LAPORAN TUGAS PEMROGRAMAN
PENGANTAR KECERDASAN BUATAN
ALGORITMA GENETIKA

Disusun Oleh :
Kelompok 1
Anyelir Belia Azzahra (1301200048)
Risma Amaliyah Mahmudah (1301204087)
Muhammad Rafi Irfansyah (1301204500)

Program Studi Informasi
Fakultas Informatika
Telkom University
2021/2022

BAB 1
PENDAHULUAN

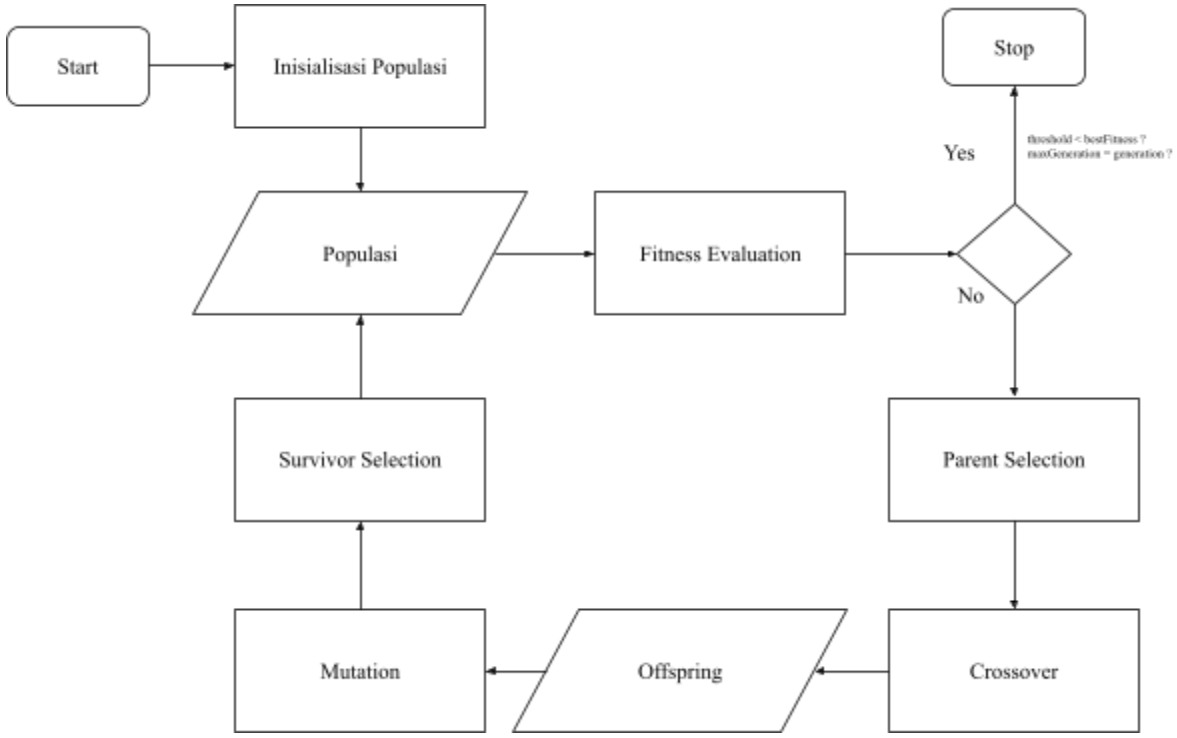
Genetic Algorithm (GA) merupakan salah satu metode heuristik yang digunakan untuk mengoptimasi sebuah proses. Karena merupakan metode heuristik, maka solusi yang diperoleh dari GA bukan yang terbaik, melainkan yang mendekati optimal.

Konsep GA terinspirasi dari teori evolusi Darwin dengan quote *"the strongest species that survive"*. Teori evolusi Darwin menegaskan bahwa organisme dapat bertahan hidup dengan beberapa proses yakni reproduksi (crossover) dan mutasi. Konsep tersebut kemudian diadaptasi pada algoritma komputasi untuk menemukan solusi pada masalah fungsi objektif.

Melalui beberapa proses yang terdapat di dalam pengimplementasian algoritma genetik pada suatu program komputer, seperti decode kromosom, perhitungan fitness, pemilihan orang tua, crossover, mutasi, dan pergantian generasi untuk menghasilkan output berupa kromosom terbaik serta hasil decode nilai kromosom terbaik yaitu nilai x dan y.

Chromosome adalah sebuah solusi yang dihasilkan oleh GA dan koleksi dari *chromosome* disebut juga populasi. Sebuah *chromosome* terdiri dari genetika dimana tiap *gene* dapat direpresentasikan dengan sebuah value berupa *numeric, binary, symbol* ataupun karakter tergantung masalah yang akan diselesaikan.

Siklus Genetic Algorithm



Dalam penyelesaian *Genetic Algorithm*, pertama harus didefinisikan populasi yang berisi individu beserta kromosomnya. Tahap selanjutnya melakukan *Fitness Evaluation* dari setiap individu dengan menghitung nilai fitness setiap kromosom. Kemudian mempertanyakan apakah generasi tersebut merupakan generasi terbaik atau tidak. Jika tidak, maka melakukan *Parent Selection* untuk menentukan individu mana yang akan dipilih untuk dilakukan *Crossover*. Kemudian dilakukan proses *Crossover* untuk menghasilkan individu baru (*offspring*). Setelah itu, melakukan mutasi untuk menggantikan gen yang hilang dari populasi selama proses seleksi serta menyediakan gen yang tidak ada pada populasi awal. Lalu, dilakukan *Survivor Selection* untuk penggantian generasi yang menghasilkan populasi baru. Kemudian melakukan proses *Fitness Evaluation*, jika hasilnya merupakan generasi terbaik, maka proses akan berhenti.

BAB 2
METODOLOGI PENELITIAN

2.1. Metode

- Desain kromosom dan metode pengkodean kami membentuk kromosom menggunakan metode pengkodean biner
- Pada pemilihan orang tua kami menggunakan metode Roulette Wheel selection
- Pada Pemilihan dan teknik operasi genetik (crossover dan mutasi) untuk proses crossover kami menggunakan metode single point crossover, sedangkan untuk mutasi menggunakan metode swap mutation
- Metode pergantian generasi (seleksi survivor) kami menggunakan metode elitisme
- Stop condition pada saat generasi = maxGeneration.

2.2. Parameter yang digunakan

- Desain kromosom : Kumpulan array dengan representasi data biner
- Probabilitas crossover (Pc) : 0.6 (60%)
- Probabilitas mutation (Pm) : 0.4 (40%)
- Ukuran populasi : 10
- Ukuran kromosom : 5

2.3. Fungsi yang dioptimasi

Pada Tugas Pemrograman 01 ini, dilakukanlah analisis, pendesainan, dan mengimplementasikan algoritma Genetic Algorithm (GA) ke dalam suatu program komputer untuk menemukan nilai minimum dari fungsi:

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

dengan domain (batas nilai) untuk *x* dan *y* :

$$-5 \leq x \leq 5\, dan\ -5 \leq y \leq 5$$

```
# Membuat array untuk data limit x dan y
x_limit = [-5, 5]
y_limit = [-5, 5]
```

```
# Perhitungan Fitness
def function(x, y):
    return (cos(x) + sin(y))**2 / (x**2 + y**2)
```

BAB 3
IMPLEMENTASI PROGRAM

3.1. Deklarasi Variabel Global

Menginisialisasikan domain (batas nilai) dari x dan y.

```
# Membuat array untuk data limit x dan y
x_limit = [-5, 5]
y_limit = [-5, 5]
```

Menginisialisasikan parameter yang digunakan.

```
# Inisialisai Populasi
population = 10

# Inisialisai Probabilitas Crossover (Pc)
prob_crossover = 0.6

# Inisialisai Probabilitas Mutasi (Pm)
prob_mutation = 0.4

# Inisialisai Kromosom
chromosom = 5

# Inisialisai Elitisme
elitisme = 2

# Inisialisai Generasi
generasi = 100

# Inisialisai Roulette
roulette = 7
```

3.2. Implementasi Fungsi

3.2.1. Fungsi Generate Populasi

Fungsi generate populasi di bawah digunakan untuk membuat populasi serta kromosom. Kelompok kami membuat populasi sebanyak 10 dengan 5 kromosom di setiap individunya.

```
# Fungsi pembuatan populasi yang berisi kromosom secara random
def generate_Populasi(populasi, kromosom):
    population = []
    for i in range(populasi):
        chromosom = []
        for j in range(kromosom):
          chromosom.append(random.randint(0,1))
        population.append(chromosom)
    return population
```

3.2.2. Fungsi Decode

Fungsi di bawah berfungsi untuk melakukan pengkodean gen yang membentuk individu agar nilainya tidak melebihi batas yang telah ditentukan.

```
# Decode Kromosom x
def decode_kromosom_x(x_limit, kromosom):
    set_krom = len(kromosom)//2

    krom_x = kromosom[:set_krom]

    len_x = len(krom_x)

    atas_x = 0
    bawah_x = 0

    for i in range(len_x):
        atas_x  += krom_x[i] * 2**-(i+1)
        bawah_x += 2**-(i+1)

    x = x_limit[0]+(atas_x*(x_limit[1]-x_limit[0])/bawah_x)
    return x
```

```
# Decode Kromosom y
def decode_kromosom_y(y_limit, kromosom):
    set_krom = len(kromosom)//2

    krom_y = kromosom[set_krom:]

    len_y = len(krom_y)

    atas_y = 0
    bawah_y = 0
    for i in range(len_y):
        atas_y += krom_y[i] * 2**-(i+1)
        bawah_y += 2**-(i+1)

    y = y_limit[0]+(atas_y*(y_limit[1]-y_limit[0])/bawah_y)
    return y
```

Di sini kami menggunakan kromosom dengan representasi biner (0 dan 1) yang di setiap kromosomnya memiliki panjang 5. Nilai fenotip akan didapatkan dengan melakukan decoding untuk setiap fenotip yang menggunakan rumus seperti di bawah ini:

$$x = r_b + \frac{(r_a - r_b)}{\sum_{i=1}^{N} 9.10^{-i}} (g_1.10^{-1} + g_2.10^{-2} + ... + g_N.10^{-N})$$

Di mana :

x = representasi kromosom

r_a = batas atas

r_b = batas bawah

g = genotype

3.2.3. Fungsi Perhitungan Fitness

Fungsi function di bawah digunakan untuk mencari nilai fitness, di mana nilai fitness tersebut sama dengan nilai fungsinya. Karena kami ingin mencari nilai minimum pada soal, kami menggunakan rumus f = h, di mana h merupakan persamaan yang ada pada soal.

```
# Perhitungan Fitness
def function(x, y):
    return (cos(x) + sin(y))**2 / (x**2 + y**2)
```

3.2.4. Fungsi Parent Selection

Pada tugas ini, kami menggunakan metode Roulette Wheel Parent Selection. Fungsi ini digunakan untuk mencari orang tua dari suatu individu secara acak dengan acuan nilai fitness pada setiap kromosom. Yang artinya, semakin besar nilai fitness pada kromosom maka semakin besar kemungkinan kromosom tersebut terpilih untuk menjadi orang tua.

```
# Pemilihan Orang Tua menggunakan Roulette Wheel Selection
def roulette_parent_selection(population, fitness, fitness_total):
    r = random.random()
    i = 0
    while r > 0:
      r -= fitness[i]/fitness_total
      i += 1
      if  i == len(population) - 1:
          break

    return population[i]
```

3.2.5. Fungsi Crossover

Pada tugas ini, kami menggunakan metode single point crossover. Fungsi ini digunakan untuk membuat kromosom baru antara dua kromosom yang menghasilkan dua kromosom dari parent yang telah dipilih sebelumnya. Proses ini akan terjadi ketika probabilitas yang didapatkan di bawah 0.6 atau kemungkinan yang terjadi adalah 60%.

```
# Crossover
def crossover(ortu1, ortu2):
    anak1 =[]
    anak2 =[]
    randnum = random.uniform(0.0, 1.0)
    titikPotong = int(round(random.uniform(0, 9)))
    if randnum <= prob_crossover:
        anak1 = ortu1[:titikPotong] + ortu2[titikPotong:]
        anak2 = ortu2[:titikPotong] + ortu1[titikPotong:]
    else : #jika tidak memiliki keturunan maka ortu1 dan ortu2 akan hidup di generasi berikutnya
        anak1 = ortu1
        anak2 = ortu2
    return anak1, anak2
```

3.2.6. Fungsi Mutasi

Kami menggunakan Mutasi dengan metode swap mutation. Fungsi mutasi ini digunakan untuk mengubah nilai fenotip dari gen atau menggantikan gen yang hilang dari populasi selama proses seleksi serta menyediakan gen yang tidak ada pada populasi awal.

```
# Mutasi
def mutation(anak, prob, len_krom):
    for i in range(len(anak)):
        for j in range(len_krom):
            if random.random() <= prob:
                anak[i][j] = [0, 1][not anak[i][j]]
    return anak
```

3.2.7. Fungsi Sort Fitness

Fungsi di bawah digunakan untuk mengurutkan fitness.

```
def sort_fitness(fitness):
    return sorted(range(len(fitness)), key=lambda k: fitness[k], reverse=True)
```

3.2.8. Fungsi Pergantian Seleksi (Selection Survivor)

Fungsi pergantian seleksi yang kami gunakan adalah menggunakan metode elitisme. Dengan menggunakan metode elitisme ini, maka satu kromosom terbaik akan selalu disimpan selama proses sehingga akan selalu ada kromosom dengan nilai fitness tertinggi. Dan jika ada kromosom baru yang lebih baik, maka satu kromosom terbaik yang disimpan sebelumnya ditukar dengan kromosom baru yang lebih baik tersebut.

```
# elitism
def elitism(populasi, best_c_generation, bad_c, total_fitness):
    if best_c_generation[1] > bad_c[0] and (best_c_generation[0] not in populasi):
        populasi[bad_c[2]] = best_c_generation[0]
        total_fitness = (total_fitness - bad_c[0] + best_c_generation[1])

        print("\nElitism Process")
        print(f'Chromosome {bad_c[2]+1}: {bad_c[1]}, fitness: {bad_c[0]}')
        print(f'changed to {best_c_generation[0]}, fitness: {best_c_generation[1]}\n')

    return populasi, total_fitness
```

3.2.9. Fungsi Kromosom Terbaik (Best Chromosome)

Pada fungsi ini digunakan untuk mencari kromosom terbaik.

```
# Fungsi untuk menentukan kromosom terbaik
def best_kromosom_selection(population):
    max_fitness = -999

    for kromosom in n_population:
        # kromosom_a, kromosom_b = split_kromosom(kromosom)
        x1 = decode_kromosom_x(x_limit, kromosom)
        x2 = decode_kromosom_y(y_limit, kromosom)
        fitness = function(x1, x2)

        if  max_fitness < fitness:
            max_fitness = fitness
            max_kromosom = kromosom

    return max_kromosom, max_fitness, x1, x2
```

3.2.10. Main Program

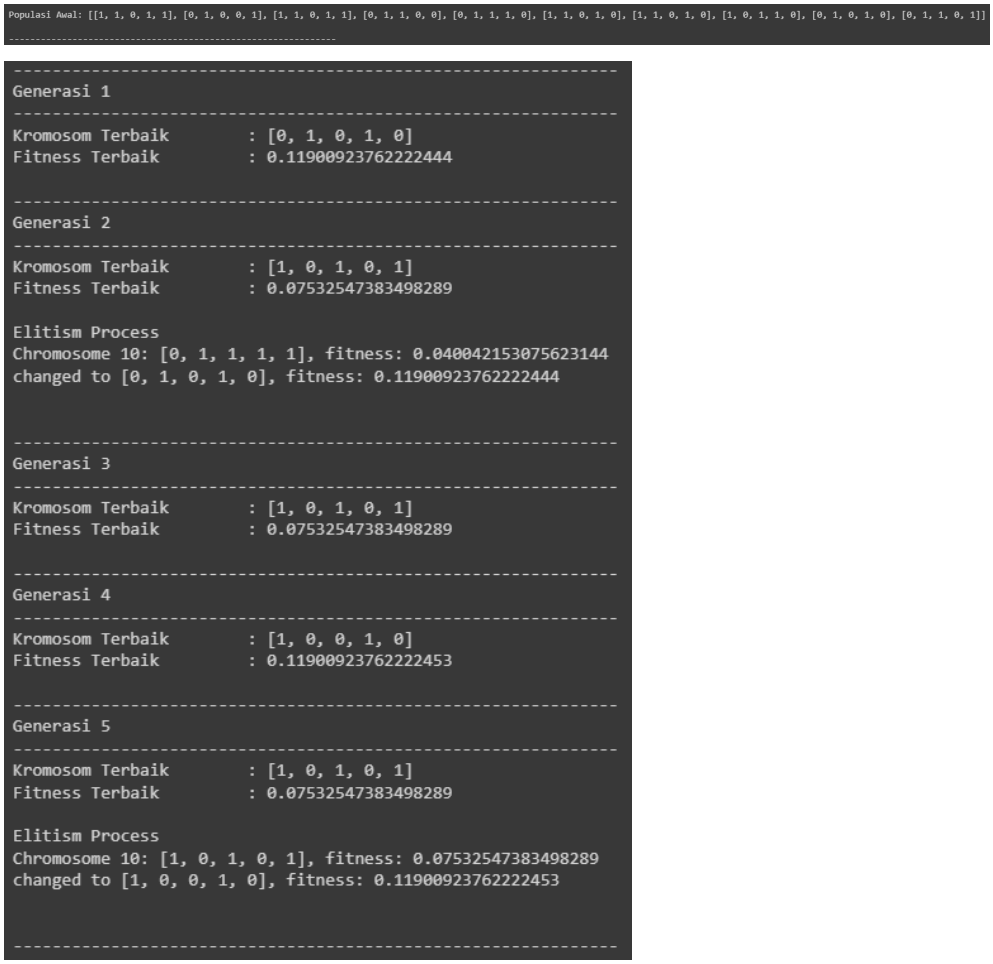
```
#Main Program
n_population = generate_Populasi(population, chromosom)
print("Populasi Awal:", n_population)

kromosom_terbaik = []

# Perulangan untuk melakukan proses seleksi populasi
for gen in range(generasi):

    # Inisialisasi variabel untuk proses perhitungan algoritma genetika
    tot_fitness, c_count, idx = 0, 999, 0
    c_data, c_best, c_bad, fitness, new_pop, anak = [], [], [], [], [], []

    print('\n-----------------------------------------------------------------')
    print('Generasi', gen+1)
    print('-----------------------------------------------------------------')

    # Perulangan untuk mencari nilai phenotype dan nilai fungsi / fitness pada setiap kromosom
    for i , kromosom in enumerate(n_population):
        # kromosom_a, kromosom_b = split_kromosom(kromosom)
        x = decode_kromosom_x(x_limit, kromosom)
        y = decode_kromosom_y(y_limit, kromosom)
        nilai_fitness = function(x, y)
        fitness.append(nilai_fitness)
        tot_fitness += nilai_fitness

    # Pencarian Fitness Terkecil Dalam Suatu Generasi
    if gen != 0 and nilai_fitness < c_count:
        c_count = nilai_fitness
        c_bad = [nilai_fitness, kromosom, i]


    # Pemilihan Kromosom Dengan Fitness Terbaik
    c_best = best_kromosom_selection(population)

    print("Kromosom Terbaik\t:", c_best[0])
    print("Fitness Terbaik\t\t:", c_best[1])
```

```
    # Proses Elitisme untuk memasukkan kromosom terbaik pada generasi sebelumnya
    if gen != 0:
        most_best = sorted(kromosom_terbaik, key=lambda x: x[1], reverse=True)[0]
        n_population, tot_fitness = elitism(n_population, most_best, c_bad, tot_fitness)

    kromosom_terbaik.append(c_best)

    # Perulangan untuk melakukan seleksi orang tua, crossover dan mutasi anak untuk mendapatkan populasi generasi selanjutnya
    if  gen != generasi-1 :
        for i in range(population // 2):
            ortu1 = roulette_parent_selection(n_population, fitness, tot_fitness)
            ortu2 = roulette_parent_selection(n_population, fitness, tot_fitness)

            anak_ = crossover(ortu1, ortu2)
            anak1, anak2 = mutation(anak_, prob_mutation, len(kromosom))

            new_pop.append(anak1)
            new_pop.append(anak2)

            n_population = new_pop

# Memanggil fungsi untuk menentukan kromosom terbaik pada keseluruhan generasi
print('\n-----------------------------------------------')
print('Hasil Akhir Kromosom Terbaik')
print('-----------------------------------------------')
print('Kromosom Terbaik\t= ', most_best[0])
print('nilai x\t\t= ', most_best[2])
print('nilai y\t\t= ', most_best[3])
print('Nilai Fungsi / Fitness\t=', most_best[1])
print('-----------------------------------------------')
```

BAB 4
HASIL IMPLEMENTASI

4.1. Hasil Output

4.1.1. Proses Generate Populasi, Kromosom, Decode, dan Nilai Fitness



Populasi akan berhenti saat mencapai Generasi ke-100, dikarenakan ukuran generasi adalah 100. Untuk proses generate kromosom dilakukan sesuai dengan parameter yang telah ditentukan sejak awal yaitu, populasi sebanyak 10 dan kromosom sebanyak 5.

4.1.2. Hasil Kromosom terbaik, serta nilai x dan y

```
-----------------------------------------------
Hasil Akhir Kromosom Terbaik
-----------------------------------------------
Kromosom Terbaik        =  [0, 1, 0, 1, 1]
nilai x                 =  5.0
nilai y                 =  -3.571428571428571
Nilai Fungsi / Fitness  =  0.17144340638327507
-----------------------------------------------
```

Nilai x didapatkan dari proses decode setengah kromosom pertama yang berhubungan dengan limit yang telah ditentukan dan nilai y didapatkan dari proses setengah kromosom selanjutnya dengan limit yang telah ditentukan serta nilai fitness pada setiap kromosom.

BAB 5
KESIMPULAN

Dari hasil di atas, dapat disimpulkan bahwa dari rumus fungsi h(x, y) pada soal dengan nilai x adalah 5.0 dan nilai y adalah -3.571428571428571, maka nilai h minimum yang didapatkan adalah 0.17144340638327507. Adapun kromosom terbaik didapat yaitu [0, 1, 0, 1, 1].

Source Code :
https://colab.research.google.com/drive/1hFPnnA5ArFRus5G9wVosd-vXkF-P_Dtn?usp=sharing

Video Presentasi :
https://www.youtube.com/watch?v=4FLTNhi0Nyc

DAFTAR PUSTAKA

Optimasi dengan Genetic Algorithm Menggunakan Python. (2020, July 29). School of Computer Science; socs.binus.ac.id. https://socs.binus.ac.id/2020/07/29/optimasi-dengan-genetic-algorithm-menggunakan-python/

Python | Single Point Crossover in Genetic Algorithm - GeeksforGeeks. (2019, October 11). GeeksforGeeks; www.geeksforgeeks.org. https://www.geeksforgeeks.org/python-single-point-crossover-in-genetic-algorithm/?ref=lbp