LAPORAN PENGANTAR KECERDASAN BUATAN

Disusun untuk memenuhi salah satu tugas mata kuliah Pengantar Kecerdasan Buatan

Oleh Kelompok 1:

1. Kurniadi Ahmad Wijaya 1301194024 / IF 43 09
2. Naufal Haritsah Luthfi 1301194073 / IF 43 09
3. Hanvito Michael Lee 1301190090 / IF 43 09

FAKULTAS INFORMATIKA

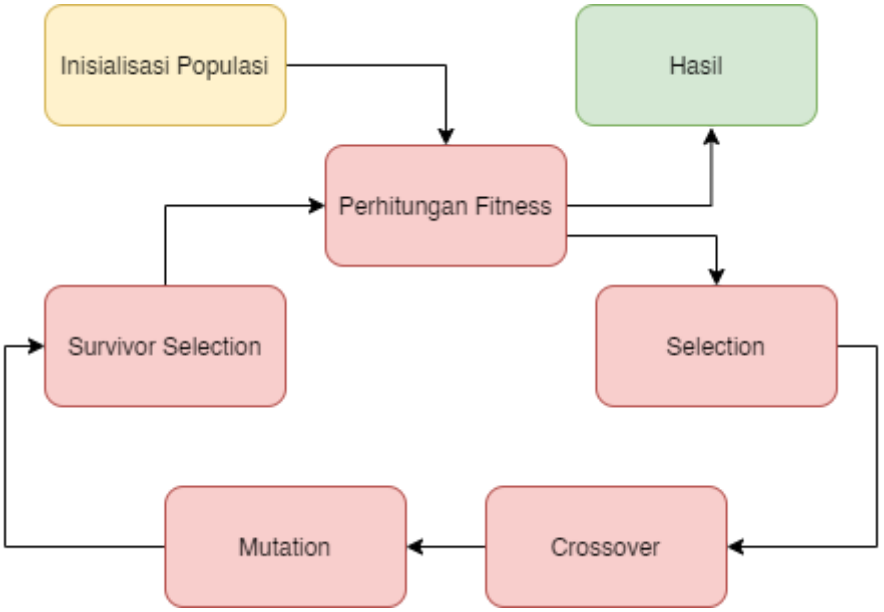
TELKOM UNIVERSITY

2021

Daftar Isi

1. Pengenalan Algoritma Genetika

Algoritma genetika merupakan algoritma komputasi yang terinspirasi teori evolusi Darwin yang kemudian ubah menjadi algoritma komputasi untuk mencari solusi suatu permasalahan dengan cara yang lebih "alamiah"(Hermawanto, 2012, hal. 1).



Dalam penyelesaian algoritma genetika, pertama kita harus mendefinisikan populasi awal yang berisi individu dengan jumlah yang diinginkan serta kromosom tiap-tiap individu. Kemudian kita harus mencari nilai fitness dari tiap tiap individu. Kemudian kita dapat melakukan seleksi berdasarkan nilai fitness tersebut untuk menentukan individu terbaik dalam suatu generasi. Lalu dilakukan proses crossover dilanjutkan dengan proses mutasi tiap tiap kromosom sehingga menghasilkan generasi selanjutnya. Hal ini akan terus berulang hingga generasi terakhir dan akan didapat hasil akhir.

2. Algoritma Genetika Pencarian Nilai Maksimum

2.1. Parameter - Parameter Yang Digunakan

- Desain kromosom : Kumpulan array dengan representasi data integer
- Probabilitas crossover : 0.9 (90%)
- Probabilitas mutasi : 0.1 (10%)
- Ukuran populasi : 10
- Ukuran kromosom : 6

2.2. Rumus

Dibawah ini merupakan rumus yang telah ditentukan untuk mencari nilai fungsi serta nilai maksimum dengan batasan x dan y. Adapun kami merepresentasikan limit atau batas atas dan bawah pada soal menjadi 2 array yaitu x_limit dan y_limit dan fungsi soal kedalam sebuah fungsi seperti pada gambar dibawah.

```
x_limit = [-1, 2]
y_limit = [-1, 1]
```

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

x_limit = Limit untuk x
y_limit = Limit untuk y
dengan batasan -1 ≤ x ≤ 2 dan -1 ≤ y ≤ -1

3. Representasi Data (Integer)

Pada pembuatan algoritma genetika kelompok kami menggunakan representasi data dalam bentuk integer sebagai acuan mencari nilai maksimum 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 dalam integer
r_b = batas bawah
r_a = batas atas
g = genotype

4. Pengkodean Algoritma Genetika

Untuk melakukan 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 pembuatan populasi beserta kromosomnya
def generate_population(p, k):
    return [[random.randint(0,9) for _ in range(k)] for _ in range(p)]
```

Fungsi generate_population diatas digunakan untuk membuat populasi beserta kromosomnya. Kelompok kami membuat sebanyak sepuluh individu dengan enam kromosom tiap individu. Kromosom bernilai acak dari nol sampai sembilan. Array pertama generate array kromosom sebanyak k yg dimasukkan pada main. Array kedua menggenerate populasi sebanyak p yang dimasukkan pada main.

```
# Fungsi untuk melakukan proses mutasi anak
def mutation(child_1, child_2):
    for i in range(len(child_1)):
        p = random.random()
        if p < 0.1:
            child_1[i] = random.randint(0,9)

        q = random.random()
        if q < 0.1:
            child_2[i] = random.randint(0,9)

    return child_1, child_2
```

Fungsi mutation diatas untuk mengganti nilai kromosom pada suatu individu (anak) secara acak dengan menggunakan metode . Mutasi akan terjadi jika probabilitas random yang didapat kurang dari 0.1 atau 10%.

4.2. Proses Decode dan Menghitung Nilai Fitness

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

4.2.1. Decode

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

    return limit[0] + (((limit[1] - limit[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.

4.2.2. Menghitung Nilai Fitness

```
def function(x,y):
    return (cos(x**2) * sin(y**2)) + (x + y)
```

Fungsi function diatas digunakan untuk mencari nilai fitness yang mana nilai fitness sama dengan nilai fungsinya. Karena kita ingin mencari nilai maksimum pada soal, kami menggunakan rumus f = h, dimana h = persamaan yang ada pada soal.

4.3. Proses Seleksi Pemilihan Orang Tua

```
# Fungsi untuk melakukan proses seleksi orang tua
def parent_roulette_selection(population, fitness, fitness_total):
    r = random.random()
    i = 0
    while r > 0:
      r -= fitness[i]/fitness_total
      i += 1
      if  i == len(population) - 1:
          break

    return population[i]
```

Fungsi parent_roulette_selection yaitu fungsi yang digunakan untuk mencari orang tua dari suatu individu secara acak dengan prinsip *roulette wheel* yaitu dengan acuan nilai fitness pada setiap kromosom. Semakin besar nilai fitness pada kromosom maka semakin besar kemungkinan kromosom tersebut terpilih untuk menjadi orang tua

4.4. Proses Crossover Orang Tua

```
def crossover(parent_1, parent_2) :
    child_1, child_2, childs = [], [], []
    pc = random.random()

    if pc < 0.9:
      child_1[:1], child_1[1:] = parent_1[:1], parent_2[1:]
      child_2[:1], child_2[1:] = parent_2[:1], parent_1[1:]
      childs.append(child_1)
      childs.append(child_2)
    else:
      childs.append(parent_1)
      childs.append(parent_2)

    return childs
```

Fungsi crossover orang tua merupakan fungsi yang digunakan untuk membuat kromosom baru antara dua kromosom yang 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%.

4.5. Proses Mutasi Anak

```
def mutation(child_1, child_2):
    for i in range(len(child_1)):
        p = random.random()
        if p < 0.1:
            child_1[i] = random.randint(0,9)

        q = random.random()
        if q < 0.1:
            child_2[i] = random.randint(0,9)

    return child_1, child_2
```

Fungsi mutasi anak merupakan fungsi yang digunakan untuk mengubah fenotype dari gen atau mengubah nilain 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%.

4.6. Proses Pergantian Generasi

```
def elitisme(population, best_kromosom_generation, bad_kromosom, total_fitness):
    if  best_kromosom_generation[1] > bad_kromosom[0] and (best_kromosom_generation[0] not in population):
        population[bad_kromosom[2]] = best_kromosom_generation[0]
        total_fitness = (total_fitness - bad_kromosom[0]) + best_kromosom_generation[1]

        print('\nProses Elitisme')
        print(f'Kromosom Ke-{bad_kromosom[2]+1}: {bad_kromosom[1]}, fitness: {bad_kromosom[0]}')
        print(f'diubah menjadi {best_kromosom_generation[0]}, fitness: {best_kromosom_generation[1]}\n')

    return population, total_fitness
```

```
def best_kromosom_selection(population):
    max_fitness = -999

    for kromosom in population:
        kromosom_a, kromosom_b = split_kromosom(kromosom)
        x1 = decode(kromosom_a, x_limit)
        x2 = decode(kromosom_b, y_limit)
        fitness = function(x1, x2)

        if  max_fitness < fitness:
            max_fitness = fitness
            max_kromosom = kromosom

    return max_kromosom, max_fitness, x1, x2
```

```
if  gen != generation-1 :
    for i in range(population_total // 2):
        parent_1 = parent_roulette_selection(population, fitness_data, total_fitness)
        parent_2 = parent_roulette_selection(population, fitness_data, total_fitness)

        childs = crossover(parent_1, parent_2)
        child_1, child_2 = mutation(childs[0], childs[1])

        new_population.append(child_1)
        new_population.append(child_2)

    population = new_population
```

Pada 2 fungsi di atas yaitu elitisme, best_kromosom_selection, 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.

5. Hasil Observasi

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

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

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

```
============================================================
Generasi 1
============================================================
Kromosom Terbaik : [9, 2, 7, 7, 7, 9]
Fitness Terbaik : 2.0355773670532904

============================================================
Generasi 2
============================================================
Kromosom Terbaik : [9, 9, 2, 5, 5, 7]
Fitness Terbaik : 2.084625050861727

Proses Elitisme
Kromosom Ke-3: [4, 0, 6, 5, 1, 6], fitness: 0.25334217351827204
diubah menjadi [9, 2, 7, 7, 7, 9], fitness: 2.0355773670532904


============================================================
Generasi 3
============================================================
Kromosom Terbaik : [9, 2, 7, 7, 7, 9]
Fitness Terbaik : 2.0355773670532904

Proses Elitisme
Kromosom Ke-10: [0, 2, 7, 2, 7, 9], fitness: -1.2317506000118692
diubah menjadi [9, 9, 2, 5, 5, 7], fitness: 2.084625050861727
```

```
============================================================
Generasi 98
============================================================
Kromosom Terbaik : [6, 2, 1, 9, 8, 9]
Fitness Terbaik : 2.4455102999051745

Proses Elitisme
Kromosom Ke-5: [5, 1, 2, 2, 9, 5], fitness: 0.2880443225477872
diubah menjadi [6, 2, 2, 9, 9, 9], fitness: 2.4817273053234574


============================================================
Generasi 99
============================================================
Kromosom Terbaik : [6, 2, 2, 9, 9, 9]
Fitness Terbaik : 2.4817273053234574

============================================================
Generasi 100
============================================================
Kromosom Terbaik : [6, 2, 2, 9, 9, 9]
Fitness Terbaik : 2.4817273053234574

====================================================
Hasil Akhir Kromosom Terbaik
====================================================
Kromosom Terbaik         =  [6, 2, 2, 9, 9, 9]
Phenotype x              =  0.8678678678678682
Phenotype y              =  1.0
Nilai Fungsi / Fitness   =  2.4817273053234574
====================================================
```

Gambar diatas merupakan contoh hasil dari generate populasi dari generasi pertama hingga generasi terakhir beserta kromosomnya. Untuk proses generate kromosom dilakukan sesuai parameter yang ditentukan jumlah populasi dan kromosomnya yang pada kasus ini yaitu 10 populasi dan 6 kromosom. Nilai x didapat dari proses decode setengah kromosom pertama yang bersangkutan dengan limit yang telah ditentukan dan nilai y didapat dari proses decode setengah kromosom selanjutnya dengan limit yang telah ditentukan serta nilai fitness pada tiap-tiap kromosom.

5.2. Proses Berhenti dan Penentuan Kromosom Terbaik

```python
# Memanggil fungsi untuk menentukan kromosom terbaik pada keseluruhan generasi
print('\n====================================================')
print('Hasil Akhir Kromosom Terbaik')
print('====================================================')
print('Kromosom Terbaik         = ', most_best[0])
print('Phenotype x              = ', most_best[2])
print('Phenotype y              = ', most_best[3])
print('Nilai Fungsi / Fitness   = ', most_best[1])
print('====================================================')
```

```
====================================================
Hasil Akhir Kromosom Terbaik
====================================================
Kromosom Terbaik         =  [6, 2, 2, 9, 9, 9]
Phenotype x              =  0.8678678678678682
Phenotype y              =  1.0
Nilai Fungsi / Fitness   =  2.4817273053234574
====================================================
```

Setelah mendapatkan kromosom dengan nilai fitness terbesar dan keseluruhan generasi perulangan generasi beserta proses selection parent, crossover, mutation dan elitisme 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 diatas.

6. Kesimpulan

Dengan menentukan dan menjalankan proses algoritma genetika sesuai langkah-langkah yang ada kita dapat membuktikan bahwa nilai maksimum yang dapat kita hasilkan dari fungsi $h(x, y) = (\cos x\text{^}2 * \sin y\text{^}2) + (x + y)$ adalah 2.48173 dengan salah satu nilai yang memenuhi persamaan pada nilai x adalah 0.86 dan nilai y adalah 1 dengan bentuk kromosom [6, 2, 2, 9, 9, 9]. Nilai tersebut dapat kita dapatkan dengan melakukan perulangan hingga generasi ke 100 dengan nilai fitness tertinggi yang awalnya didapatkan pada generasi ke 87 (lampiran ke-2).

```
=====================================================
Hasil Akhir Kromosom Terbaik
=====================================================
Kromosom Terbaik          = [6, 2, 2, 9, 9, 9]
Phenotype x               = 0.8678678678678682
Phenotype y               = 1.0
Nilai Fungsi / Fitness    = 2.4817273053234574
=====================================================
```

7. Daftar Pustaka

Hermawanto, D. (2012). Algoritma Genetika dan Contoh Aplikasinya. *Universitas Brawijaya*, *5*(4), 1–8. wayanfm@ub.ac.id

Arkeman, Y. (2012). Algoritma Genetika, Teori Dan Aplikasinya Untuk Bisnis Dan Industri. PT Penerbit IPB Press

Salman, F. (2020, Juli 1). Ilustrasi tentang algoritma genetika [Halaman Web]. Diakses dari https://medium.com/milooproject/panduan-ilustrasi-tentang-algoritma-genetika-902ca22b27dc

8. Lampiran

1. Link github proses pengerjaan:
   https://github.com/ShinyQ/Tugas-Pengantar-AI-1_Genetic-Algorithm

2. Link hasil pengerjaan:
   https://colab.research.google.com/drive/1kJhtByTEJDpaN2Zv2nTSEIm4d0Mxvh-w?usp=sharing

3. Link Video Presentasi:
   https://drive.google.com/file/d/1nkJfLaWNfnK3Jkv3f9HbvSHXGSqu64hj/view?usp=sharing

4. Kalkulasi solusi maksimal berdasarkan fungsi dan batasannya:
   https://bit.ly/maksimal_fungsi_tupro1_ai