar pula peluangnya untuk terpilih sebagai orangtua.

4. Metode operasi genetik (pindah silang dan mutasi)

Crossover (pindah silang) adalah proses persilangan antara dua kromosom yang ada, nantinya akan membentuk kromosom baru yang dimana diharapkan menjadi solusi terbaik. Metode *crossover* yang digunakan pada laporan ini adalah rekombinasi satu titik.

Proses ini dilakukan dengan penambahan nilai acak yang sangat kecil dengan probabilitas rendah pada variabel keturunan. Proses mutase ini akan melakukan penggantian suatu gen dengan gen yang baru yang dilakukan secara acak. Metode mutasi yang digunakan pada laporan ini adalah mutasi untuk representasi biner.

5. Probabilitas operasi genetik ($\square_c$ dan $\square_m$)

Probabilitas *Crossover* adalah berapa kali persilangan terjadi untuk kromosom dalam satu generasi, yaitu peluang dua kromosom bertukar beberapa bagiannya. Kami menggunakan P_c sebanyak 1.

Probabilitas Mutasi berguna untuk menentukan berapa banyak kromosom yang harus bermutasi dalam satu generasi; tingkat mutasi berada dalam kisaran [0, 1]. Tujuan dari mutasi adalah untuk mencegah GA dari konvergen ke optima lokal. Kami menggunakan P_m sebesar 1.

6. Metode pergantian generasi (seleksi *survivor*)

Suatu skema penggantian individu, yang disebut seleksi *survivor*, diterapkan sehingga menghasilkan populasi baru. Proses ini akan terus berulang sampai kondisi berhenti dipenuhi. Kondisi berhenti bisa berupa sejumlah generasi atau sejumlah individu tertentu. Metode seleksi *survivor* yang kami gunakan yaitu *Steady State Model*, pada *Steady State Model* tidak semua kromosom diganti. Penggantian dilakukan hanya pada sejumlah kromosom tertentu kami menggunakan 50 kromosom. *Survivor* dipilih dengan cara membuang kromosom dengan nilai fitness terkecil didalam populasi, sehingga menyisakan kromosom dengan nilai fitness terbaik

7. **Kriteria penghentian evolusi (*loop*)**

Loop pada kode yang kami buat, akan berhenti jika kandidat kromosom untuk seleksi orang tua selalu sama dengan kandidat yang sudah terpilih dan generasi sudah mencapai generasi ke 100.

2.2 Proses yang Diimplentasikan

1. **Dekode kromosom**

Metode yang kami gunakan adalah representasi biner, dimana kromosom berbentuk bilangan biner 0 atau 1. Bilangan-bilangan tersebut akan merepresentasikan nilai x dan nilai y pada rumus

$$h(\square, \square) = \frac{(\square\square\square\ \square + \square\square\square\ \square)^2}{\square^2+\square^2}$$

dengan *domain* (batas nilai) untuk x dan y : $-5 \leq \square \leq 5$ dan $-5 \leq \square \leq 5$

Untuk menentukan representasi kromosom biner, digunakan rumus sebagai berikut,

$$x = r_b + \frac{(r_a - r_b)}{\sum_{i=1}^{N} 2^{-i}} (g_1.2^{-1} + g_2.2^{-2} + ... + g_N.2^{-N})$$

```
def Representation(self, x1, x2, g): #fungsi representasi biner
    tp = [2**-i for i in range(1, len(g) + 1)]
    return x2 + ((x1 - x2) / sum(tp) * sum([g[i] for i in range(len(g))]))
```

2. **Perhitungan *fitness***

Minimisasi: f = 1/h + a. Tujuannya adalah memaksimalkan sebuah fungsi, maka fungsi fitness yang digunakan adalah fungsi yang dioptimasi tersebut.

```
def fitness(x, y):
    return heuristic(x, y)
```

3. **Pemilihan orangtua**

Metode yang digunakan adalah *Roulette Wheel*

```
def seleksi_ortu(k):
    ortu = []
    (variable) list_weight: list  b: fitness(b.x, b.y), population))
    list_weight = [list_fitness[i] / sum(list_fitness) for i in range(len(population))]
    while len(ortu) != k:
        candidate = random.choices(population, weights=list_weight)[0]
        if not exist(ortu, candidate):
            ortu.append(candidate)
    return ortu
```

4. ***Crossover* (pindah silang)**

Metode yang digunakan adalah rekombinasi satu titik (*1-point crossover*).

```
def Crossover(ortu1, ortu2):
    position = random.randint(1, len(ortu1.biner) - 2)
    biner_child1 = ortu1.biner[:position] + ortu2.biner[position:]
    biner_child2 = ortu2.biner[:position] + ortu1.biner[position:]
```

5. **Mutasi**

Metode yang digunakan adalah mutasi biner. Untuk setiap posisi gen di dalam suatu kromosom, bangkitkan suatu bilangan acak antara 0 sampai 1. Jika bilangan acak tersebut lebih kecil atau sama dengan probabilitas mutasi P_m, maka gen pada posisi tersebut dimutasi. Jika sebaliknya, maka gen tersebut tidak dimutasi.

```
pm = random.uniform(0, 50)
if pm < 1:
    idx_mutation = random.randint(0, len(biner_child2) - 1)
    if biner_child2[idx_mutation] == 1:
        biner_child2[idx_mutation] = 0
    else:
        biner_child2[idx_mutation] = 1
```

6. **Pergantian Generasi**

Metode yang digunakan adalah *Steady State*. Tidak semua kromosom diganti, Penggantian dilakukan hanya pada sejumlah kromosom tertentu.

```
def seleksi_survivor():
    population.sort(key=lambda b: heuristic(b.x, b.y), reverse=True)
    while len(population) != 50:
        population.pop()
```

BAB III
PENUTUP

3.1 Kesimpulan

Algoritma genetika merupakan teknik pencarian dalam ilmu komputer untuk menyelesaikan optimisasi dan masalah pencarian algoritma ini bersifat evolusioner yang menggunakan teknik pada ilmu biologi seperti warisan (elitisme), mutasi, seleksi alam (seleksi survivor) dan rekombinasi (crossover). Pada tugas ini kami mengimplementasikan algoritma genetika dengan menggunakan bahasa pemrograman python. Setelah program dijalankan, output yang dihasilkan program adalah sebagai berikut.

```
Generasi 1
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 2
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 3
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 4
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 5
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 6
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 7
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 8
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 9
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 10
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 11
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 12
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 13
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 14
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 15
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 16
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 17
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 18
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
```

…

```
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 83
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 84
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 85
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 86
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 87
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 88
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 89
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 90
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 91
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 92
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 93
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 94
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 95
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 96
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 97
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 98
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 99
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
Generasi 100
Terbaik [0, 1, 0, 1, 0, 1, 1, 1, 1, 0] (-15.64516129032258, -36.29032258064516) - 99.99920780509323
```

3.2 Referensi

Maulik, U. and Bandyopadhyay, S. (2000) '*Genetic algorithm-based clustering technique*', *Pattern Recognition*.

Putra, I. M. S. (2018). *PENERAPAN ALGORITMA GENETIKA DAN IMPLEMENTASI DALAM MATLAB*, 2.

Suyanto. *Metaheuristic Search.* CII-2M3 Pengantar Kecerdasan Buatan Pokok Bahasan 04.

Link video presentasi :

https://drive.google.com/file/d/1rZvv52RFDHQvegMsxU0J340ZqquQnWX-/view?usp=sharing