LAPORAN
TUGAS PEMROGRAMAN 1

Dibuat untuk memenuhi salah satu tugas mata kuliah Pengantar Kecerdasan Buatan

Oleh:
Priyan Fadhil Supriyadi / 1301190442
Claudia Mei Serin Sitio / 1301190424
Vena Erla Candrika / 1301194040

PROGRAM STUDI INFORMATIKA
FAKULTAS INFORMATIKA
2021

DAFTAR ISI

DAFTAR ISI…… 2
I. Pendahuluan……3
II. Hal yang diobservasi:…….. 4
1. Desain Kromosom dan Metode Pendekodean……...5
2. Ukuran Populasi……6
3. Pemilihan orangtua…… 6
4. Pemilihan dan teknik operasi genetik (*crossover* dan mutasi)…….. 7
5. Probabilitas operasi genetik (Pc dan Pm)…….. 8
6. Metode Pergantian Generasi (Seleksi Survivor)……9
7. Kriteria Penghentian Evolusi……...10
III. Nilai-nilai parameter GA yang Anda anggap paling optimum……11
IV. Kesimpulan……11
V. Link Presentasi…….. 12
References……13

I. Pendahuluan

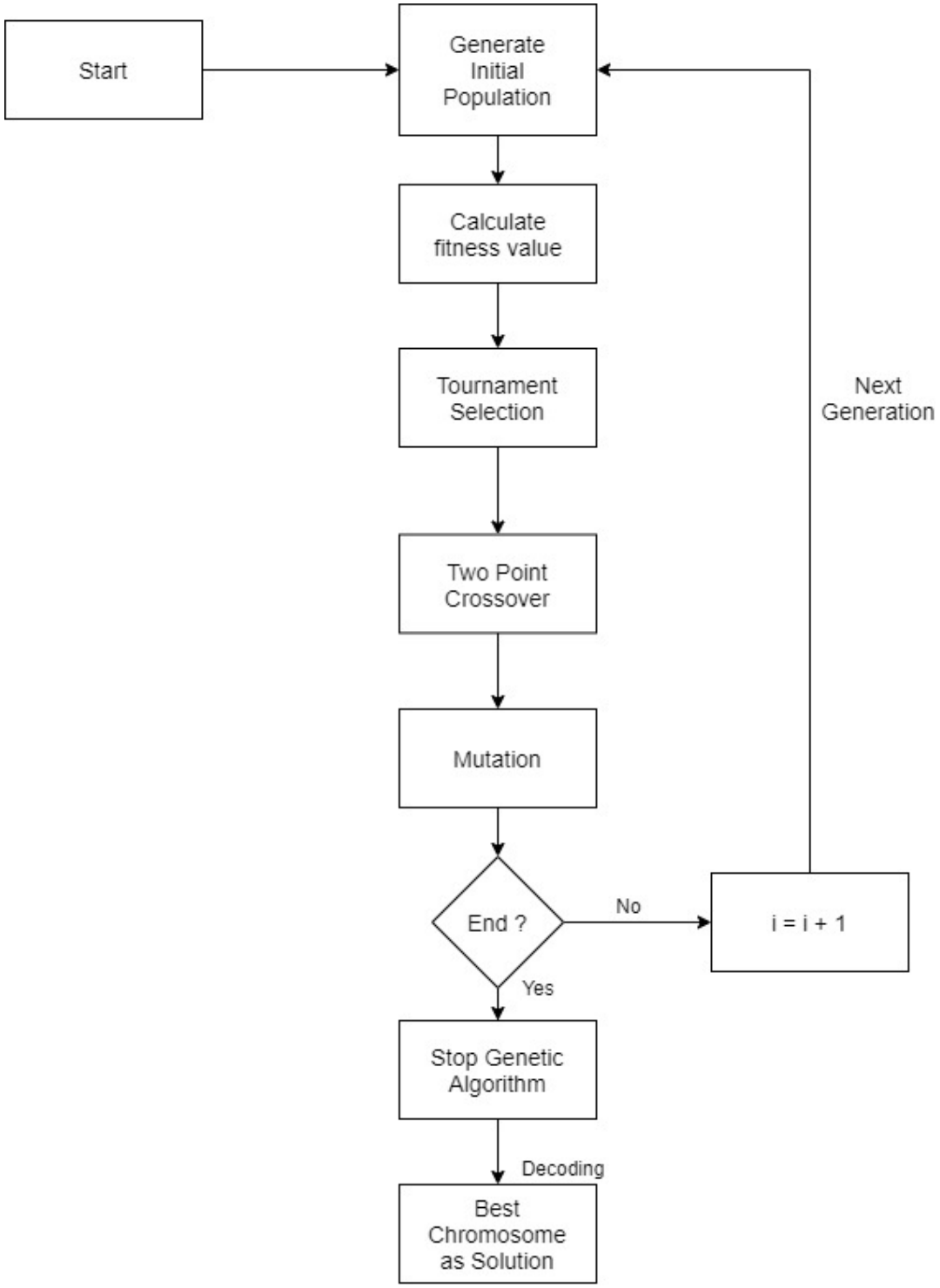
Algoritma adalah salah satu konsep matematika dan analisis untuk membantu seseorang dalam menyelesaikan suatu masalah atau persoalan. Algoritma disusun dari sekumpulan langkah berhingga, masing-masing langkah mungkin memerlukan satu operasi atau lebih. Algoritma umumnya dirancang untuk menyelesaikan suatu masalah spesifik dan dengan usaha yang paling minimum.

Algoritma genetika merupakan salah satu contoh metode heuristik dalam menyelesaikan masalah optimisasi. Sesuai namanya, cara kerja algoritma genetika adalah meniru proses evolusi dan perubahan struktur genetik pada makhluk hidup. Dalam seleksi alam, individu-individu yang lebih kuat akan mempunyai peluang untuk bertahan hidup lebih besar dan individu yang lebih kuat dari orang tuanya akan dilahirkan melalui proses penyilangan serta mutasi. Karena merupakan metode heuristik, maka solusi yang diperoleh dari GA bukan yang terbaik, melainkan yang mendekati optimal. (suhartono, 2020)

Algoritma genetika telah digunakan untuk memecahkan masalah sulit dengan fungsi objektif. Algoritma ini memelihara dan memanipulasi keluarga, atau populasi, solusi dan menerapkan strategi "survival of the fittest" dalam pencarian mereka untuk solusi yang lebih baik. (tang, zhu, yan, wu, & tian, 2002)

Dalam penerapan algoritma genetika yang mencari nilai fungsi maksimum kelompok 6 memilih menggunakan menggunakan Python. Penggunaan Python sangat membantu dalam memperoleh hasil algoritma genetika dengan iterasi yang relatif banyak. Banyaknya iterasi berpeluang menghasilkan calon solusi optimal karena terlahir dari individu-individu unggul seiring banyaknya perubahan generasi yang dilakukan. Selanjutnya, script Phyton yang dihasilkan diharapkan dapat dikembangkan lebih lanjut dalam menyelesaikan kasus pencarian nilai yang paling optimal dari sebuah fungsi objektif yang lebih kompleks.

II. Hal yang diobservasi:



Skema Proses Algoritma Genetik

1. Desain Kromosom dan Metode Pendekodean

Kromosom adalah sebuah solusi yang dihasilkan oleh genetic algorithm dan koleksi dari kromosom disebut juga populasi. Sebuah kromosom terdiri dari genetika dimana setiap gen dapat direpresentasikan dengan sebuah value berupa numeric, binary, symbol ataupun karakter tergantung masalah yang akan diselesaikan. (suhartono, 2020)

Pada kasus kelompok 6, dalam merepresentasikan genotype pada sebuah kromosom kelompok 6 menggunakan representasi biner. Hal ini telah disepakati karena pendefinisian representasi biner dianggap lebih mudah untuk dilakukannya pemrosesan data karena hanya berorientasi pada 2 jenis angka yaitu 0 dan 1. Kami mendesain panjang kromosom dengan jumlah sebanyak 8 bit karena nantinya dalam proses pendekodean kami akan membagi 8 bit ini masing masing menjadi 4 bit untuk setiap x dan y.

```
[ ]  def individual(length,min,max):
       individu = [random.randint(min,max) for x in range(length)]
       return individu
```

Berikut salah satu contoh individu yang terbentuk :

```
[0, 0, 1, 1, 1, 1, 0, 1]
```

Dalam melakukan metode pendekodean kami menggunakan rumus :

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

Dengan batasan yang diberikan yaitu $-1 \leq x \leq 2$ dan $-1 \leq y \leq 1$.

```
[ ]  limitX1 = -1
     limitX2 = 2
     limitY1 = -1
     limitY2 = 1

     length = 8
     min = 0
     max = 1

     populationSize = 50

     nGen = 100

     tournamentSize = 6
     crossoverProbability = 0.6
     mutationProbability = 0.1
```

```
[ ]  def decode(individu):

         indLength = int(len(individu))
         indLengthHalf = int(len(individu) / 2)
         tempIndividu = individu.copy()
         sumGen = 0

         for i in range(indLengthHalf):
             tempIndividu[i] = tempIndividu[i] * (2 ** -(i + 1))
             tempIndividu[i + indLengthHalf] = individu[i + indLengthHalf] * (2 ** -(i + 1))
             sumGen = sumGen + (2 ** -(i + 1))

         x = limitX1 + ((limitX2 - limitX1) / sumGen) * sum(tempIndividu[0:indLengthHalf])
         y = limitY1 + ((limitY2 - limitY1) / sumGen) * sum(tempIndividu[indLengthHalf:indLength])
         del tempIndividu
         return x, y
```

2. Ukuran Populasi

Populasi, merupakan sekumpulan individu yang akan diproses bersama dalam satu siklus proses evolusi. Individu yang dimaksud disini merupakan sebuah kromosom. Dalam menentukan ukuran populasi pada kasus kelompok 6, kami sepakat untuk menentukan jumlah individu pada setiap populasi sebanyak 50 karena kami telah melakukan perhitungan secara kombinasi yang merupakan suatu aturan pencacahan/penyusunan tanpa memperhatikan urutan objek dan menemukan yaitu sejumlah 30 kombinasi individu yang merupakan jumlah maksimal dalam menentukan kombinasi berbeda untuk kromosom dengan panjang 8 bit yang diisi masing masing gen-nya dengan variasi angka 0 dan 1.

```
populationSize = 50
```

```
[ ]  def population(populationSize):
         population = []
         for i in range(populationSize):
             population.append(individual(length,min,max))
         return population
```

Berikut contoh populasi dengan beberapa individu yang terbentuk :

```
[[0, 0, 1, 1, 0, 0, 1, 1], [1, 0, 0, 1, 1, 0, 1, 0], [1, 0, 0, 1, 0, 1, 1, 1], [0, 1, 1, 0, 1, 1, 0, 0], [0, 1, 1, 0, 1, 0, 0, 0]
```

3. Pemilihan orangtua

Selection merupakan proses pemilihan koleksi chromosome yang akan digunakan sebagai parent untuk menemukan genetika baru melalui proses crossover yang menghasilkan komposisi genetika baru dengan kombinasi genetik dari parent. (suhartono, 2020)

Dalam melakukan pemilihan orangtua kami menggunakan metode tournament selection, dalam bentuk paling sederhana, metode ini mengambil dua kromosom secara random dan kemudian menyeleksi salah satu yang memiliki nilai fitness paling tinggi untuk menjadi orang tua pertama. Cara yang sama dilakukan lagi untuk mendapatkan orang tua

kedua. Selain itu, pemilihan kromosom dapat dilakukan dengan cara sebuah grup, Kelompok 6 sepakat dengan memilih 6 kromosom yang dipilih secara acak untuk dilakukan tournament selection, karena jumlah 6 merupakan jumlah yang cukup untuk mewakili jumlah populasi yaitu sebanyak 50 individu.

$$h(x, y) = (\cos x\ 2 * \sin y\ 2\ ) + (x + y)$$

Fungsi h pada soal di atas akan dicari nilai maksimumnya dengan menggunakan library math pada fungsi fitness di bawah. Dimana untuk mencari nilai maksimum nilai fitnessnya akan sama dengan nilai fungsi h.

```
[ ]  def fitness(individu):
         x, y = decode(individu)
         return ((math.cos(x**2) * math.sin(y**2)) + (x + y))
```

```
[ ]  def tournament_selection(population, tournamentSize):
         winner = None
         randomNum = random.sample(range(0, len(population) - 1), tournamentSize)
         for i in randomNum:
             individu = (population[i])
             if winner is None or fitness(individu) > fitness(winner):
                 winner = individu
         return winner
```

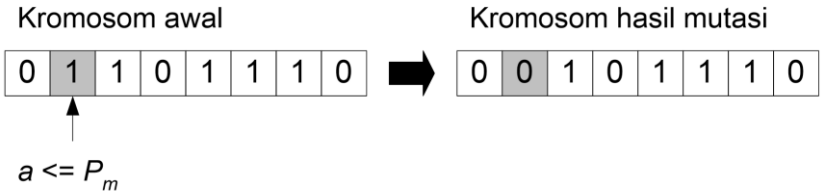
4. Pemilihan dan teknik operasi genetik (*crossover* dan mutasi)

Teknik operasi genetik crossover dipilih berdasarkan dengan metode representasi biner yang kami gunakan. Penggunaan representasi biner seperti yang telah disebutkan di atas akan lebih mempermudah proses crossover karena hanya menggunakan dua materi genetik yaitu 0 dan 1. Crossover sendiri merupakan operator genetik yang digunakan untuk memvariasikan pemrograman kromosom dari satu generasi ke generasi berikutnya. Oleh karenanya, kami memilih Two-Point Crossover yaitu dua titik acak dipilih pada kromosom individu dan materi genetik dipertukarkan pada batas titik tersebut. Idealnya, untuk kromosom yang panjang semakin banyak titik pada kromosom akan menghasilkan banyak kombinasi. Namun jika diterapkan pada individu dengan kromosom yang pendek memungkinkan munculnya kombinasi yang tidak bagus. Sehingga dengan hanya menggunakan Two-Point Crossover, individu tersebut akan memungkinkan munculnya banyak kombinasi dengan hasil yang relatif bagus.

Pada fungsi two_point_crossover ini digunakan probabilitas crossover (crossoverProbability) yang akan dibandingkan dengan suatu randomProbability untuk menentukan randomPoint sebagai titik potong. Jika randomProbability lebih kecil sama dengan crossoverProbability maka randomPoint akan dibangkitkan dan menghasilkan child1 dan child2.

```
[ ]  def two_point_crossover(individu1, individu2, crossoverProbability):
         randomProbability = random.random()
         if (randomProbability <= crossoverProbability):
             randomPoint = random.sample(range(0, length - 1), 2)
             randomPoint.sort()
             child1 = (individu1[0:randomPoint[0]] + individu2[randomPoint[0]:randomPoint[1]] + individu1[randomPoint[1]:length])
             child2 = (individu2[0:randomPoint[0]] + individu1[randomPoint[0]:randomPoint[1]] + individu2[randomPoint[1]:length])
             return child1, child2
         return individu1, individu2
```

Teknik operasi genetik lainnya ialah mutasi. Mutasi sendiri bersifat kecil, acak, berbahaya, dan jarang terjadi. Jika terjadi pun, kemungkinan besar mutasi itu tidak berguna. Sama halnya dengan teknik operasi genetik crossover, teknik operasi genetik mutasi juga dipilih berdasarkan metode representasi yang digunakan. Representasi biner juga memudahkan proses mutasi karena hanya perlu menukarkan materi genetik 0 menjadi 1 dan sebaliknya. Penentuan materi genetik suatu kromosom dilakukan dengan membangkitkan suatu bilangan acak antara 0 sampai 1 untuk setiap posisi gen. Kemudian bilangan tersebut akan dibandingkan dengan probabilitas mutasi (Pm). Jika bilangan acak (randomProbability) tersebut lebih kecil sama dengan mutationProbability (Pm) maka akan dilakukan mutasi dan sebaliknya. Dapat dilihat seperti pada gambar dibawah.




```
[ ]  def mutation(chromosome, mutationProbability):
         randomProbability = random.random()
         if (randomProbability <= mutationProbability):
             randomIndex = random.randint(0, length - 1)
             if chromosome[randomIndex] == 1:
               chromosome[randomIndex] = 0
             else :
               chromosome[randomIndex] = 1
```

5. Probabilitas operasi genetik (Pc dan Pm)

Dalam menjalankan teknik operasi genetik diatas terdapat probabilitas yang dibutuhkan. Probabilitas crossover (Pc) merupakan nilai yang akan menjadi pembanding untuk menentukan random point yaitu titik batas dimana gen akan saling disilangkan. Kami menggunakan variabel crossoverProbability dan memasukkan nilainya sebesar 0,6.

Seperti yang sudah dijelaskan diatas bahwa mutasi bersifat berbahaya, maka probabilitas mutasi (Pm) memiliki nilai yang kecil. Sehingga kemungkinan terjadinya akan sedikit. Banyaknya bit yang mengalami mutasi ditentukan dengan probabilitas mutasi (Pm). Kami menggunakan variabel mutationProbability dan memasukkan nilainya sebesar 0,1.

```
crossoverProbability = 0.6
mutationProbability = 0.1
```

6. Metode Pergantian Generasi (Seleksi Survivor)

Seleksi survivor menentukan individu-individu mana saja yang akan bertahan dari suatu generasi ke generasi berikutnya. Karena menggunakan model populasi generasional, dimana ketika terjadi regenerasi maka fungsi maksimumnya mungkin akan menurun, maka dilakukan seleksi survivor dengan elitisme yang dapat mempertahankan nilai fitness suatu generasi. Dengan cara mengambil nilai fitness tertinggi dari generasi sebelumnya. Setelah proses seleksi orang tua, crossover, mutasi, akan diambil nilai fitness paling rendah dari generasi selanjutnya dan digantikan oleh nilai fitness yang paling tinggi. Nilai fitness terbaik dari generasi sebelumnya akan menggantikan nilai fitness terkecil dari generasi selanjutnya. Dibandingkan dengan dua model seleksi survivor lainnya, elitisme memungkinkan individu terbaik dapat diregenerasi pada generasi selanjutnya sehingga menghasilkan individu dengan nilai fitness tetap yang terbaik.

Terdapat dua fungsi elitisme yang dibuat yaitu, fungsi elitism_first_best yang akan membandingkan nilai fitness dan mengembalikan individu terbaik pertama. Fungsi yang kedua ialah elitism_second_best yang juga membandingkan nilai fitness dan mengembalikan individu terbaik kedua.

```
[ ]  def elitism_first_best(population):
         best = None
         for i in range(len(population)):
             if (best == None or fitness(population[i]) > fitness(best)):
                 best = population[i]
         return best
```

```
[ ]  def elitism_second_best(population):
        best = elitism_first_best(population)
        best2nd = None
        for i in range(len(population)):
            if (best2nd == None or (fitness(population[i]) > fitness(best2nd) and fitness(population[i]) < fitness(best) and population[i] != best)):
                best2nd = population[i]
        return best2nd
```

Untuk pergantian generasi dilakukan dengan fungsi changeGeneration yang akan memanggil fungsi-fungsi operator genetik seperti tournament selection, two-point crossover, mutation dan membuat variabel generasi baru dan memanggil fungsi elitism_first_best dan elitism_second_best untuk mendapatkan generasi dengan individu terbaik.

```
[ ]  def changeGeneration(currentPopulation):
         newPopulation = []

         while len(newPopulation) != len(currentPopulation) - 2:
             parent1 = tournament_selection(currentPopulation, tournamentSize)
             parent2 = tournament_selection(currentPopulation, tournamentSize)
             while parent1 == parent2:
                 parent2 = tournament_selection(currentPopulation, tournamentSize)
             child1, child2 = two_point_crossover(parent1, parent2, crossoverProbability)

             mutation(child1, mutationProbability)
             mutation(child2, mutationProbability)

             newPopulation.append(child1)
             newPopulation.append(child2)

         newPopulation.append(elitism_first_best(currentPopulation))
         newPopulation.append(elitism_second_best(currentPopulation))

         return newPopulation
```

7. Kriteria Penghentian Evolusi

Penghentian evolusi merupakan proses yang menyatakan kondisi berhentinya evolusi dalam Algoritma Genetika. Penghentian evolusi terdapat pada berbagai macam kriteria. Kami menggunakan kriteria penghentian evolusi berdasarkan gen ke-n. Dimana proses evolusi akan berhenti setelah iterasi sebanyak n tersebut. Kami membuat variabel nGen dan memasukkan angka 100 sehingga evolusi akan berhenti pada gen ke-100.

```
gen ke- 97 --- dengan Cromosome : 10011111 --- Nilai Fitness Tertinggi : 2.474940307297132 --- (x, y) : (0.8, 1.0)
gen ke- 98 --- dengan Cromosome : 10011111 --- Nilai Fitness Tertinggi : 2.474940307297132 --- (x, y) : (0.8, 1.0)
gen ke- 99 --- dengan Cromosome : 10011111 --- Nilai Fitness Tertinggi : 2.474940307297132 --- (x, y) : (0.8, 1.0)
gen ke- 100 --- dengan Cromosome : 10011111 --- Nilai Fitness Tertinggi : 2.474940307297132 --- (x, y) : (0.8, 1.0)
================================ Gen Terbaik ===================================
Cromosome     :  10011111
Fitness       :  2.474940307297132
(X, Y)        :  (0.8, 1.0)
```

Program Utama :

```
[ ]  bestIndividu = []
     x = population(populationSize)
     print("First Gen", "--- dengan Cromosome :", "".join(str(j) for j in elitism_first_best(x)),
           "--- Nilai Fitness Tertinggi :", fitness(elitism_first_best(x)),
           "--- (x, y) :", decode(elitism_first_best(x)))
     bestIndividu.append(elitism_first_best(x))
     i = 0

     while i < nGen:
         i = i + 1
         x = changeGeneration(x)
         bestIndividu.append(elitism_first_best(x))
         print('gen ke-', i, "--- dengan Cromosome :", "".join(str(j) for j in elitism_first_best(x)),"--- Nilai Fitness Tertinggi :",

     print("================================ Gen Terbaik ===================================")
     BestGen = elitism_first_best(bestIndividu)
     print("Cromosome     : ", "".join(str(i) for i in BestGen))
     print("Fitness       : ", fitness(BestGen))
     print("(X, Y)        : ", decode(BestGen))
```

```
[ ]
    ` j in elitism_first_best(x)),
    ist(x)),

    j) for j in elitism_first_best(x)),"--- Nilai Fitness Tertinggi :", fitness(elitism_first_best(x)),"--- (x, y) :", decode(elitism_first_best(x)))

    ==========================")
```

III. Nilai-nilai parameter GA yang Anda anggap paling optimum

- Limit X1,X2,Y1,Y2 sebagai batasan sesuai dengan soal yang diberikan.
  - LimitX1 bernilai -1 sebagai batas bawah
  - LimitX2 bernilai 2 sebagai batas atas
  - LimitY1 bernilai -1 sebagai batas bawah
  - LimitY2 bernilai 1 sebagai batas atas
- Length bernilai 8 untuk panjang 8 bit
- Min berguna untuk nilai minimal suatu bilangan yang di random. Untuk biner bernilai 0
- Max berguna untuk nilai maksimal suatu bilangan yang di random. Untuk biner bernilai 1
- PopulationSize untuk banyak kromosom, nilai yang di set 50 untuk jumlah kromosom untuk 1 individu
- nGen di set 100 fungsinya untuk pergantian dari generasi 0 hingga 100
- TournamentSize digunakan untuk seleksi orang tua, dimana di set dengan nilai 6.
- CrossoverProbability untuk menentukan titik potong pada two-point crossover yang di set sebesar 0,6.
- MutationProbability untuk menentukan indeks bit kromosom yang dimutasi yang di set sebesar 0,1.

IV. Kesimpulan

Dari hasil yang telah diobservasi dapat ditarik kesimpulan, bahwa menggunakan Algoritma Genetika terdapat beberapa langkah yaitu desain kromosom dan pengdekodean, menentukan populasi, pemilihan orangtua, crossover dan mutasi, probabilitas operasi genetik, seleksi survivor, dan kriteria penghentian evolusi. Sehingga, dengan metode tersebut didapatkan gen terbaik yaitu kromosom 10011111, fitness 2.47, dan (X,Y) (0.8 , 1.0).

Hasil Running :

```
First Gen --- dengan Cromosome : 11111101 --- Nilai Fitness Tertinggi : 2.3985183082405594 --- (x, y) : (2.0, 0.7333333333333334)
gen ke- 1 --- dengan Cromosome : 10101111 --- Nilai Fitness Tertinggi : 2.454648713412841 --- (x, y) : (1.0, 1.0)
gen ke- 2 --- dengan Cromosome : 10101111 --- Nilai Fitness Tertinggi : 2.454648713412841 --- (x, y) : (1.0, 1.0)
gen ke- 3 --- dengan Cromosome : 10101111 --- Nilai Fitness Tertinggi : 2.454648713412841 --- (x, y) : (1.0, 1.0)
gen ke- 4 --- dengan Cromosome : 10101111 --- Nilai Fitness Tertinggi : 2.454648713412841 --- (x, y) : (1.0, 1.0)
gen ke- 5 --- dengan Cromosome : 10101111 --- Nilai Fitness Tertinggi : 2.454648713412841 --- (x, y) : (1.0, 1.0)
gen ke- 6 --- dengan Cromosome : 10101111 --- Nilai Fitness Tertinggi : 2.454648713412841 --- (x, y) : (1.0, 1.0)
gen ke- 7 --- dengan Cromosome : 10101111 --- Nilai Fitness Tertinggi : 2.454648713412841 --- (x, y) : (1.0, 1.0)
gen ke- 8 --- dengan Cromosome : 10101111 --- Nilai Fitness Tertinggi : 2.454648713412841 --- (x, y) : (1.0, 1.0)
gen ke- 9 --- dengan Cromosome : 10101111 --- Nilai Fitness Tertinggi : 2.454648713412841 --- (x, y) : (1.0, 1.0)
```

```
gen ke- 90 --- dengan Cromosome : 10011111 --- Nilai Fitness Tertinggi : 2.474940307297132 --- (x, y) : (0.8, 1.0)
gen ke- 91 --- dengan Cromosome : 10011111 --- Nilai Fitness Tertinggi : 2.474940307297132 --- (x, y) : (0.8, 1.0)
gen ke- 92 --- dengan Cromosome : 10011111 --- Nilai Fitness Tertinggi : 2.474940307297132 --- (x, y) : (0.8, 1.0)
gen ke- 93 --- dengan Cromosome : 10011111 --- Nilai Fitness Tertinggi : 2.474940307297132 --- (x, y) : (0.8, 1.0)
gen ke- 94 --- dengan Cromosome : 10011111 --- Nilai Fitness Tertinggi : 2.474940307297132 --- (x, y) : (0.8, 1.0)
gen ke- 95 --- dengan Cromosome : 10011111 --- Nilai Fitness Tertinggi : 2.474940307297132 --- (x, y) : (0.8, 1.0)
gen ke- 96 --- dengan Cromosome : 10011111 --- Nilai Fitness Tertinggi : 2.474940307297132 --- (x, y) : (0.8, 1.0)
gen ke- 97 --- dengan Cromosome : 10011111 --- Nilai Fitness Tertinggi : 2.474940307297132 --- (x, y) : (0.8, 1.0)
gen ke- 98 --- dengan Cromosome : 10011111 --- Nilai Fitness Tertinggi : 2.474940307297132 --- (x, y) : (0.8, 1.0)
gen ke- 99 --- dengan Cromosome : 10011111 --- Nilai Fitness Tertinggi : 2.474940307297132 --- (x, y) : (0.8, 1.0)
gen ke- 100 --- dengan Cromosome : 10011111 --- Nilai Fitness Tertinggi : 2.474940307297132 --- (x, y) : (0.8, 1.0)
================================ Gen Terbaik ===================================
Cromosome     :  10011111
Fitness       :  2.474940307297132
(X, Y)        :  (0.8, 1.0)
```

V. Link Presentasi

Berikut link presentasi Tugas Pemrograman 1 Pengantar Kecerdasan Buatan Kelompok 6 : https://youtu.be/4e6YzFKoFNs

References

suhartono, d. (2020, 07 29). *Optimasi dengan Genetic Algorithm Menggunakan Python*. From socs.binus.ac.id: https://socs.binus.ac.id/2020/07/29/optimasi-dengan-genetic-algorithm-menggunakan-python/

tang, s., zhu, q., yan, g., wu, m., & tian, y. (2002). EFFECTS OF GA ON THE INVERSION OF LINEAR AND NONLINEAR REMOTE SENSING MODELS. *ieeexplore*.