# TUGAS PEMROGRAMAN 1

**Laporan**

diajukan untuk memenuhi tugas pada mata kuliah Pengantar Kecerdasan Buatan


oleh:

Ismar Apuandi (1301194382)

Kadek Rizky Francisca Putra(1301194035)

Rafly Rhamadhan (1301194167)




**S1 INFORMATIKA**

**FAKULTAS INFORMATIKA**

**UNIVERSITAS TELKOM**

**BANDUNG**


**2021**

# KATA PENGANTAR

*Bismillahirrahmanirahim*

Assalamu'alaikum Warahmatullahi Wabarakatuh

Dengan Menyebut nama Tuhan Yang Maha Pengasih lagi Maha Penyayang. Puji dan syukur kita panjat kan kepada Tuhan Yang Maha Esa yang telah memberikan rahmat serta karunianya sehingga kami dapat menyelesaikan makalah ini dalam waktu yang telah ditentukan.

Penulis telah menyusun makalah ini dengan semaksimal mungkin dengan mendapat bantuan dari berbagai pihak sehingga laporan ini bisa diselesaikan. Oleh karena itu, penulis berterima kasih kepada seluruh pihak yang telah berkontribusi dalam pembuatan laporan ini. Terkhusus kami ucapkan terima kasih kepada Dosen pembimbing dan Dosen mata kuliah Pengantar Artificial Intelligence yang telah mendukung secara penuh dalam pengerjaan makalah ini.

Terlepas dari semua itu, penulis menyadari sepenuhnya bahwa masih ada kekurangan baik dari segi susunan kalimat maupun tata bahasanya. Oleh karena itu, dengan tangan terbuka kami menerima segala saran dan kritik dari pembaca agar penulis dapat memperbaiki dan memperoleh hasil yang lebih baik kedepannya.

## Abstrak

Algoritma genetika atau disingkat dengan GA merupakan sebuah algoritma yang memanfaatkan proses seleksi alamiah yang dikenal dengan proses evolusi. Algoritma genetika ini secara keseluruhan merupakan proses yang terinspirasi dari proses evolusi biologi yang berdasarkan teori evolusi Charles Darwin.

Dalam proses evolusinya, individu-individu akan diseleksi secara terus menerus sehingga akan mengalami perubahan gen pada individu tersebut untuk menyesuaikan dengan lingkungan hidupnya yang baru, disini hanya individu-individu kuatlah yang nantinya dapat bertahan. Oleh karena itu evolusi yang terjadi pada algoritma genetika menghasilkan individu yang lebih baik dari individu sebelumnya.



Keyword : algoritma genetika, GA, evolusi, seleksi, individu


## 1. Pendahuluan

Genetic Algorithm (GA) atau Algoritma Genetika merupakan metode metaheuristic yang terinspirasi dari proses seleksi natural. Algoritma genetika ini pertama kali diperkenalkan oleh John Holland dan teman-temannya di universitas Michigan untuk aplikasi seluler automata. Dan teknik tersebut menjadi populer di kalangan saintis dan rekayasawan untuk memecahkan permasalahan optimasi mereka.

## 2. Strategi Penyelesaian Masalah

### 2.1 Desain Kromosom dan Ukuran Populasi

Pada program yang kami buat untuk kasus ini, kami menggunakan desain kromosomnya yang dimana gen nya berjumlah 14 dan populasi yang berjumlah 200 kromosom

Alasannya karena jika populasi semakin banyak maka akan semakin cepat pula menemui nilai maksimumnya dikarenakan dengan jumlah populasi yang besar akan lebih cepat untuk mencapai nilai maksimal dengan menggenerasikan populasi jadi tidak banyak meregenerasi populasinya.

```
popul_len = 200
chrom_len = 14
```

```python
def generateChromosome(chrom_len):
    chromosome = [int(random.choice([1, 0])) for i in range(chrom_len)]
    return chromosome

def generatePopulation(popul_len, chrom_len):
    population = [generateChromosome(chrom_len) for i in range(popul_len)]
    return population
```

### 2.2 Metode *Pendekodean*

Untuk metode *pendekodean*-nya dari program yang kami buat, kami menggunakan metode representasi biner dimana gen-gen di dalam kromosomnya berbentuk biner yang nantinya akan diubah menjadi desimal pada *decode* yang kami buat.

Untuk rumus pendekodean dari representasi biner yang kami gunakan yaitu :

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

Dengan menggunakan rumus di atas didapatkan nilai X dan Y dengan ketentuan batas atas X sebesar 2 dan batas atas Y sebesar 1, sedangkan untuk batas bawah X adalah -1 dan batas bawah Y juga sebesar -1. Setelah itu, nilai X dan Y dimasukkan ke dalam fungsi fitness berikut :

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

```python
def decodeChrom(chrom):
    rax = 2
    ray = 1
    rbx = -1
    rby = -1
    sumGx = 0
    sumGy = 0
    sumKuadratx = 0
    sumKuadraty = 0
    for i in range(len(chrom)//2):
        sumGx = sumGx + chrom[i] * (2**(-(i+1)))
        sumKuadratx = sumKuadratx + (2**(-(i+1)))
    for i in range(len(chrom)//2, len(chrom)):
        sumGy = sumGy + chrom[i] * (2**(-(i+1)))
        sumKuadraty = sumKuadraty + (2**(-(i+1)))
    x = rbx + ((rax - rbx)/sumKuadratx)*sumGx
    y = rby + ((ray - rby)/sumKuadraty)*sumGy
    return x, y
```

```python
def generateFitness(population):
    fitness = list()
    for i in population:
        x, y = decodeChrom(i)
        hxy = ((math.cos(x**2))*(math.sin(y**2))) + (x+y)
        fitness.append(hxy)
    return fitness
```

### 2.3 pemilihan orang tua

Pemilihan orang tua dalam algoritma genetika yang kami buat menggunakan metode tournament selection dimana cara kerja dari tournament selection ini adalah dilakukan dengan teknik sampling, dimana kami memilih kromosom sebanyak ¼ dari jumlah kromosom di dalam populasi secara acak. Lalu kromosom-kromosom yang terpilih akan masuk ke dalam turnamen, kromosom yang memiliki nilai fitness terbesar akan memenangkan turnamen dan menjadi orang tua.

```python
def tournament(population, fitness, n):
    idx_chrom = random.sample(range(n), round(n/4))
    # print(idx_chrom)
    parent_candidate = [(fitness[idx_chrom[i]], population[idx_chrom[i]])
                        for i in range(round(n/4))]
    grade = sorted(parent_candidate, key=lambda x: x[0], reverse=True)
    parent = grade[0][1]
    return parent
```

### 2.4 Pemilihan dan teknik operasi genetik (crossover dan mutasi)

Pemilihan teknik operasi untuk crossover kami menggunakan metode multi point crossover atau banyak titik jadi disini kami memilih sejumlah titik potong secara acak dengan jumlah antara satu hingga panjang kromosom dikurangi satu, setelah itu dilakukan persilangan di antara titik titik yang telah dipilih dan didapatlah kromosom yang baru, kromosom baru ini didapatkan ketika titik yang dipilih lebih besar daripada Pc(probabilitas operasi genetik crossover yang kami tentukan) maka akan dilakukan crossover pada titik yang dipilih dan setelah itu di dapatlah kromosom baru. Alasan kami menggunakan metode crossover multipoint adalah untuk dapat melakukan lebih banyak pertukaran nilai antar kromosom.

```python
def crossover(parent1, parent2, pc):
    if pc > random.random():
        titik1 = random.randint(0, len(parent1)//2)
        titik2 = random.randint((len(parent1)//2)+1, len(parent1))
        parent1[titi parent2: list ent2[titik1:titik2] = parent2[titik1:titik2], parent1[titik1:titik2]
    return parent1, parent2
```

Untuk teknik mutasi operasi genetik kami menggunakan metode mutasi untuk representasi biner, teknik ini dilakukan dengan cara yang sederhana dimana pada setiap posisi kromosom di bangkitkan suatu bilangan acak antara 0 atau 1, ketika bilangan acak yang dipilih lebih besar daripada Pm(probabilitas operasi genetik mutasi yang kami tentukan) maka gen tersebut akan di mutasi jika lebih rendah maka gen tersebut tidak akan dimutasi, cara untuk melakukan mutasinya adalah dengan cara membalikan angka biner 1 dan 0, jadi jika bernilai 1 maka akan dibalikan menjadi 0 sedangkan jika bernilai 0 maka akan dibalikan menjadi 1. Alasan kami memilih teknik ini karena pada awalnya kami menggunakan representasi biner untuk desain kromosomnya oleh karena itu dipilih teknik mutasi representasi biner karena teknik mutasi untuk representasi biner hanya ada satu.

```
def mutation(chrom, pm):
    for i in range(len(chrom)):
        if pm > random.random():
            if chrom[i] == 0:
                chrom[i] = 1
            else:
                chrom[i] = 0
    return chrom
```

**2.5 Probabilitas operasi genetik (Pc dan Pm)**

Pc atau probabilitas operasi genetik crossover adalah suatu peluang untuk ditentukannya bahwa titik yang sudah dipilih pada kromosom tersebut akan di crossover atau tidak. Disini kami menentukan Pc sebesar 0.7.

```
pc = 0.7
```

Pm atau probabilitas operasi genetik mutasi sebenarnya tidak jauh berbeda dengan Pc yaitu merupakan sebuah peluang untuk menentukan apakah kromosom itu akan dimutasi atau tidak. Disini kami menentukan Pm nya sebesar 0.2, alasan kami memilih angka ini karena menurut hasil observasi kami semakin kecil nilai Pm maka akan semakin cepat dalam mencapai nilai fitness maksimal.

```
pm = 0.2
```

**2.6 Metode pergantian generasi (seleksi survivor)**

Dalam program kami, kami menggunakan metode generation model cara kerja dari generation model adalah suatu populasi berukuran N kromosom pada suatu generasi diganti dengan N kromosom baru pada generasi berikutnya, dan pada generational model ini kita juga menggunakan elitisme. Alasan kami memilih metode ini adalah untuk menjaga kromosom terbaik dengan nilai fitness terbesar dan mempercepat pencarian nilai maksimum.

```python
def regeneration(new_population, population, fitness, popul_len, chrom_len, pc, pm):
    while len(new_population) < popul_len:
        parent1 = tournament(population, fitness, popul_len)
        parent2 = tournament(population, fitness, popul_len)
        parent1 = list(parent1)
        parent2 = list(parent2)
        child1, child2 = crossover(parent1, parent2, pc)
        child1 = mutation(child1, pm)
        child2 = mutation(child2, pm)
        new_population.extend([child1, child2])
```

```python
def elitisme(population, fitness):
    bestChrom = [(fitness[i], population[i]) for i in range(len(population))]
    grade = sorted(bestChrom, key=lambda x: x[0], reverse=True)
    return [grade[0][1]]
```

### 2.7 Pemberhentian

Dalam observasi kami, kami sudah mencoba berbagai kemungkinan dengan mengubah nilai dari Pm, Pc, panjang kromosom, banyak populasi dan jumlah generasi. Dari hasil observasi tersebut kami mendapatkan nilai fitness maksimum yaitu 2.4817210964013796, jadi dengan hasil tersebut kami menentukan bahwa kriteria penghentian evolusi terjadi jika salah satu nilai fitness kromosom telah mencapai nilai fitness maksimum yang sudah kami dapatkan tadi.

```python
while fitness[0] < 2.4817210964013796:
    new_population = elitisme(population, fitness)
    regeneration(new_population, population, fitness, popul_len, chrom_len, pc, pm)
    population = new_population
    count += 1
    print("Generasi ke ", count,"\n", population, end="\n")
    fitness = generateFitness(population)
```

### 2.8 Output Program

Dari strategi beberapa metode yang digunakan diatas dan telah diimplementasikan dalam kode pemrograman bahasa python, berikut adalah contoh output dari program yang dibuat:

```
Generasi ke  1
 [[1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1], [1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0], [1, 1, 0, 1, 1, 1, 0,
Generasi ke  2
 [[1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1], [1, 0, 0, 1, 1, 0, 0,
Nilai Fitness Maksimum Didapatkan Pada Generasi Ke 2
Kromosom Terbaik:  [1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
Nilai x dan y Kromosom Terbaik: x = 0.8661417322834646 dan y = 1.0
Fitness Kromosom Terbaik:  2.4817210964013796
```

## 3. Hasil Kesimpulan

Jadi kesimpulannya pada program yang kami buat adalah, pertama kami membuat desain kromosom yang memiliki panjang 14 dan untuk metode *pendekodeannya* menggunakan representasi biner. Populasi yang kami buat terdiri dari 200 kromosom,. Untuk pemilihan orang tua kami memilih metode tournament selection dengan memasukkan ¼ dari jumlah populasi ke dalam tournament. Untuk operasi genetik crossover kami menggunakan teknik multi point crossover dan untuk mutasi menggunakan representasi biner. Untuk Pc kami mengisinya dengan nilai 0.7 dan untuk Pm bernilai 0.2. Untuk pergantian generasi kami menggunakan metode generation model.  Dan untuk kriteria penghentian dapat dicapai ketika nilai fitness salah satu kromosom mencapai 2.4817210964013796.

## 4. Link Video Presentasi:

1. Kadek Rizky Fransisca Putra : https://youtu.be/ikDOrQKaLP0

2. Ismar Apuandi : https://youtu.be/QYQcZKzFlpg

3. Rafly Rhamadhan : https://youtu.be/FPfOOI0Mjas

## Referensi


1. https://www.geeksforgeeks.org/genetic-algorithms/
2. https://www.youtube.com/watch?v=zumC_C0C25c
3. https://www.youtube.com/playlist?list=PLxYRczqMg8Qrx_5F0ukVZs5WSCVru6qRn
4. https://www.youtube.com/watch?v=4XZoVQOt-0I&t=838s