TUGAS PEMROGRAMAN 01 - SEARCHING

Mata Kuliah Pengantar Kecerdasan Buatan

Disusun oleh:

Berlian Muhammad Galin Al Awienoor	(1301204378)
Kiki Dwi Prasetyo	(1301204027)

KELOMPOK 7

PROGRAM STUDI S1 INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
2021/2022

DAFTAR ISI

JUDUL 1
DAFTAR ISI 2
PENDAHULUAN 3
- Latar Belakang 3
- Tujuan 3
- Rumusan Masalah 3

PEMBAHASAN 4
- Algoritma Genetika 4
  - Inisialisasi Populasi 5
  - Fungsi *Fitness* 5
  - Seleksi 5
  - *Crossover* 5
  - Mutasi 6
- Pencarian Nilai Minimum 7
  - Parameter yang Digunakan 7
  - Rumus Fungsi 7
  - Representasi Data (*Integer*) 7

PENGKODEAN ALGORITMA GENETIKA 8
- Proses Generate Populasi Awal beserta Kromosom 8
- Proses Decode dan Menghitung Nilai Fitness 8
  - Decode 9
  - Menghitung Nilai Fitness 9
- Proses Seleksi 10
- Proses Crossover 10
- Proses Mutasi 11
- Proses Pergantian Generasi 11

HASIL ANALISIS 13
- Hasil Analisis dan Desan 13
- Proses Generate Populasi, Kromosom, Decode, dan Nilai Fitness 13
- Proses Berhenti dan Penentuan Kromosom Terbaik 14

PENUTUP 16
- Kesimpulan 16
- Daftar Pustaka 17
- Referensi 17
- Lampiran 17

PENDAHULUAN

A. Latar Belakang

Genetic algorithm adalah teknik pencarian (*searching*) yang di dalam ilmu komputer untuk menemukan penyelesaian perkiraan untuk optimisasi dan masalah pencarian. *Genetic algorithm* adalah kelas khusus dari algoritma evolusioner dengan menggunakan teknik yang terinspirasi oleh biologi evolusioner seperti warisan, mutasi, seleksi alam dan rekombinasi (*crossover*). Salah satu fungsi dari *genetic algorithm* adalah untuk mencari nilai minimum dari fungsi matematika yang sulit untuk dipecahkan oleh manusia.

B. Rumusan Masalah

Rumusan masalah pada tugas ini adalah mencari nilai *x* dan *y* sehingga diperoleh nilai minimum dari fungsi dibawah dengan melakukan analisis dan desain algoritma *Genetic Algorithm* (GA) serta mengimplementasikannya ke dalam suatu program komputer.

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

C. Tujuan

Tujuan dari disusunnya laporan ini adalah sebagai berikut:

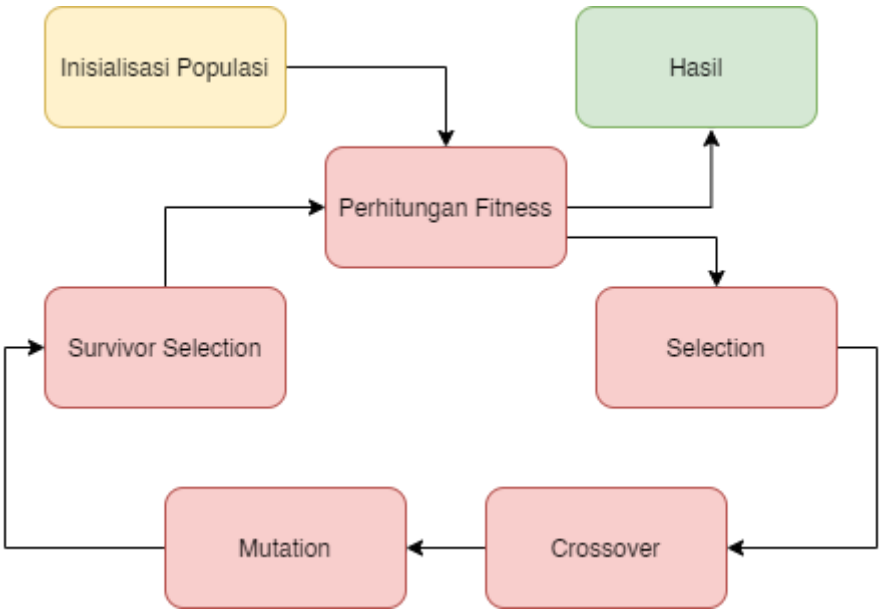
1. Untuk memenuhi tugas mata kuliah Pengantar Kecerdasan Buatan
2. Untuk menunjukkan dan menjelaskan hasil analisis dengan desain *Genetic Algorithm*
3. Untuk menunjukkan dan menjelaskan hasil akhir yang didapat dari fungsi yang diberikan

PEMBAHASAN

A. Algoritma Genetika

Genetic Algorithm (GA) pertama kali dikembangkan oleh John Holland pada tahun 1970-an di Amerika Serikat. Dia beserta murid-murid dan teman kerjanya menghasilkan buku berjudul "*Adaption in Natural and Artificial Systems*" pada tahun 1975.

Algoritma Genetik khususnya diterapkan sebagai simulasi komputer di mana sebuah populasi representasi abstrak (kromosom) dari solusi-solusi calon (individual) pada sebuah masalah optimisasi akan berkembang menjadi solusi-solusi yang lebih baik. Solusi-solusi tersebut dilambangkan dalam biner sebagai string '0' dan '1', walaupun dimungkinkan juga penggunaan penyandian (*encoding*) yang berbeda. Evolusi dimulai dari sebuah populasi individual acak yang lengkap dan terjadi dalam generasi-generasi. Dalam tiap generasi, kemampuan keseluruhan populasi dievaluasi, kemudian *multiple individuals* dipilih dari populasi sekarang *(current)* tersebut secara *stochastic* (berdasarkan kemampuan mereka), lalu dimodifikasi melalui mutasi atau rekombinasi menjadi bentuk populasi baru yang menjadi populasi sekarang (*current*) pada iterasi berikutnya dari algoritma.



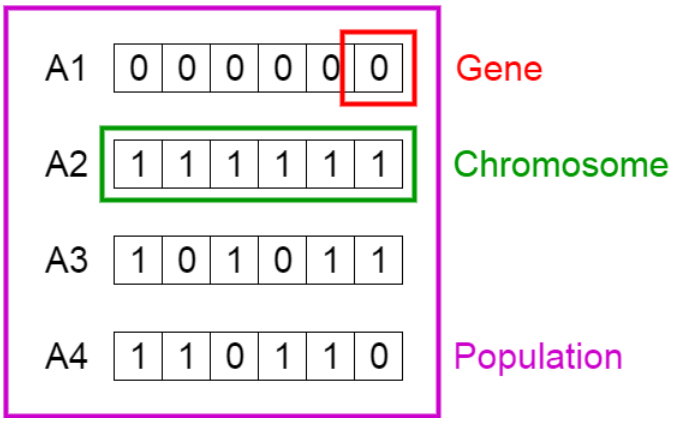
Algoritma genetika adalah heuristik pencarian yang terinspirasi oleh teori evolusi alami Charles Darwin. Algoritma ini mencerminkan proses seleksi alam di mana individu yang paling cocok dipilih untuk reproduksi untuk menghasilkan keturunan dari generasi berikutnya. Proses seleksi alam dimulai dengan pemilihan individu yang paling cocok dari suatu populasi. Mereka menghasilkan keturunan yang mewarisi karakteristik orang tua dan akan ditambahkan ke generasi berikutnya. Jika orang tua memiliki kebugaran yang lebih baik, keturunannya akan lebih baik daripada orang tua dan memiliki peluang lebih baik untuk bertahan hidup. Proses ini terus berulang dan pada akhirnya akan ditemukan generasi dengan individu yang paling cocok.

Gagasan ini dapat diterapkan untuk masalah pencarian. Kami mempertimbangkan satu set solusi untuk masalah dan memilih set yang terbaik dari mereka. Lima fase yang dipertimbangkan dalam algoritma genetika sebagai berikut:

1. Inisialisai Populasi

   Prosesnya dimulai dengan sekumpulan individu yang disebut populasi. Setiap individu adalah solusi untuk masalah yang ingin dipecahkan. Seorang individu dicirikan oleh seperangkat parameter (variabel) yang dikenal sebagai Gen. Gen bergabung menjadi string untuk membentuk kromosom (solusi).

   Dalam algoritma genetika, himpunan gen dari suatu individu direpresentasikan menggunakan string, dalam bentuk alfabet. Biasanya, nilai biner digunakan (string 1 dan 0). Kami mengatakan bahwa kami mengkodekan gen dalam kromosom.



2. Fungsi *Fitness*

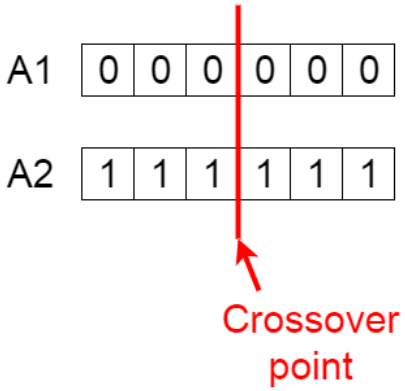
   Fungsi *fitness* menentukan seberapa *fit individual* (kemampuan individu untuk bersaing dengan individu lain). Ini memberikan skor *fitness* untuk setiap individu. Probabilitas bahwa individu akan dipilih untuk reproduksi didasarkan pada skor *fitness*nya.

3. Seleksi

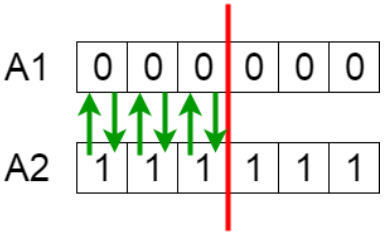
   Tujuan utama pada fase seleksi adalah untuk memilih individu yang paling cocok dan membiarkan mereka mewariskan gen mereka ke generasi berikutnya. Dua pasang individu (*parents*) dipilih berdasarkan skor *fitness* mereka. Individu dengan *fitness* yang tinggi memiliki peluang lebih besar untuk dipilih untuk reproduksi.

4. *Crossover*

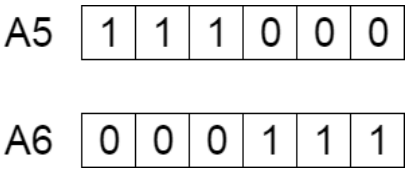
   Crossover adalah fase paling signifikan dalam algoritma genetika. Untuk setiap pasangan tertua yang akan dikawinkan, titik persilangan dipilih secara acak dari dalam gen. Misalnya, pertimbangkan titik *crossover* menjadi 3 seperti yang ditunjukkan di bawah ini.



Keturunan dibuat dengan menukar gen orang tua di antara mereka sendiri sampai titik *crossover* tercapai.

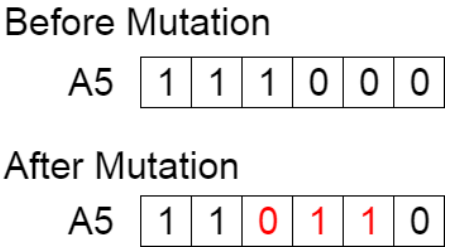


Keturunan baru ditambahkan ke populasi.



5. Mutasi

Pada keturunan baru tertentu yang terbentuk, beberapa gen mereka dapat mengalami mutasi dengan probabilitas acak yang rendah. Ini menyiratkan bahwa beberapa bit dalam string bit dapat dibalik.



Mutasi terjadi untuk menjaga keragaman dalam populasi dan mencegah konvergensi dini.

B. Algoritma Genetika Pencarian Nilai Minimum

1. Parameter yang Digunakan

   Desain kromosom : *array* dengan representasi data *integer*
   Probabilitas *crossover* : 0.9 (90%)
   Probabilitas mutasi : 0.1 (10%)
   Ukuran populasi : 10
   Ukuran kromosom : 10

2. Rumus Fungsi

   Dibawah ini merupakan rumus yang telah diberikan dari soal untuk mencari nilai fungsi serta nilai minimum dengan batasan x dan y. Kami merepresentasikan limit atau batas atas dan batas bawah pada soal menjadi 2 array yaitu batas_x dan batas_y, dan kami juga merepresentasikan fungsi dari soal kedalam sebuah fungsi seperti pada gambar dibawah.

```
# Membuat array untuk data batas x dan y
batas_x = [-5, 5]
batas_y = [-5, 5]
```

```
# Fungsi rumus yang akan dicari nilai minimumnya
def function(x,y):
    return (1/(((cos(x)+sin(y))**2/(x**2 + y**2))+ 0.01)) #sesuai soal dengan menggunakan rumus 1/h+a
```

   batas_x : batasan untuk x
   batas_y : batasan untuk y
   dengan batasan $-5 \leq x \leq 5$ dan $-5 \leq y \leq 5$ (sesuai soal)

3. Representasi Data (*Integer*)

   Pada desain algoritma genetika, kelompok kami menggunakan representasi data dalam bentuk integer sebagai acuan mencari nilai minimum pada fungsi yang diberikan.

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

   x : representasi kromosom (*integer*)
   r_b : batas bawah
   r_a : batas atas
   g : *genotype*

PENGKODEAN ALGORITMA GENETIKA

Pengkodean Algoritma Genetika

Untuk melakukan desain algoritma genetika, proses awal adalah pembentukan atau inisialisasi populasi beserta kromosomnya. Kemudian dilakukan proses *decode* serta menghitung nilai *fitness* dari tiap kromosom. Setelah itu, dilakukan proses seleksi pemilihan orang tua (*parents*) dengan cara *roulette wheel*. Lalu dilakukan proses *crossover parents* dan mutasi anak. Proses diatas dilakukan secara berulang-ulang sebanyak n-generasi dan mencari kromosom terbaik dari semua generasi (*looping* berhenti) dan proses algoritma genetika akan berhenti.

1. Proses Generate Populasi Awal beserta Kromosom

```
# Fungsi pembuatan populasi beserta kromosom-kromosomnya
def generate_populasi(n_populasi, n_kromosom):
    return [[(random.randint(0,9)) for _ in range(n_kromosom)] for _ in range(n_populasi)]
```

Fungsi generate_populasi merupakan fungsi yang digunakan untuk membuat suatu populasi beserta kromosomnya. Kelompok kami membuat sebanyak sepuluh individu dengan sepuluh kromosom tiap individu. Kromosom bernilai acak (*random*) dari nol sampai sembilan. *Array* pertama *generate array* kromosom sebanyak n yang dimasukkan pada main. *Array* kedua meng*generate* populasi sebanyak n yang dimasukkan pada main.

```
# Fungsi untuk melakukan proses mutasi pada anak
def mutasi(anak_1, anak_2):
    for i in range(len(anak_1)):
        p = random.random()
        if p < 0.1:
            anak_1[i] = random.randint(0,9)

        q = random.random()
        if q < 0.1:
            anak_2[i] = random.randint(0,9)

    return anak_1, anak_2
```

Fungsi mutasi merupakan fungsi untuk mengganti nilai kromosom pada suatu individu (anak) secara acak. Mutasi akan terjadi jika probabilitas *random* yang didapat kurang dari 0,1 atau 10%.

2. Proses Decode dan Menghitung Nilai Fitness

   Dibawah ini kode untuk proses *decode* serta menghitung nilai *fitness* dari setiap kromosom yang ada pada setiap generasi.

   2.1 Decode

```
# Fungsi decode untuk setiap kromosom pada suatu populasi
def decode(kromosom, batas) :
    kali = 0
    pembagi = 0
    for i in range(len(kromosom)) :
        num = kromosom[i]
        kali += num * (10**-(i+1))
        pembagi += 9 * (10**-(i+1))

    return batas[0] + (((batas[1] - batas[0]) / pembagi) * kali)
```

   Fungsi *decode* merupakan fungsi yang digunakan untuk melakukan pengkodean gen pembentuk individu agar nilainya tidak melebihi *range* yang telah ditentukan (*limit*). Karena kita menggunakan representasi integer maka untuk nilai pada pembagi terdapat angka 9 yang merepresentasikan bilangan integer yaitu dari 0 sampai 9.

   2.2 Menghitung Nilai Fitness

```
# Fungsi rumus yang akan dicari nilai minimumnya
def function(x,y):
    return (1/(((cos(x)+sin(y))**2/(x**2 + y**2))+ 0.01))
```

   Fungsi *function* merupakan fungsi yang digunakan untuk mencari nilai *fitness* yang mana nilai *fitness* sama dengan nilai fungsinya. Karena kita ingin mencari nilai minimum pada soal, kami menggunakan rumus f = 1/h+a, dimana h = persamaan yang diberikan pada soal dan a adalah pembagi agar hasil akhir tidak sama dengan 0 nilai a yang kami pakai adalah 0,01.

3. Proses Seleksi

```
# Fungsi untuk melakukan proses seleksi orang tua (parent) menggunakan cara roulette whell
def seleksi_roulette_whell(populasi, fitness, fitness_total):
    r = random.random()
    i = 0
    while r > 0:
      r -= fitness[i]/fitness_total
      i += 1
      if  i == len(populasi) - 1:
          break

    return populasi[i]
```

Fungsi seleksi_roulette_whell merupakan fungsi yang digunakan untuk mencari orang tua dari suatu individu secara acak dengan prinsip atau metode *roulette wheel* yaitu dengan acuan nilai *fitness* pada setiap kromosom. *Roulette Wheel* adalah salah satu metode seleksi orangtua yang paling populer. 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. Proses *Crossover*

```
# Fungsi untuk melakukan proses crossover anak
def crossover(ortu_1, ortu_2) :
    anak_1, anak_2, child = [], [], []
    pc = random.random() #pc = satu titik potong

    if pc < 0.9:
      anak_1[:1], anak_1[1:] = ortu_1[:1], ortu_2[1:]
      anak_2[:1], anak_2[1:] = ortu_2[:1], ortu_1[1:]
      child.append(anak_1)
      child.append(anak_2)
    else:
      child.append(ortu_1)
      child.append(ortu_2)

    return child
```

Fungsi *crossover* merupakan fungsi yang digunakan untuk membuat kromosom baru antara dua kromosom yang berbeda, lalu hasilnya adalah dua kromosom dari *parent* yang telah dipilih sebelumnya. Adapun proses *crossover* pada program ini menggunakan jenis *order crossover* yaitu dengan melakukan pembagian kedua *parent* untuk ditukar dan disatukan menjadi *child* baru. Proses ini akan terjadi ketika probabilitas yang didapatkan dibawah 0.9 atau kemungkinan terjadi adalah 90%.

5. Proses Mutasi Anak

```
# Fungsi untuk melakukan proses mutasi pada anak
def mutasi(anak_1, anak_2):
    for i in range(len(anak_1)):
        p = random.random()
        if p < 0.1:
            anak_1[i] = random.randint(0,9)

        q = random.random()
        if q < 0.1:
            anak_2[i] = random.randint(0,9)

    return anak_1, anak_2
```

Fungsi mutasi merupakan fungsi yang digunakan untuk mengubah *fenotype* dari gen atau mengubah nilai gen. Adapun pada proses mutasi ini digunakan jenis pemilihan nilai secara acak dalam interval 0-9 adapun kemungkinan dari mutasi di bawah 0.1 atau 10%.

6. Proses Pergantian Generasi

```
# Fungsi seleksi_survivor untuk memasukkan kromosom terbaik pada generasi sebelumnya
def seleksi_survivor(populasi, generasi_kromosom_terbaik, kromosom_terburuk, total_fitness):
    if  generasi_kromosom_terbaik[1] > kromosom_terburuk[0] and (generasi_kromosom_terbaik[0] not in populasi):
        populasi[kromosom_terburuk[2]] = generasi_kromosom_terbaik[0]
        total_fitness = (total_fitness - kromosom_terburuk[0]) + generasi_kromosom_terbaik[1]

        print('\nProses seleksi_survivor')
        print(f'Kromosom Ke-{kromosom_terburuk[2]+1}: {kromosom_terburuk[1]}, fitness: {kromosom_terburuk[0]}')
        print(f'diganti oleh   {generasi_kromosom_terbaik[0]}, fitness: {generasi_kromosom_terbaik[1]}\n')

    return populasi, total_fitness
```

```
# Fungsi untuk menentukan kromosom terbaik
def seleksi_kromosom_terbaik(populasi):
    max_fitness = -999

    for kromosom in populasi:
        kromosom_a, kromosom_b = bagi_kromosom(kromosom)
        x1 = decode(kromosom_a, batas_x)
        x2 = decode(kromosom_b, batas_y)
        fitness = function(x1, x2)

        if  max_fitness < fitness:
            max_fitness = fitness
            max_kromosom = kromosom

    return max_kromosom, max_fitness, x1, x2
```

```
# Perulangan untuk melakukan seleksi orang tua, crossover dan mutasi anak untuk mendapatkan populasi generasi selanjutnya
if  gen != generasi-1 :
    for i in range(n_populasi // 2):
        ortu_1 = seleksi_roulette_whell(populasi, data_fitness, total_fitness)
        ortu_2 = seleksi_roulette_whell(populasi, data_fitness, total_fitness)

        child = crossover(ortu_1, ortu_2)
        anak_1, anak_2 = mutasi(child[0], child[1])

        populasi_baru.append(anak_1)
        populasi_baru.append(anak_2)

    populasi = populasi_baru
```

Pada 2 fungsi di atas yaitu seleksi_survivor, seleksi_kromosom_terbaik, dan perulangan untuk melakukan seleksi orang tua, *crossover* dan mutasi anak untuk mendapatkan populasi generasi selanjutnya. Kelompok kami menggunakan prinsip elitisme, yang mana selalu menyimpan satu kromosom terbaik selama proses sehingga akan selalu ada kromosom dengan nilai fitness tertinggi. Dan jika muncul kromosom terbaru yang lebih baik maka satu kromosom yang disimpan sebelumnya ditukar dengan kromosom baru yang lebih baik tersebut.

HASIL ANALISIS

A. Hasil Analisis dan Desain

Berikut merupakan hasil analisis dari proses generate populasi mulai dari populasi generasi pertama hingga generasi terakhir.

```
Populasi Awal: [[9, 5, 3, 1, 2, 1, 4, 3, 1, 5], [6, 9, 5, 9, 9, 1,
8, 7, 1, 8], [8, 9, 5, 9, 4, 4, 9, 3, 0, 5], [6, 1, 7, 7, 4, 0, 7,
2, 9, 1], [9, 3, 0, 2, 4, 3, 9, 4, 2, 7], [2, 8, 7, 6, 2, 2, 7, 9,
4, 2], [4, 3, 2, 2, 8, 3, 3, 2, 1, 0], [9, 4, 1, 2, 1, 7, 6, 2, 2,
3], [1, 4, 2, 6, 8, 2, 0, 7, 3, 4], [8, 1, 2, 7, 9, 1, 5, 9, 7, 3]]
```

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


```
Populasi Awal: [[9, 5, 3, 1, 2, 1, 4, 3, 1, 5], [6, 9, 5, 9, 9, 1,
8, 7, 1, 8], [8, 9, 5, 9, 4, 4, 9, 3, 0, 5], [6, 1, 7, 7, 4, 0, 7,
2, 9, 1], [9, 3, 0, 2, 4, 3, 9, 4, 2, 7], [2, 8, 7, 6, 2, 2, 7, 9,
4, 2], [4, 3, 2, 2, 8, 3, 3, 2, 1, 0], [9, 4, 1, 2, 1, 7, 6, 2, 2,
3], [1, 4, 2, 6, 8, 2, 0, 7, 3, 4], [8, 1, 2, 7, 9, 1, 5, 9, 7, 3]]
```

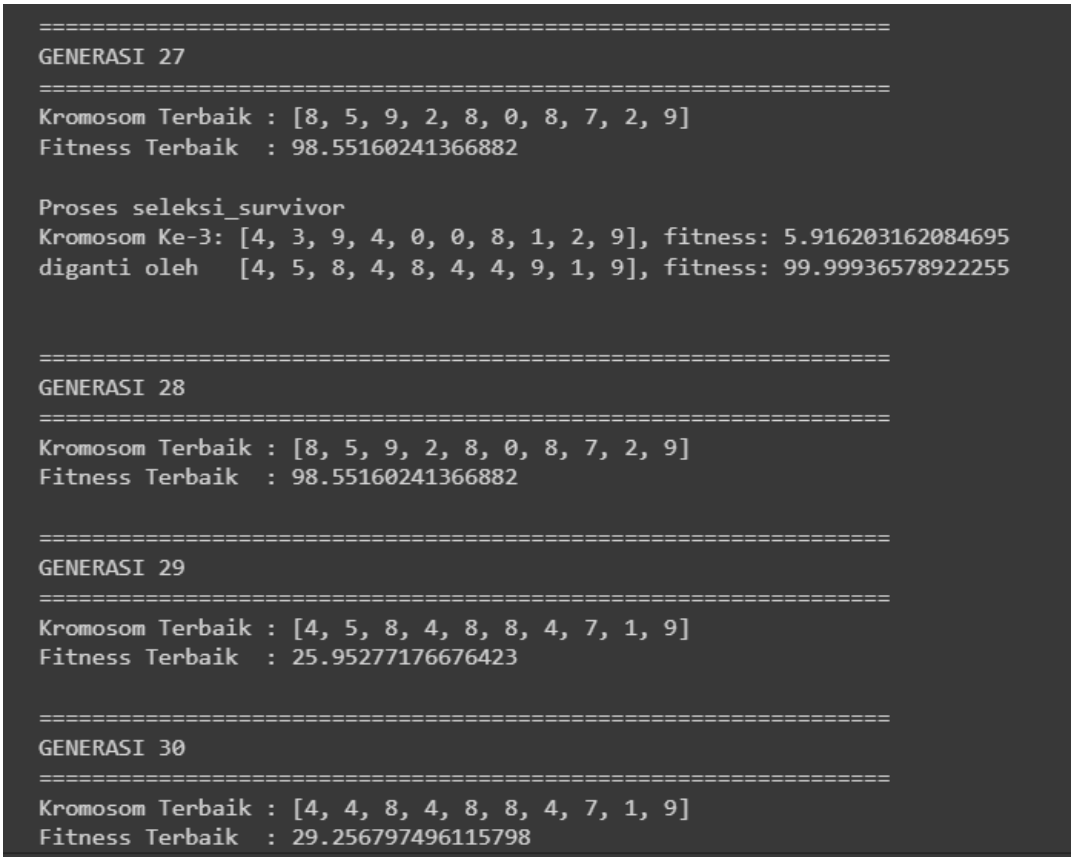

```
======================================================================
GENERASI 1
======================================================================
Kromosom Terbaik : [9, 4, 1, 2, 1, 7, 6, 2, 2, 3]
Fitness Terbaik  : 86.76143274919986

======================================================================
GENERASI 2
======================================================================
Kromosom Terbaik : [9, 9, 5, 9, 9, 1, 8, 7, 1, 8]
Fitness Terbaik  : 86.49857096844491

Proses seleksi_survivor
Kromosom Ke-4: [8, 3, 2, 2, 8, 3, 3, 2, 1, 0], fitness: 3.4221997832898827
diganti oleh   [9, 4, 1, 2, 1, 7, 6, 2, 2, 3], fitness: 86.76143274919986


======================================================================
GENERASI 3
======================================================================
Kromosom Terbaik : [9, 1, 9, 8, 9, 1, 4, 9, 6, 3]
Fitness Terbaik  : 94.10238090981774

Proses seleksi_survivor
Kromosom Ke-2: [4, 9, 5, 9, 5, 5, 8, 2, 1, 8], fitness: 0.22530597211469963
diganti oleh   [9, 4, 1, 2, 1, 7, 6, 2, 2, 3], fitness: 86.76143274919986
```



Gambar diatas merupakan contoh hasil dari *generate* populasi dari generasi pertama hingga generasi terakhir (ke-30) beserta kromosomnya. Untuk proses *generate* kromosom dilakukan sesuai parameter yang ditentukan jumlah populasi dan kromosomnya yang pada kasus ini yaitu 10 populasi dan 10 kromosom. Nilai x didapat dari proses *decode* setengah kromosom pertama (bagian kiri) yang bersangkutan dengan *limit* atau batas yang telah ditentukan dan nilai y didapat dari proses *decode* setengah kromosom selanjutnya (bagian kanan) dengan *limit* atau batas yang telah ditentukan serta nilai *fitness* pada tiap-tiap kromosom.

2. Proses Berhenti dan Penentuan Kromosom Terbaik

Setelah mendapatkan kromosom dengan nilai *fitness* terbesar dan keseluruhan generasi perulangan generasi beserta proses seleksi orang tua, *crossover*, mutasi dan *seleksi survivor* yang telah diterapkan telah dipenuhi maka proses akan berhenti dan menampilkan hasil akhir dari kromosom terbaik yang ada pada seluruh generasi seperti pada gambar kode dan hasil dibawah ini.

```
print('\n=====================================================')
print('           HASIL AKHIR KROMOSOM TERBAIK')
print('=====================================================')
print('Kromosom Terbaik   = ', hasil_terbaik[0])
print('x                  = ', hasil_terbaik[2])
print('y                  = ', hasil_terbaik[3])
print('Nilai Fitness      = ', hasil_terbaik[1])
print('=====================================================')
```

```
=====================================================
           HASIL AKHIR KROMOSOM TERBAIK
=====================================================
Kromosom Terbaik   =  [4, 5, 8, 4, 8, 4, 4, 9, 1, 9]
x                  =  -3.3151831518315182
y                  =  4.4909949099491
Nilai Fitness      =  99.99936578922255
=====================================================
```

PENUTUP

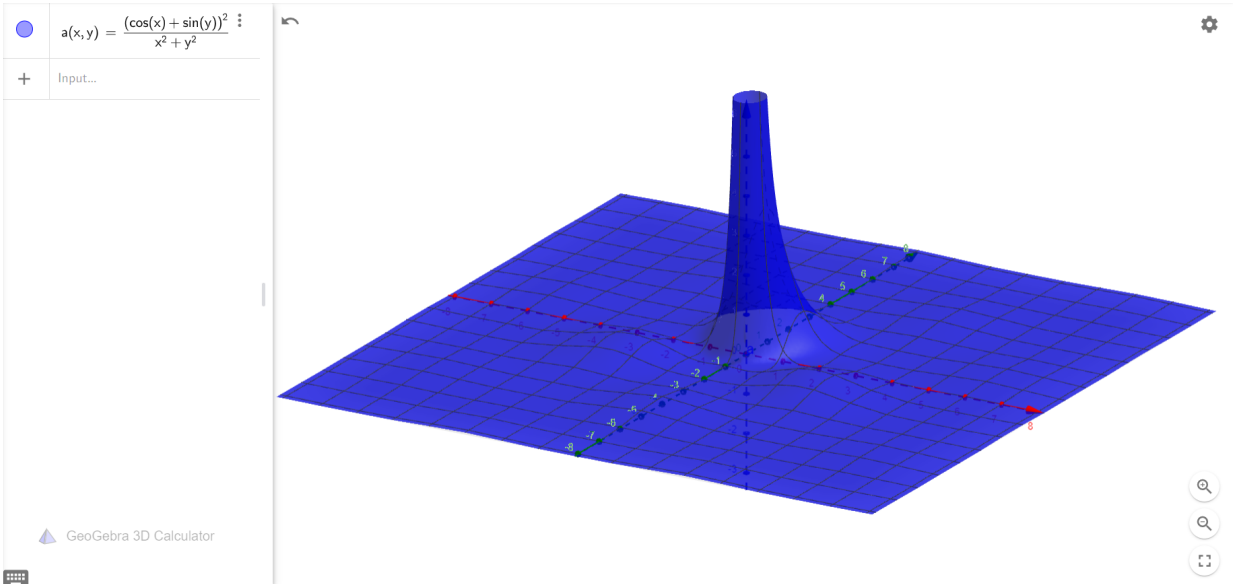
A. Kesimpulan

Setelah menjalankan proses algoritma genetika sesuai proses yang ada dengan benar, kita dapat membuktikan bahwa nilai minimum yang dapat kita hasilkan dari fungsi :

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

yaitu 99,9993 dengan salah satu nilai yang memenuhi persamaan fungsi pada nilai x adalah -3,3151 dan nilai y adalah 4,4909 dengan bentuk kromosom [4, 5, 8, 4, 8, 4, 4, 9, 1, 9]. Nilai tersebut dapat kita dapatkan dengan melakukan perulangan hingga generasi ke terakhir yaitu ke-30 dengan nilai fitness tertinggi yang awalnya didapatkan pada generasi ke-20.

```
============================================================
            HASIL AKHIR KROMOSOM TERBAIK
============================================================
Kromosom Terbaik    =  [4, 5, 8, 4, 8, 4, 4, 9, 1, 9]
x                   =  -3.3151831518315182
y                   =  4.4909949099491
Nilai Fitness       =  99.99936578922255
============================================================
```



Berdasarkan hasil pada gambar diatas, kita dapat menyimpulkan bahwa permasalahan yang diberikan terkait nilai minimum suatu fungsi dapat diselesaikan dengan algoritma genetika dengan baik. Algoritma genetika dapat memberikan solusi yang terbaik dari permasalahan yang telah diberikan.

B. Daftar Pustaka

Vijini, M. (2017). Introduction to Genetic Algorithms. [Halaman Web]. https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3. Diakses pada tanggal 31 Maret 2022.

Salman, F. (2020). Ilustrasi tentang algoritma genetika [Halaman Web]. https://medium.com/milooproject/panduan-ilustrasi-tentang-algoritma-genetika-902ca22b27dc. Diakses pada tanggal 31 Maret 2022.

Hermawanto, D. "Algoritma Genetika dan Contoh Aplikasinya".

David E. Goldberg, "Genetic Algorithms in Search, Optimization & Machine Learning", Addison-Wesley, 1989.

Lee, W. Kim, H. "Genetic Algorithm Implementation in Python".

C. Lampiran

1. Link GitHub:
   https://github.com/berlianm/Genetic-Algorithm

2. Link Hasil pengerjaan pemrograman:
   https://colab.research.google.com/drive/1IcuqooW30eaXeV0sJTha-YIZqFwJcH-3?usp=sharing#scrollTo=AWT13FDQv7Jp

3. Link Dokumen Laporan:
   https://docs.google.com/document/d/19ebISb2yYBukRHaP8fsxcvVF6sw5CQU_LBBUlAaL8TI/edit#

4. Link Dokumen Presentasi:
   https://docs.google.com/presentation/d/1KZBQndOoHxNg3YCdy-qbke-FC0jy1nnRiYBtCp10Iuo/edit#slide=id.g1095195a55a_0_0

5. Link Video Presentasi:
   https://drive.google.com/file/d/1CR0HWQ6D8Sev5t7m2JL0goBaFL0vJ0lU/view?usp=sharing