**LAPORAN TUGAS BESAR KECERDASAN BUATAN**

**Amalul Anugrah Amin 1103213001, Faraday Barr Fatahillah 110321302, Purwantoro Aji Nugroho 11032130023**

1. **Deskripsi Masalah**

Kami mengimplementasikan Algoritma Genetika dalam Python untuk menyelesaikan *Knapsack Problem* yang kami definisikan sendiri. Dalam masalah ini, Bobe ingin membawa barang-barang penting ke kantor dalam salah satu dari tiga tas dengan batasan berat masing-masing 3kg, 4kg, dan 5kg. Barang-barang yang tersedia memiliki berat dan nilai tertentu. Tujuan algoritma ini adalah memilih kombinasi barang yang dapat dimasukkan ke setiap tas dengan total berat maksimum. Barang-barang dapat dilihat di bagian Kasus Pengujian.

Pengambilan hasil terbaik (*Best Individu*) diambil dari total dari prioritas barang yang dimiliki oleh salah satu kromosom yang telah dibuat.

1. **Perancangan Algoritma Genetik**

Urutan pengimplementasian Algoritma Genetika terhadap permasalahan di atas serta penjelasan singkat terkait setiap bagian adalah sebagai berikut,

1. *Initialize Population*, menentukan populasi pertama dengan cara membuat populasi secara *random*. Populasi ditentukan sebanyak 4 kromosom.
2. *Fitness Calculation*, memvalidasi kromosom dan menentukan individu terbaik. Kromosom yang tidak valid akan dimodifikasi agar memenuhi kondisi kasus
3. *Parent Selection*, menentukan orang tua dari populasi selanjutnya dan menampilkan *cumulative fitness, relative fitness,* dan area dari kromosom. Setelah itu akan menentukan testing untuk kromosom parent.
4. *Parent Gen Crossover*, Bagian ini akan melakukan rekombinasi terhadap Parent yang sudah ditentukan dari bagian sebelumnya. Setiap parent relationship akan ditentukan probabilitas yang digunakan untuk menentukan rekombinasi dari parent tersebut dan juga probabilitas untuk rekombinasi. Apabila probabilitas rekombinasi *parent relationship* lebih kecil dari probabilitas rekombinasi maka akan terjadi rekombinasi. Selain itu, ditentukan juga titik rekombinasinya.
5. *Gen Mutation*, Bagian ini akan melakukan mutasi gen ke setiap kromosom yang ada dengan probabilitas mutasi gen setiap kromosom dan probabilitas untuk mutasi. Apabila probabilitas mutasi gen kromosom lebih kecil dari probabilitas mutasi maka akan terjadi mutasi. Selain itu, ditentukan juga titik mutasi gen nya.
6. *Loop back to Fitness Calculation* (tergantung input user), Bagian ini terdapat di program utama yang merupakan inputan untuk berapa banyak tes yang akan dilakukan serta kasus (berat maksimum tas).
7. **Kasus Pengujian**

Pada Pengujian ini, kita akan menggunakan dataset berikut ini :

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Items | Weight(g) | Priority |
| B1 | Laptop Charger | 300 | 8 |
| B2 | Laptop | 1400 | 9 |
| B3 | Power Bank | 300 | 6 |
| B4 | Phone Charger Pouch | 100 | 7 |
| B5 | ANC Headset | 300 | 5 |
| B6 | Earbuds | 100 | 4 |
| B7 | Wireless Mouse | 100 | 7 |
| B8 | Flash Drive | 10 | 6 |
| B9 | Keyboard | 750 | 4 |
| B10 | Notebook | 50 | 2 |
| B11 | Pencil Case | 100 | 2 |
| B12 | Water Bottle | 1500 | 5 |
| B13 | Umbrella | 230 | 6 |
| B14 | USB Hub | 50 | 5 |
| B15 | Medicinal Pouch | 150 | 6 |
| B16 | Sunscreen Bottle | 100 | 2 |

1. **Kasus Pengujian 1**
   1. **Pengujian Dengan Program**

Pengujian dilakukan dengan menjalankan algoritma genetika sebanyak 5 iterasi. Dengan maksimal berat yang dibawa adalah 4kg.Berikut adalah hasil pengujiannya:

* Iterations: 5
* Maximum Weight: 5000g
* All Chromosomes Priority Value: [57, 54, 54, 54, 54]
* Best Individu Overall: 57
* Best Individu Chromosome: [0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0]

Items Carried:

1. Laptop

2. Power Bank

3. ANC Headset

4. Earbuds

5. Wireless Mouse

6. Flash Drive

7. Keyboard

8. Notebook

9. Pencil Case

10. Umbrella

11. Medicinal Pouch

* 1. **Pengujian dengan Perhitungan manual**

1. Inisialisasi Populasi dan perhitungan Fitness

Menghitung berat dan nilai fitness

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Code | Items | Weight(g) | Priority | K0 | K1 | K2 | K3 |
| B1 | Laptop Charger | 300 | 8 | 0 | 1 | 0 | 1 |
| B2 | Laptop | 1400 | 9 | 0 | 1 | 1 | 0 |
| B3 | Power Bank | 300 | 6 | 0 | 0 | 1 | 0 |
| B4 | Phone Charger Pouch | 100 | 7 | 1 | 1 | 0 | 0 |
| B5 | ANC Headset | 300 | 5 | 1 | 0 | 1 | 1 |
| B6 | Earbuds | 100 | 4 | 1 | 0 | 1 | 1 |
| B7 | Wireless Mouse | 100 | 7 | 0 | 0 | 1 | 1 |
| B8 | Flash Drive | 10 | 6 | 1 | 1 | 1 | 0 |
| B9 | Keyboard | 750 | 4 | 1 | 1 | 1 | 0 |
| B10 | Notebook | 50 | 2 | 1 | 0 | 1 | 1 |
| B11 | Pencil Case | 100 | 2 | 0 | 1 | 1 | 0 |
| B12 | Water Bottle | 1500 | 5 | 1 | 1 | 0 | 1 |
| B13 | Umbrella | 230 | 6 | 1 | 0 | 1 | 0 |
| B14 | USB Hub | 50 | 5 | 0 | 0 | 0 | 0 |
| B15 | Medicinal Pouch | 150 | 6 | 0 | 1 | 1 | 1 |
| B16 | Sunscreen Bottle | 100 | 2 | 0 | 0 | 0 | 1 |
| Total berat (kg) = 13,440 | | | | 3,040 | 4,310 | 3,490 | 2,600 |
| Total fitness = 182 | | | | 39 | 47 | 57 | 39 |

Best individu setelah inisialisai nilai fitness masing” individu terpilih K1 dengan nilai fitness 53.

1. Selection

Menentukan nilai fungsi relative serta cumulative dari populasi yang ada untuk melakukan proses Seleksi Roulette Wheel

|  |  |  |
| --- | --- | --- |
| Kromosom | Fitness Relative (K[i]) | Fitness Cummulative (C[i]) |
| 0 |  | 0 – 0,214 |
| 1 |  | 0,214 – 0,472 |
| 2 |  | 0,472 – 0,785 |
| 3 |  | 0,785 – 0,999 |

Diketahui jika Nilai Acak Kromosom R[i] adalah sebagai berikut :

|  |  |
| --- | --- |
| * + - R[1] = 0. 14     - R[2] = 0.79 | * + - R[3] = 0.65     - R[4] = 0.47 |

Kromosom yang terpilih sebagai *Parent* berdasarkan nilai acak :

K0, K3, K2, K1

Pasangan 1 : K0 – K3

Pasangan 2 : K2 – K1

1. Crossover

Probabilitas Crossover : 0,62

Angka acak : R[1] = 0,46 ; R[2] = 0,21

Titik Rekombinasi = 6

|  |  |  |  |
| --- | --- | --- | --- |
| **Kromosom Orang Tua** | | **Titik Rekombinasi** | **Kromosom Anak Hasil Rekombinasi** |
| K0 | 0001110111011000 | 6 | 0001110111011000 |
| K3 | 1000111001010011 | 6 | 1000110111011000 |
| K2 | 0110111111101010 | 6 | 0110110110110010 |
| K1 | 1101000110110010 | 6 | 1101001111101010 |

1. Mutasi

Probabilitas Mutasi : 0,34

Nilai acak :

R[1] = 0,49

R[2] = 0,55

R[3] = 0,21

R[4] = 0,04

Titik mutasi : 6

|  |  |  |  |
| --- | --- | --- | --- |
| **Kromosom Anak Hasil Rekombinasi** | | **Hasil Mutasi** | **Total nilai prioritas maing-masing kromosom** |
| K0 | 0001110111011000 | 0001110111011000 | 38 |
| K3 | 1000110111011000 | 1000110111011000 | 40 |
| K2 | 0110110110110010 | 0110100110110010 | 54 |
| K1 | 1101001111101010 | 1101001111101010 | 50 |

**Best Individu iterasi pertama adalah masih K1 dengan nilai fitness 53.**

* 1. **Analisa perhitungan manual dan perhitungan program**

Hasil perhitungan perhitungan dengan program pada pengujian pertama. Berikut hasil iterasi pertama pada program :

* Iterations: 1
* Maximum Weight: 5000g
* All Chromosomes Priority Value: [38, 40, 54, 50]
* Best Individu Overall: 54
* Best Individu Chromosome: [0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0]

Perbandingan hasil perhitungan manual dan perhitungan pada program dengan menggunakan nilai acak yang sama mendapatkan hasil yang sama pada iterasi pertama yakni dengan kromosom K1 menjadi best individu dengan nilai fitness 53. Ini menunjukkan keakuratan implementasi algoritma genetika dalam pemilihan orangtua untuk crossover pada iterasi pertama.

1. **Kasus Pengujian 2**

Pengujian dilakukan dengan menjalankan algoritma genetika sebanyak 5 iterasi. Dengan maksimal berat yang dibawa adalah 4kg.Berikut adalah hasil pengujiannya:

* Iterations: 5
* Maximum Weight: 4000g
* All Chromosomes Priority Value: [51, 45, 43, 48, 50]
* Best Individu Overall: 51
* Best Individu Chromosome: [1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0]

Items Carried:

* + - 1. Laptop Charger
      2. Laptop
      3. Power Bank
      4. Phone Charger Pouch
      5. Flash Drive
      6. Keyboard
      7. Notebook
      8. Water Bottle
      9. Umbrella
      10. Medicinal Pouch

1. **Kasus Pengujian 3**

Pengujian dilakukan dengan menjalankan algoritma genetika sebanyak 5 iterasi. Dengan maksimal berat yang dibawa adalah 4kg.Berikut adalah hasil pengujiannya:

* Iterations: 5
* Maximum Weight: 3000g
* All Chromosomes Priority Value: [52, 54, 54, 48, 56]
* Best Individu Overall: 56
* Best Individu Chromosome: [1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0]

Items Carried:

Laptop Charger

Laptop

ANC Headset

Wireless Mouse

Keyboard

Pencil Case

USB Hub

Medicinal Pouch

1. **Daftar Referensi**

Novianty, A. (2023). Algoritma Genetika (AG)

Novianty, A. (2023). Algoritma Genetika (AG) Studi Kasus-1

Novianty, A. (2023). Algoritma Genetika (AG) Studi Kasus-2

Kie Codes. (2020). Genetic Algorithm from Scratch in Python (tutorial with code). Diakses dari https://www.youtube.com/watch?v=nhT56blfRpE

1. **Lampiran Source Code**

Source code dapat dilihat di <https://www.github.com/FaradayB/GeneticAlgorithm_Knapsack>

Berikut terlampir source code yang telah disusun untuk implementasi Algoritma Genetika dalam Python. Fokusnya pada penyelesaian *Knapsack Problem* yang telah didefinisikan.

|  |
| --- |
| **Get data from CSV file** |
| import pandas as pd  import random  df = pd.read\_csv('Carry\_Items.csv') |
| **Population Initialization** |
| def initiate\_Population():      list\_Population = []        for \_ in range(4):          chromosome = [random.randint(0,1) for \_ in range(df.shape[0])]          list\_Population.append(chromosome)      return list\_Population |
| **Fitness Calculations for each chromosome** |
| import copy  def fitness\_func(list\_popu, weight\_cond, index\_prog):      list\_pop = list\_popu      list\_popAwal = copy.deepcopy(list\_pop)          for i in range(len(list\_pop)):          for j in range(len(list\_pop[i])):              if list\_pop[i][j] == 1:                  list\_pop[i][j] = df.iloc[j, 2]              else:                  list\_pop[i][j] = 0      index = 0      while index < len(list\_pop):          if sum(list\_pop[index]) > weight\_cond:              while sum(list\_pop[index]) > weight\_cond:                  chromosome = [random.randint(0,1) for \_ in range(df.shape[0])]                  list\_pop[index] = chromosome          for l in range(len(list\_pop[index])):              if list\_pop[index][l] != 0:                  list\_pop[index][l] = df.iloc[l, 3]              else:                  continue          index += 1        all\_pop = [sum(list\_pop[i]) for i in range(len(list\_pop))]      print("\*\*Fitness Function\*\*")      print(f"Best individu {index\_prog} = {max(all\_pop)}")      return list\_pop, list\_popAwal, max(all\_pop) |
| **Parent Selection Roulette Wheeler** |
| from table import table\_popu  def selection\_func(list\_pop, list\_popAwal):      print("\*\*Selection Function\*\*")      cumulative\_prob = table\_popu(list\_pop, list\_popAwal)        rand\_R = [round(random.uniform(0, 1), 2) for \_ in range(len(list\_pop))]      print(f'Random selected probability: {rand\_R}')        parent = []        for rand\_val in rand\_R:          cumulative\_sum = 0          for i, prob in enumerate(cumulative\_prob):              cumulative\_sum += prob              if rand\_val <= cumulative\_sum:                  parent.append(i)                  break        print(f"Selected parent: {parent}")        return list\_popAwal |
| **Selected Parent gen Crossover** |
| def crover\_func(list\_popAwal):      prob\_crover = round(random.uniform(0,1), 2)      r\_crover = [round(random.uniform(0,1), 2) for \_ in range(int(len(list\_popAwal)/2))]      crover\_point = random.randint(0, 10)        print(" ")      print("\*\*Crossover Function\*\*")      print(f"Crossover Probability: {prob\_crover}")      print(f"Probability per Relations: {r\_crover}")      print(f"Crossover Point: {crover\_point}")        i = 0      j = 0        while i<2:          print("")          print(f"i = {i}")          print(list\_popAwal[j])          print(list\_popAwal[j+1])          if r\_crover[i] < prob\_crover:              print("changed")              temp = list\_popAwal[j+1][crover\_point:16]              list\_popAwal[j+1][crover\_point:16] = list\_popAwal[j][crover\_point:16]              list\_popAwal[j][crover\_point:16] = temp          print(list\_popAwal[j])          print(list\_popAwal[j+1])          i += 1          j += 2        return list\_popAwal |
| **Gen Mutation** |
| def mutation\_func(list\_popAwal):      prob\_mut = round(random.uniform(0,1), 2)      r\_mut = [round(random.uniform(0,1), 2) for \_ in range(int(len(list\_popAwal)))]      mut\_point = random.randint(0, 10)        print(" ")      print("\*\*Mutation Function\*\*")      print(f"Mutation Probability: {prob\_mut}")      print(f"Probability per Relations: {r\_mut}")      print(f"Mutation Point: {mut\_point}\n\n")        for i in range(len(list\_popAwal)):          if r\_mut[i] < prob\_mut:              if list\_popAwal[i][mut\_point] == 1:                  list\_popAwal[i][mut\_point] = 0              elif list\_popAwal[i][mut\_point] == 0:                  list\_popAwal[i][mut\_point] = 1        return list\_popAwal |
| **Main Function** |
| df.head(16)  print('''Bobe wants to put some items into his bag with three different maximum weights, but he can only carry a few items as mentioned above''')  print('''The problem is, Bobe has a bag with a maximum weight of:        1. 3kg        2. 4kg        3. 5kg        ''')  all\_pop = []  max\_val = []  while True:        prog = input("Run program?(Y/n)")        if prog.lower() == "y":              iter = int(input("How many iterations do you want to perform: "))              probl = int(input("Which maximum weight of the bag would Bobe use: "))              if probl == 1:                    cond = 3000              elif probl == 2:                    cond = 4000              elif probl == 3:                    cond = 5000                for i in range(iter+1):                    if i == 0:                          list\_pop = initiate\_Population()                      list\_pop, list\_popAwal, max\_ValLoop = fitness\_func(list\_pop, cond, i)                    temp\_list = copy.deepcopy(list\_popAwal)                    for x in range(len(list\_popAwal)):                          if sum(list\_pop[x]) == max\_ValLoop:                                all\_pop.append(temp\_list[x])                                max\_val.append(sum(list\_pop[x]))                    list\_popAwal = selection\_func(list\_pop, list\_popAwal)                    list\_popAwal = crover\_func(list\_popAwal)                    list\_pop = mutation\_func(list\_popAwal)                print(f"Itterations: {iter}")              print(f"Maximum Weight: {cond}g")              print(f"All Chromosomes Priority Value:\n{max\_val}\n")              print(f"Best Individu Overall       : {max(max\_val)}")              print(f"Best Individu Chromosome    : {all\_pop[max\_val.index(max(max\_val))]}")                print("Items Carried: ")              j = 1              for x in range(len(all\_pop[max\_val.index(max(max\_val))])):                  if all\_pop[max\_val.index(max(max\_val))][x] == 1:                        print(f"{j}. {df.iloc[x][1]}")                        j+=1              all\_pop.clear()              max\_val.clear()              print("\n\n\n")        elif prog.lower() == "n":              print("Terima kasih")              break        else:              continue |