TUGAS PEMROGRAMAN 1
PENGANTAR KECERDASAN BUATAN

Oleh :

Diva Annisa Febecca (1301204302)
Johanes Raphael Nandaputra (1301204243)

PROGRAM STUDI S1 INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
BANDUNG
2022

DAFTAR ISI

BAB I

PENDAHULUAN

1.1. Latar Belakang

Genetic Algorithm (GA) atau disebut juga Algoritma Genetika adalah algoritma pencarian heuristik yang didasari pada pemikiran mengenai seleksi alam yang terjadi pada proses evolusi dan operasi genetika (Picek et al., 2013). Menurut Holland dalam bukunya yang berjudul *"Adaptation in Natural and Artificial Systems"* tujuan algoritma genetika adalah untuk mempelajari secara formal fenomena adaptasi seperti terjadi di alam dan mengembangkan cara-cara adaptasi di alam ke dalam sistem komputer. Dalam algoritma genetika, terdapat beberapa tahapan dalam suatu proses yang mendukung keberhasilan seperti pembentukan kromosom, menghitung nilai fitness tiap kromosom, seleksi kromosom, *crossover* dan mutasi kromosom.

1.2. Permasalahan

Permasalahan tugas pemrograman ini adalah kami ditugaskan untuk membuat algoritma genetika beserta analisis, desain, dan implementasi algoritma ke dalam suatu program komputer untuk mencari nilai x dan y sehingga diperoleh nilai minimum dari fungsi:

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

dengan domain x dan y, sebagai berikut:

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

dan fungsi fitness untuk memperoleh minimum fungsi objektif diatas, yaitu:

$$f = \frac{1}{h + a}$$

Selain itu, kami juga ditugaskan untuk melakukan analisis mengenai desain kromosom dan metode pengkodean, ukuran populasi, metode pemilihan orangtua, metode operasi genetik *(crossover* dan mutasi), probabilitas operasi genetik (Pc dan Pm), metode pergantian generasi (seleksi survivor), dan kriteria penghentian evolusi. Output yang diminta dari sistem adalah kromosom terbaik dengan nilai x dan y hasil decode kromosom terbaiknya.

BAB II

PEMBAHASAN

2.1. Desain Kromosom dan Metode Dekode

Desain kromosom yang kami gunakan adalah 8 bit, dimana representasi yang kami gunakan adalah representasi biner (0 atau 1). Desain ini membagi kromosom menjadi 2 bagian, dengan 4 bit pertama untuk x dan 4 bit terakhir untuk y, dimana 2 bagian ini akan dilakukan pendekodean untuk menemukan nilai fitnessnya menggunakan:

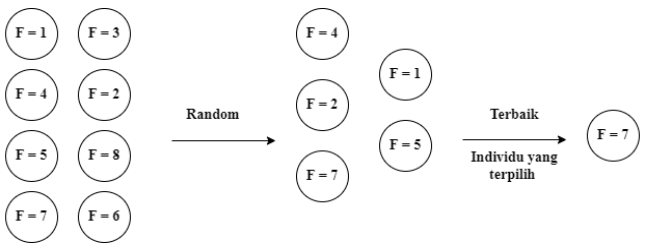
$$x = r_b + \frac{(r_a - r_b)}{\sum_{i=1}^{N} 2^{-i}} (g_1.2^{-1} + g_2.2^{-2} + ... + g_N.2^{-N})$$

2.2. Ukuran Populasi

Ukuran populasi adalah banyaknya kromosom dan gen di dalam satu kromosom yang terlibat selama proses pencarian (Taiwo et al.,2013). Dalam permasalahan ini ukuran populasi yang kami buat adalah sebanyak 50 kromosom per generasi.

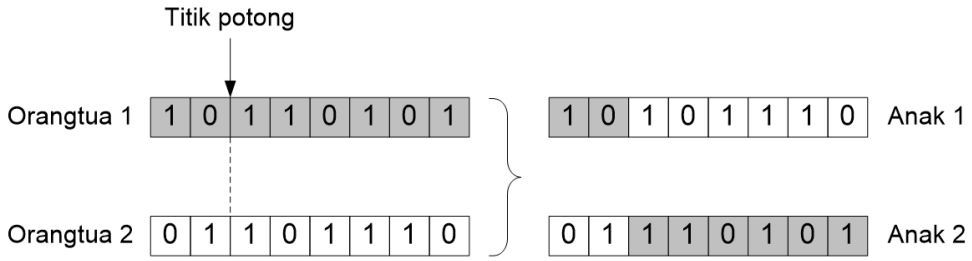
2.3. Metode Pemilihan Orang Tua

Tujuan proses seleksi adalah untuk memilih kromosom (individu) yang terbaik dalam populasi dengan harapan bahwa keturunan yang baru akan memiliki nilai fitness yang baik. (Mitchell, 1999). Pada tahap ini, kami menggunakan metode Tournament Selection karena berdasarkan penelitian yang dilakukan Razali & Geraghty (2011) menyatakan bahwa metode Tournament Selection lebih optimal dibandingkan dengan metode lainnya. Selain itu, seleksi ini dapat menemukan nilai fitness dengan lebih cepat dan konstan sehingga waktu yang dibutuhkan untuk konvergen lebih cepat. Metode ini akan menyeleksi kromosom terbaik dengan nilai fitness tertinggi untuk menjadi orang tua. Kami merandom 5 kromosom yang terdapat di populasi untuk dipertandingkan satu sama lain sehingga kami akan mendapat 1 kromosom terbaik dengan nilai fitness tertinggi.



2.4. Metode Operasi Genetik

Crossover adalah operator genetik yang menggabungkan dua kromosom (parent) untuk menghasilkan kromosom baru (*offspring*) dengan *probability crossover* (Pc). (Otman, 2011). Pada tahap *crossover* ini, kami akan menukar bit pada kromosom kedua orang tua yang terpilih dengan *Single Point* atau Rekombinasi Satu Titik, dimana kami akan merandom index yang akan dijadikan titik potongnya, lalu menukar potongan kromosom antar kedua orang tua menjadi anak hasil kedua potongan tersebut, seperti yang tertera pada gambar.



2.5. Probabilitas Operasi Genetik

Pada probabilitas operasi genetik, kami menetapkan nilai 0.7 untuk probabilitas crossover-nya dan nilai 0.05 untuk probabilitas mutasi-nya. Kami menetapkan probabilitas mutasinya kecil senilai 0.05 karena mutasi tidak selamanya menghasilkan kromosom yang bagus.

2.6. Metode Pergantian Generasi

Pada tahap pergantian generasi, kami menggunakan metode Generational Replacement, dimana metode ini akan menyimpan 2 kromosom *elite* di tiap generasi dan ditambahkan ke generasi berikutnya. Hal ini akan memastikan bahwa kualitas kromosom tidak akan menurun atau berkurang pada generasi berikutnya.

2.7. Kriteria Penghentian Evolusi

Pada tahap penghentian evolusi, kami menetapkan generasi maksimum sebanyak 50 generasi. Jadi, program algoritma genetika ini akan mengulang sebanyak 50 kali dengan tiap generasi akan menghasilkan 50 kromosom.

BAB III

KODE PROGRAM

3.1. Import Library

```
import random
from matplotlib import pyplot as plt
```

Untuk library yang kami gunakan adalah kami mengimport random dan menggunakan matplotlib. Random bertujuan untuk membantu pembuatan kromosom dimana akan merandom angka biner 0 atau 1 ke tiap bit kromosom. Matplotlib bertujuan untuk visualisasi data pertumbuhan nilai fitness.

3.2. Pembuatan Kromosom

```
def generate_chromosome(alel):
  chromosome = []
  for i in range(alel):
    chromosome.append(random.randint(0,1))
  return chromosome
```

Fungsi ini berguna untuk menghasilkan/membuat kromosom yang terdiri sebanyak 8 bit (alel), dimana tiap bit akan di random bilangan binary antara 0 atau 1. Fungsi ini akan mengembalikan list dengan panjang 8 dengan tiap bit atau indexnya akan berupa 0 atau 1, dimana 4 bit kiri untuk x dan 4 bit kanan untuk y.

3.3. Pembuatan Populasi

```
def generate_population(n):
  population = []
  for i in range(n):
    population.append(generate_chromosome(alel))
  return population
```

Fungsi ini berguna untuk membuat populasi dari kumpulan kromosom sebanyak ukuran populasi (n), dimana akan melakukan perulangan sebanyak n kali dan melakukan pembuatan kromosom. Fungsi ini akan mengembalikan populasi yang berisi 50 kromosom.

3.4. Decode Kromosom

```
def decode(chromosome):
  gen_x = chromosome[0:4]
  gen_y = chromosome[4:8]
  sum_gen = 2**(-1) + 2**(-2) + 2**(-3) + 2**(-4)

  x = x_min + ( ((x_max - x_min) / (sum_gen)) * ((gen_x[0] * 2**-1) + (gen_x[1] * 2**-2) + (gen_x[2] * 2**-3) + (gen_x[3] * 2**-4) ))
  y = y_min + ( ((y_max - y_min) / (sum_gen)) * ((gen_y[0] * 2**-1) + (gen_y[1] * 2**-2) + (gen_y[2] * 2**-3) + (gen_y[3] * 2**-4) ))

  return x,y
```

Fungsi ini berguna untuk mendekode kromosom, dimana gen X mengambil 4 bit pertama kromosom dan gen Y mengambil 4 bit terakhir kromosom. Rumus dekode ini menggunakan rumus yang terlampir pada BAB II Pembahasan di 2.1. Desain Kromosom dan Metode Dekode dan akan mengembalikan hasil dari dekode x dan y.

3.5. Minimasi Fungsi

```
def func(x,y):
  e = 2.718281828459045
  a = 0.01
  h = ((e**(x*1j)).imag + (e**(y*1j)).real)**2 / (x**2) + (y**2)
  return 1 / (h + a)
```

Fungsi ini berguna untuk melakuan minimasi fungsi dari hasil pendekodean x dan y, dengan rumus dari minimasi fungsi ini menggunakan f = 1 / (h + a), dimana :
h = (cos(x) + sin(y))^2 / (x^2) + (y^2) dan a = 0.01.
Fungsi ini akan mengembalikan nilai f = 1 / (h + a).

3.6. Nilai Fitness

```
def fitness_value(population):
  fitness = []
  for i in population:
    x,y = decode(i)
    fitness.append(func(x,y))
  return fitness
```

Fungsi ini berguna untuk menghitung nilai fitness dari kromosom. Dengan menggunakan bantuan dari beberapa fungsi lain, decode kromosom dan minimasi fungsi, dengan melakukan perulangan untuk mengecek tiap kromosom yang ada di populasi, lalu melakukan pendekodean kromosom untuk nilai x dan y, dan melakukan minimasi fungsi dari hasil pendekodean tiap kromosom. Hasil minimasi fungsi ini akan disimpan dalam list fitness dan mereturn fitness setelah selesai perulangan.

3.7. Pemilihan Orang Tua

```
def tournament_selection(population, fitness, n, k):
  nomine = random.sample(range(n), k);
  print(nomine)
  winner = None

  for i in nomine:
    print(population[i], ":", fitness[i])
    if (winner == None) or (fitness[i] > fitness[winner]):
      winner = i
  return population[winner]
```

Fungsi ini berguna untuk mengambil kromosom yang kedepannya akan dijadikan orangtua untuk crossover dan menghasilkan anak (kromosom baru). Kami menggunakan metode tournament selection dengan tournament size nya adalah 5, ini berarti kami akan merandom 5 kromosom untuk "dipertandingkan" satu sama lain dengan cara mencari nilai fitness tertinggi dari 5 kromosom tersebut. Setelah ditemukan, akan mereturn kromosom pemenangnya.

3.8. Crossover dan Rekombinasi

```
def crossover(p1, p2, pC):
  if random.random() <= pC:
    point = random.randint(1, alel - 1)
    print("Titik Potong :", point)
    child1 = (p1[0:point] + p2[point:alel])
    child2 = (p2[0:point] + p1[point:alel])
    return child1,child2
  return p1,p2
```

Fungsi ini berguna untuk men-crossover kedua orang tua yang sudah kita seleksi dengan cara merandom 1 - 7 sebagai titik potongnya. Child1 akan mengambil dari bit awal hingga bit titik potong dari parent1 dan bit titik potong hingga bit terakhir dari parent2. Begitu juga dengan child2, dimana ia akan mengambil bit awal hingga bit titik potong dari parent2 dan bit titik potong hingga bit terakhir dari parent1. Akan mengembalikan child1 dan child2 jika probabilitas crossover lebih dari angka random, mengembalikan parent1 dan parent2 jika tidak.

3.9. Mutasi

```
def mutation(chromosome, pM):
  for i in range(alel):
    if random.random() <= pM:
      if chromosome[i] == 0:
        chromosome[i] = 1
      else:
        chromosome[i] = 0
  return chromosome
```

Fungsi ini berguna untuk memutasi kromosom, yaitu dari tiap bit kromosom jika probabilitas kromosom lebih besar dari angka random, maka akan membalikkan nilai binernya (0 menjadi 1, 1 menjadi 0). Dan akan mengembalikan kromosom hasil mutasi tersebut.

3.10. Elitisme

```
def elitism(population, fitness):
  max = 2
  elite = []

  bestChromosome = [(fitness[i], population[i]) for i in range(len(population))]
  grade = sorted(bestChromosome, key=lambda x: x[0], reverse=True)

  for i in range(max):
    elite.append(grade[i][1])
  return elite
```

Fungsi ini berguna untuk menyimpan 2 kromosom terbaik atau *elite* di tiap generasi, dengan cara mencari kromosom dengan nilai fitness tertinggi. Hal ini berguna agar kualitas kromosom di generasi-generasi selanjutnya tidak berkurang. Fungsi ini akan mengembalikan list elite yang menyimpan 2 kromosom terbaik.

3.11. Pergantian Generasi

```
def generational_replacement(n, maxGen, pC, pM, k):
  population = generate_population(n)
  for gen in range(maxGen):
    print("Gen", gen+1)
    fitness = fitness_value(population)

    for i in range(len(population)):
      print("Kromosom", i+1, ":",  population[i], "   |  Nilai Fitness:", fitness[i])
    print("============================================")
    newPopulation = elitism(population, fitness)

    # Untuk Grafik
    fit.append(fitness_value(newPopulation))

    print("Hasil Elitisme")
    for elit in newPopulation:
      x,y = decode(elit)
      print(elit, ":", func(x,y))
    print("============================================")
    print("Kromosom Ditambahkan :", newPopulation)
    print("Jumlah Populasi Baru :", len(newPopulation))
    print("============================================")

    while len(newPopulation) < n:
      print("Memulai Tournament Selection")
      parent1 = tournament_selection(population, fitness, n, k)
      print("Parent 1:", parent1)
      parent2 = tournament_selection(population, fitness, n, k)
      while parent1 == parent2:
        print("Parent 2:", parent2)
        print("Parent Sama, Melakukan Pemilihan Parent Baru")
        parent2 = tournament_selection(population, fitness, n, k)
      print("Parent 2:", parent2)
      print("============================================")
      print("Memulai Crossover Parent")
      child1, child2 = crossover(parent1, parent2, pC)
      print("Child 1:", child1)
      print("Child 2:", child2)
      print("============================================")
      print("Memulai Mutasi Child")
      child1 = mutation(child1, pM)
      child2 = mutation(child2, pM)
      print("Hasil Mutasi Child 1:" , child1)
      print("Hasil Mutasi Child 2:", child2)
      print("============================================")
      newPopulation.extend([child1, child2])
      print("Kromosom Ditambahkan :", child1, child2)
      print("Jumlah Populasi Baru :", len(newPopulation))
      print("============================================")
    population = newPopulation

  highest_fitness = max(fitness)
  best_chromosome = fitness.index(highest_fitness)
  return population[best_chromosome]
```

Fungsi ini merupakan pergantian generasi-survivor selection yang menggunakan metode general replacement, dimana populasi yang baru akan digantikan dengan hasil elitisme dari generasi sebelumnya dan hasil mutasi dari child1 dan child2. Fungsi ini akan mengembalikan kromosom terbaik di generasi yang terakhir.

3.12. Program Utama

```
x_min = -5
x_max = 5
y_min = -5
y_max = 5

population_size = 50
crossover_probability = 0.7
mutation_probability = 0.05
alel = 8
tournament_size = 5
max_generation = 50
fit = []

best = generational_replacement(population_size, max_generation, crossover_probability, mutation_probability, tournament_size)
x,y = decode(best)
fitness_value = func(x,y)
print("==================== SOLUSI TERBAIK ====================")
print("KROMOSOM :", best)
print("X :", x)
print("Y :", y)
print("NILAI FITNESS :", fitness_value)
```

Pada program utama, kami mendefinisikan domain dari x dan y, ukuran populasi, probabilitas crossover, probabilitas mutasi, alel (panjang kromosom), ukurang tournament, generasi maksimum. Lalu kami menyimpan hasil dari fungsi pergantian generasi (generational_replacement) ke dalam variabel “best”. Hasil dari variabel best merupakan hasil kromosom terbaik setelah melakukan pergantian generasi, dan datanya yang berupa, kromosom, nilai dekode x dan y, dan nilai fitnessnya akan ditampilkan ke layar.

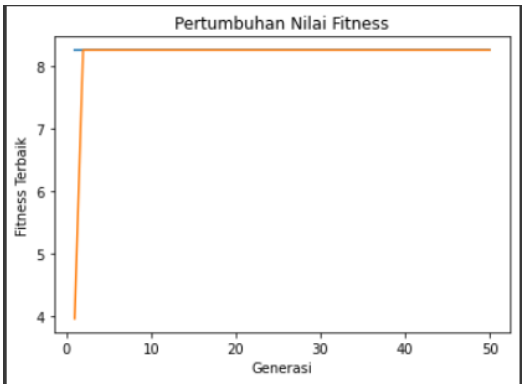
BAB IV

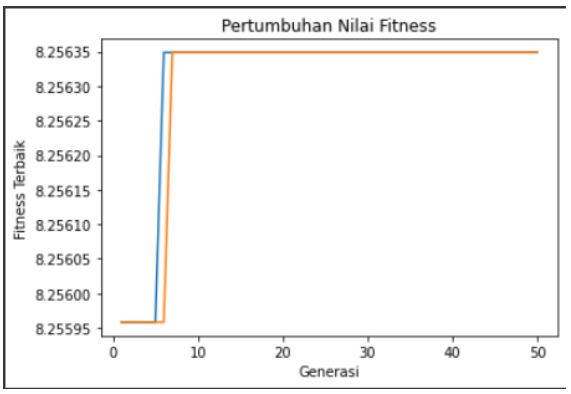
HASIL PERCOBAAN DAN KESIMPULAN

4.1. Hasil Percobaan

Berdasarkan hasil percobaan dan uji coba kami dalam menjalankan program, didapatkan grafik pertumbuhan nilai fitness tiap generasi sebagai berikut:

```
==================== SOLUSI TERBAIK ====================
KROMOSOM : [1, 1, 1, 1, 1, 0, 0, 0]
X : 5.0
Y : 0.33333333333333304
NILAI FITNESS : 8.256348759563485
```





4.2. Hasil Analisis

Dapat diketahui dengan hasil grafik diatas, kami dapat menganalisis bahwa pertumbuhan nilai fitness terbaik, dapat kami temukan dalam 10 generasi pertama. Dan pada generasi selanjutnya hingga generasi 50, tidak mendapatkan pengurangan atau penambahan nilai fitness lagi. Hal ini dapat disebabkan karena function elitism yang berguna untuk menyimpan kromosom terbaik di generasinya dan di teruskan ke generasi selanjutnya agar tidak terjadi penurunan kualitas nilai fitness kromosom.

4.3 . Kesimpulan

Berdasarkan hasil percobaan program *Genetic Algorithm* (GA) dengan fungsi:

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

dengan batasan :

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

dapat disimpulkan bahwa algoritma genetika dapat menghasilkan nilai yang optimal dalam mencari nilai minimum fungsi objektif. Hal ini dapat dibuktikan dengan tiap pergantian generasi akan menyimpan nilai kromosom terbaiknya hingga ke generasi terakhirnya.

BAB V

DAFTAR PUSTAKA

Mitchel, Mielanie. 1999. An Introduction to Genetic Algorithm. MIT Press: Massachusets

Picek, Stjepan, Jakobovic, Domagoj and Gloub, Marin. 2013. On the Recombination Operator in The Real-Code Genetic Algorithms, 2013 IEEE Congress on Evolutionary Computation, pp. 3103-3110

Razali, N. M., & Geraghty, J. (2011). Genetic Algorithm Performance with Different Selection Strategies in Solving TSP, Proceedings of the World Congress on Engineering 2011 Vol II WCE 2011, July 6 - 8, 2011, London, U.K

Taiwo, Oloruntoyin Sefiu, Olukehinde Olutosin Mayowa and Kolapo Bukola Ruka. 2013. Application of Genetic Algorithm to Solve Traveling Salesman Problem. International Journal of Advance Research (IJOAR)1(4): 27-46

BAB VI

LAMPIRAN

Nama	NIM	Peran
Johanes Raphael Nandaputra	1301204243	Coding, Video, Laporan
Diva Annisa Febecca	1301204302	Coding, Video, Laporan

Link Souce Code :
https://colab.research.google.com/drive/1g6yJoDNpt6y3TY7yMv9bVVhwnKUwbQIZ?hl=id#scrollTo=kQI8I01RtcH1

Link Video :
https://youtu.be/ksi0RsY6HC4