LAPORAN PENGANTAR KECERDASAN BUATAN

Disusun untuk memenuhi salah satu tugas mata kuliah Pengantar Kecerdasan Buatan

Oleh Kelompok 15:

Mirai Tsuchiya - 1301203555 - IF-44-08

Khalilullah Al Faath - 1301204376 - IF-44-08

FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
2022

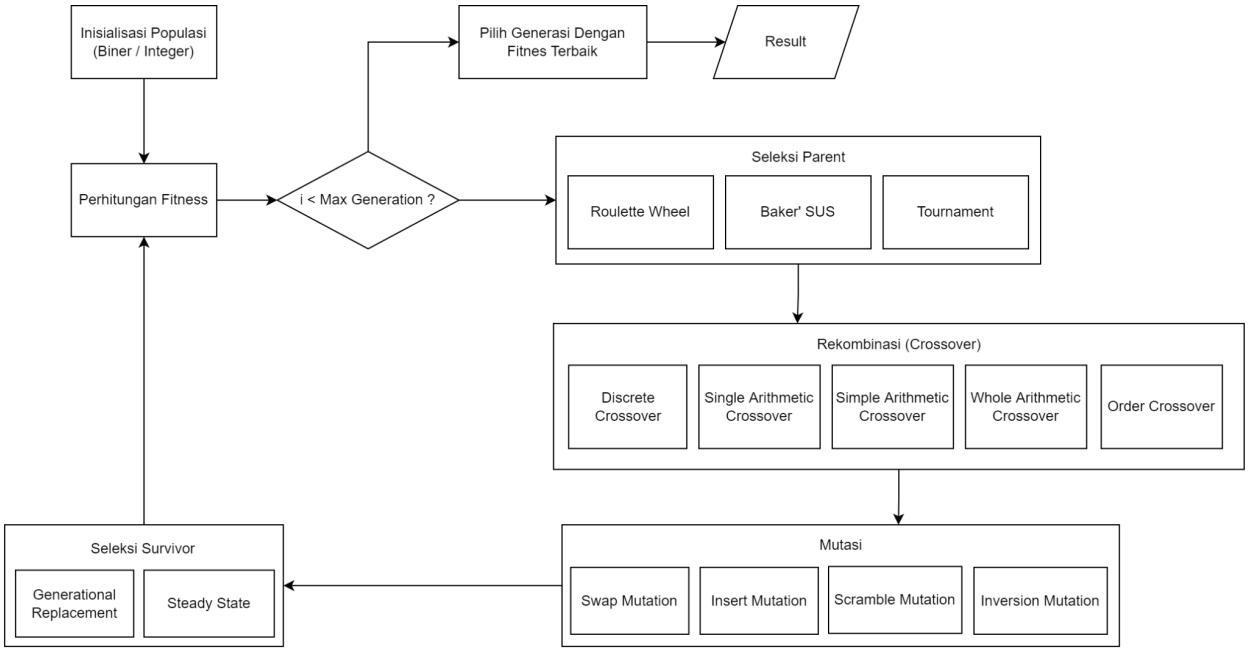
Daftar Isi

1. Tinjauan Pustaka

1.1. Definisi Algoritma Genetika

Algoritma genetika adalah suatu proses optimasi yang dikembangkan berdasarkan prinsip genetika dan proses seleksi alamiah (Haupt & Haupt 2004: 22).

1.2. Alur Algoritma Genetika



2. Pembahasan

2.1. Istilah-istilah penting

Kromosom	: Solusi yang mungkin
Populasi	: Kumpulan Individu
Kromosom	: Cetak biru dari sebuah individu
Fitness	: Kualitas dari sebuah solusi

2.2. Parameter-parameter yang digunakan

- Desain kromosom	: Kumpulan array dengan representasi data binary
- Probabilitas crossover	: 0.8 (80%)
- Probabilitas mutasi	: 0.05 (5%)
- Total populasi/ individu	: 10
- Panjang kromosom	: 10
- Total generasi	: 100

2.3. Fungsi yang dioptimasi

Akan dilakukan analisis dan desain algoritma Genetics Algorithm (GA) serta mengimplementasikannya ke dalam suatu program komputer untuk mencari nilai *x* dan *y* sehingga diperoleh nilai minimum dari fungsi:

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

Untuk x, di mana $-5 \leq X \leq 5$ dan y, di mana $-5 \leq Y \leq 5$. Limit tersebut disimpan dalam variabel limitX dan limitY sebagai berikut.

```
#Initializing domain of x and y based on the assignment
limitX = [-5,5]
limitY = [-5,5]
```

Untuk fungsi yang akan diminimasi, implementasinya dalam Python adalah sebagai berikut.

```
[ ] #initializing the function that will be minimized using genetics algorithm
    def h(x,y):
      return (cos(x)+sin(y))**2 / ((x**2)+(y**2))
```

3. Representasi Data dengan Biner

Pada pembuatan algoritma genetika, kelompok kami menggunakan representasi data dalam bentuk biner untuk pencarian nilai minimum pada fungsi yang diberikan.

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

di mana :
x = representasi kromosom dalam biner
r_b = batas bawah
r_a = batas atas
g = gen

4. Pemrosesan Kode Algoritma Genetika

Dalam pengerjaan algoritma genetika, proses awal adalah pembuatan populasi beserta kromosomnya. Kemudian dilakukan proses decode serta menghitung nilai fitness dari tiap kromosom. Lalu dilakukan proses seleksi pemilihan orang tua dengan metode roulette wheel. Setelah itu, dilakukan proses crossover orang tua dan mutasi anak. Proses diatas dilakukan secara berulang-ulang sebanyak n-generasi dan mencari kromosom terbaik dari semua generasi dan proses algoritma genetika akan berhenti.

4.1. Proses Generate Populasi Awal beserta Kromosom

Fungsi generatepopulation dibawah digunakan untuk membuat populasi beserta kromosomnya. Kelompok kami membuat sebanyak sepuluh individu dengan sepuluh kromosom tiap individu. Kromosom bernilai acak dari 0 sampai 1. Array pertama generate array kromosom sebanyak chromosomeLength yg dimasukkan pada main. Array kedua menggenerate populasi sebanyak populationSize yang sudah diset dari awal.

```
[36] #function to generate population including its chromosomes
     def generatePopulation(populationSize,chromosomeLength):
       #binary representation
       return ([[random.randint(0,1) for _ in range(chromosomeLength)] for _ in range(populationSize)])
```

4.2. Pencarian Nilai Fitness

```
[39] #this function will return fitness value of a chromosome
     def fitnessFunction(x1,x2):
       #a is a very small number to avoid zero division
       a = 0.001

       #fitness = 1/(h+a)
       return 1/(a+h(x1,x2))
```

Fungsi di atas 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 ada pada soal dan a adalah bilangan yang sangat kecil.

4.3. Proses Seleksi Pemilihan Orang Tua

```
def parentRouletteWheel(population, fitness, fitnessTotal):
  r = random.random()
  i = 0
  while r > 0:
    r -= fitness[i]/fitnessTotal
    i += 1
    if i == len(population) - 1:
      break
  return population[i]
```

```
#this function will return a parent to generate a new generation in the mating pool
def parentStochasticRouletteWheel(population,maxFitness):
  while(True):
    individu = random.uniform(0,1)*individuTotal
    chromosomeIndividu = population[int(individu)]

    firstGenotype,secondGenotype = splitChromosome(chromosomeIndividu)
    x1 = decode(firstGenotype,limitX)
    x2 = decode(secondGenotype,limitY)

    r = random.uniform(0,1)
    if r < (fitnessFunction(x1,x2)/maxFitness):
      return chromosomeIndividu
```

Kami pada bagian ini memberikan dua alternatif pemilihan orang tua, yaitu metode roulette wheel dan stochastic roulette wheel.

Fungsi parentRouletteWheel yaitu fungsi yang digunakan untuk 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. Secara sederhana, Roulette Wheel bisa digambarkan seperti permainan roda roulette pada permainan judi, yang menggunakan sebuah jarum.

Karena metode pemilihan orang tua sebelumnya memiliki kelemahan, yaitu bisa saja kromosom dengan nilai tertinggi yang berada di paling belakang tidak terpilih, maka dibuatlah fungsi baru, yaitu stokastik roulette wheel yang mana kami memilih secara random satu individu i dengan probabilitas uniform (1/N). Kemudian orang tua dipilih dengan probabilitas fi/fmax, yang mana fmax adalah fitness maksimal pada suatu populasi.

4.4. Proses Crossover Orang Tua

```
#recombination using single point crossover
def recombination(firstParent,secondParent):
  r = random.random() #random number [0,1]
  firstChild,secondChild = [],[] #Every two parents will generate two children

  if r < recombinationProbability:
    point = random.randint(1,len(firstParent)-1) #select random point from index 1 to the second last gen in a chromosome
    firstChild = firstParent[:point] + secondParent[point:]
    secondChild = secondParent[:point] + firstParent[point:]
  else: #if the random r bigger than recombination probability, the children will be same as their parents
    firstChild = firstParent
    secondChild = secondParent

  return firstChild,secondChild
```

Pada rekombinasi satu titik, pilih satu titik potong secara acak pada posisi antara 1 hingga $G - 1$, di mana G adalah jumlah gen dalam kromosom. Misalnya, terpilih titik potongnya adalah posisi T. Selanjutnya, buat Anak 1 dengan gen 1 sampai T diambil dari orangtua 1 dan gen T+1 sampai G diambil dari orangtua 2. Sebaliknya, Anak 2 mendapatkan gen 1 sampai T dari orangtua 2 dan gen T+1 sampai G dari orangtua 1

4.5. Proses Mutasi Anak

Fungsi mutation di bawah adalah untuk mengganti nilai kromosom pada suatu individu (anak) secara acak dengan menggunakan metode switching. Mutasi untuk representasi biner dilakukan dengan cara yang sederhana. 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 Pm, maka gen pada posisi tersebut dimutasi. Jika sebaliknya, maka gen tersebut tidak dimutasi. Mutasi dilakukan dengan cara membalik nilai binernya. Gen yang bernilai 1 dimutasi menjadi 0. Sedangkan gen yang bernilai 0 dimutasi menjadi 1.

```
#mutation by simple replacement, switching 0 to 1 and 1 to 0
def mutation(firstChild,secondChild):
  for i in range(len(firstChild)): #the mutation loop will occur for every genes in a chromosome
    r = random.random() #random number between 0 and 1
    if r < mutationProbability:
      if firstChild[i] == 1:
        firstChild[i] = 0
      else:
        firstChild[i] = 1

    r = random.random()
    if r < mutationProbability:
      if secondChild[i] == 1:
        secondChild[i] = 0
      else:
        secondChild[i] = 1

  return firstChild,secondChild
```

4.6. Proses Seleksi Survivor

```
def elitism(population,bestChromosomeGeneration,badChromosome,totalFitness):
  if bestChromosomeGeneration[1] > badChromosome[0] and (bestChromosomeGeneration[0] not in population):
    population[badChromosome[2]] = bestChromosomeGeneration[0]
    totalFitness = (totalFitness - badChromosome[0]) + bestChromosomeGeneration[1]

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

  return population,totalFitness
```

```
population = generatePopulation(individuTotal,chromosomeLength)
print("First Population: ",population)

bestChromosomeGeneration = []

for generationTotal in range(generation):
  #these will be reset for every generation
  chromosomeData, bestChromosome, badChromosome, fitnessData, newPopulation = [],[],[],[],[]
  totalFitness,countChromosome,index = 0,99999,0

  print("\n==================================================")
  print("Generasi ke-",generationTotal+1)
  print("==================================================") #
  for i,chromosome in enumerate(population):
    genotypeA,genotypeB = splitChromosome(chromosome)
    x1 = decode(genotypeA,limitX)
    x2 = decode(genotypeB,limitY)

    fitnessValue = fitnessFunction(x1,x2)
#this list will store every fitness value of a generation
    fitnessData.append(fitnessValue)
#this variable will store total fitness of a generation. This variable will be used in parent selection
    totalFitness += fitnessValue

    #find the smallest fitness value from a generation (the most bad chromosome)
    if generationTotal != 0 and fitnessValue < countChromosome:
      countChromosome = fitnessValue
      badChromosome = [fitnessValue,chromosome,i]
```

```
#selecting the best chromosome
bestChromosome = bestChromosomeSelection(population)

print("Best Chromosome      : ", bestChromosome[0])
print("Best Fitness         : ", bestChromosome[1])
print("Best value x         : ", bestChromosome[2])
print("Best value y         : ", bestChromosome[3])
print("Best value function h: ", h(bestChromosome[2],bestChromosome[3]))

#elitism
if generationTotal != 0:
  mostBest = sorted(bestChromosomeGeneration,key=lambda x:x[1],reverse=True)[0]
  population,totalFitness = elitism(population,mostBest,badChromosome,totalFitness)

#best after elitism
if bestChromosome[1] <  bestChromosomeSelection(population)[1]:
  bestChromosome = bestChromosomeSelection(population)
  print("Best Chromosome      : ", bestChromosome[0])
  print("Best Fitness         : ", bestChromosome[1])
  print("Best value x         : ", bestChromosome[2])
  print("Best value y         : ", bestChromosome[3])
  print("Best value function h: ", h(bestChromosome[2],bestChromosome[3]))


#best chromosome for all generation
bestChromosomeGeneration.append(bestChromosome)
```

```
 #looping for parent selection, recombination, and mutation
 if generationTotal != generation-1:
   for i in range(individuTotal // 2):
     #firstParent = parentRouletteWheel(population,fitnessData,totalFitness)
     #secondParent = parentRouletteWheel(population,fitnessData,totalFitness)
     firstParent = parentStochasticRouletteWheel(population,bestChromosome[1])
     secondParent = parentStochasticRouletteWheel(population,bestChromosome[1])
     while (firstParent == secondParent):
       secondParent = parentStochasticRouletteWheel(population,bestChromosome[1])
     firstChild,secondChild = recombination(firstParent,secondParent)
     firstChild,secondChild = mutation(firstChild,secondChild)
     newPopulation.append(firstChild)
     newPopulation.append(secondChild)

   population = newPopulation
print("\n==================================")
print("Best Chromosome      : ", mostBest[0])
print("Best Fitness         : ", mostBest[1])
print("Best value x         : ", mostBest[2])
print("Best value y         : ", mostBest[3])
print("Best value function h: ", h(mostBest[2],mostBest[3]))
print("==================================")
```

Pada generational model, suatu populasi berukuran N kromosom pada suatu generasi diganti dengan N kromosom baru pada generasi berikutnya. Prosesnya dimulai dari seleksi orangtua, yaitu memilih N kromosom untuk diletakkan ke dalam *Mating Pool*. Kemudian, N kromosom orangtua di dalam *Mating Pool* dipasangkan secara acak sehingga dihasilkan $N/2$ pasangan orangtua. Selanjutnya, setiap pasangan direkombinasi berdasarkan probabilitas P_c yang ditentukan user. Hasil rekombinasi kemudian dimutasi berdasarkan probabilitas P_m yang juga ditentukan oleh user. Kedua proses ini menghasilkan N kromosom baru. N kromosom baru ini menggantikan N kromosom lama tanpa memperhatikan nilai-nilai *fitness*-nya.

5. Hasil Output

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

```
First Population:  [[1, 0, 1, 1, 1, 1, 1, 0, 0, 1], [0, 1, 1, 1, 1, 0, 1, 1, 0, 0],

================================================
Generasi ke- 1
================================================
Best Chromosome      :  [0, 0, 1, 0, 0, 1, 0, 1, 0, 1]
Best Fitness         :  475.9355799725695
Best value x         :  -3.709677419354839
Best value y         :  1.774193548387097
Best value function h:  0.0011011246943496745

================================================
Generasi ke- 2
================================================
Best Chromosome      :  [0, 0, 1, 0, 0, 1, 0, 0, 1, 1]
Best Fitness         :  801.2814543899267
Best value x         :  -3.709677419354839
Best value y         :  1.129032258064516
Best value function h:  0.0002480009296625641

================================================
Generasi ke- 3
================================================
Best Chromosome      :  [0, 0, 1, 0, 1, 1, 0, 1, 0, 1]
Best Fitness         :  994.0293013466509
Best value x         :  -3.3870967741935485
Best value y         :  1.774193548387097
Best value function h:  6.006562025143884e-06
```

```
================================================
Generasi ke- 99
================================================
Best Chromosome      :  [0, 0, 1, 0, 1, 0, 0, 0, 0, 1]
Best Fitness         :  974.7814593987385
Best value x         :  -3.3870967741935485
Best value y         :  -4.67741935483871
Best value function h:  2.5870968675190758e-05

Elitism Process
Chromosome 6: [0, 1, 1, 0, 1, 0, 0, 0, 0, 1],with fitness value =: 7.812829958975118
changed to [0, 0, 1, 1, 0, 0, 0, 1, 0, 1],with fitness value = :999.8222852169366


================================================
Generasi ke- 100
================================================
Best Chromosome      :  [0, 0, 1, 0, 1, 0, 0, 0, 0, 0]
Best Fitness         :  996.6388558904632
Best value x         :  -3.3870967741935485
Best value y         :  -5.0
Best value function h:  3.3724794991397128e-06

Elitism Process
Chromosome 9: [0, 0, 1, 0, 1, 0, 0, 1, 0, 1],with fitness value =: 41.609752693338685
changed to [0, 0, 1, 1, 0, 0, 0, 1, 0, 1],with fitness value = :999.8222852169366
```

5.2. Penentuan Kromosom Terbaik dan Proses Berhentinya

Best Chromosome : [0, 0, 1, 1, 0, 0, 0, 1, 0, 1]
Best Fitness : 999.8222852169366
Best value x : -3.064516129032258
Best value y : -4.67741935483871
Best value function h: 1.777463712212467e-07

```
====================================
Best Chromosome      :  [0, 0, 1, 1, 0, 0, 0, 1, 0, 1]
Best Fitness         :  999.8222852169366
Best value x         :  -3.064516129032258
Best value y         :  -4.67741935483871
Best value function h:  1.777463712212467e-07
====================================
```

Proses berhenti terjadi apabila generation >= 100. Maka akan didapatkan best chromosome, phenotype x, phenotype Y, fitness value, dan nilai fungsi.

6. Kesimpulan

Nilai h minimum global dari fungsi $h(x,y) = \frac{(cos\ x + sin\ y)^2}{x^2+y^2}$ dengan x = -3.064516129032258 dan y = -4.67741935483871 adalah 1.777463712212467e-07.

```
====================================
Best Chromosome      :  [0, 0, 1, 1, 0, 0, 0, 1, 0, 1]
Best Fitness         :  999.8222852169366
Best value x         :  -3.064516129032258
Best value y         :  -4.67741935483871
Best value function h:  1.777463712212467e-07
====================================
```

Kromosom terbaik tersebut pertama kali muncul di generasi ke-50 dengan nilai fitness = 999.8222852169366

```
================================================
Generasi ke- 50
================================================
Best Chromosome      :  [0, 0, 1, 1, 0, 0, 0, 0, 0, 1]
Best Fitness         :  999.8222852169366
Best value x         :  -3.064516129032258
Best value y         :  -4.67741935483871
Best value function h:  1.777463712212467e-07

Elitism Process
Chromosome 1: [1, 0, 1, 0, 0, 1, 0, 0, 1, 0],with fitness value =: 3.8860706018384543
changed to [1, 0, 1, 1, 0, 1, 1, 1, 0, 0],with fitness value = :998.522354823782
```

Link Presentasi

https://youtu.be/_L6HJ5a4pnY

Daftar Pustaka

Dewi, Y. A., & Fitriana Yuli, S. (2007). IMPLEMENTASI ALGORITMA GENETIKA DENGAN VARIASI SELEKSI DALAM PENYELESAIAN CAPACITATED VEHICLE ROUTING PROBLEM WITH TIME WINDOWS (CVRPTW) UNTUK OPTIMASI RUTE PENDISTRIBUSIAN RASKIN DI KOTA YOGYAKARTA. *UNY Journal*, 10–41. http://eprints.uny.ac.id/48910/