

## **TUGAS PEMROGRAMAN 01**

*Diajukan untuk memenuhi tugas mata kuliah Pengantar Kecerdasan Buatan*



Disusun oleh:

**Bima Mahardika Wirawan (1301194304)**

**M. Aldi Haryojudanto (1301194025)**

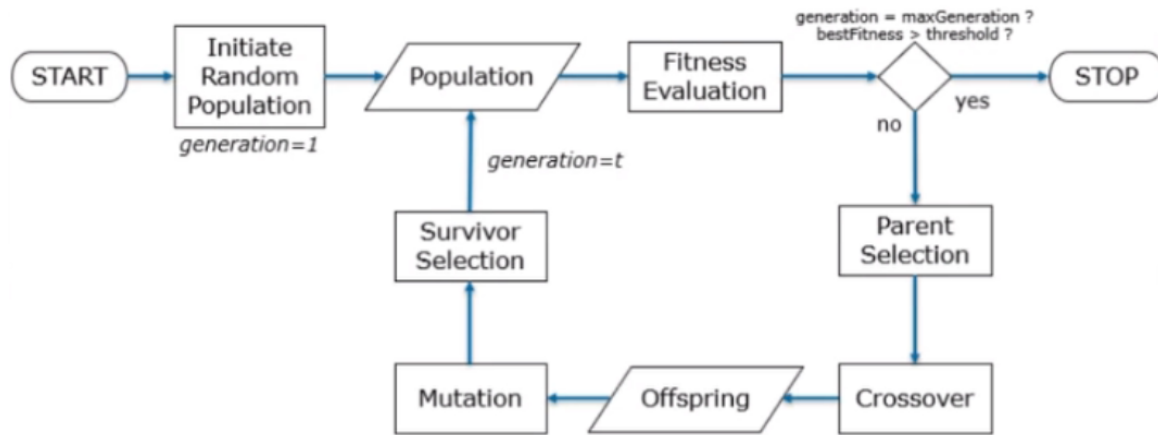
**Fiyona Anmila Syamsir (1301194201)**

**PROGRAM STUDI S1 INFORMATIKA  
FAKULTAS INFORMATIKA  
TELKOM UNIVERSITY  
2021**

Lakukan analisis, desain, dan implementasi algoritma Genetic algorithm (GA) ke dalam suatu program komputer untuk menemukan nilai maximum dari fungsi:

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

Dengan batasan  $-1 \leq x \leq 2$  dan  $-1 \leq y \leq 1$



Sumber : Responsi Pengantar Kecerdasan Buatan

Gambar 1 Langkah-langkah Algoritma Genetika

Hal yang diobservasi:

- Desain Kromosom dan Metode Pendekodean
- Ukuran Populasi
- Pemilihan orangtua
- Pemilihan dan teknik operasi genetik (crossover dan mutasi)
- Probabilitas operasi genetik (Pc dan Pm)
- Metode Pergantian Generasi (Seleksi Survivor)
- Kriteria Penghentian Evolusi

Proses yang harus dibangun (bisa berupa fungsi/prosedur):

- Dekode kromosom
- Perhitungan fitness
- Pemilihan orangtua
- Crossover (pindah silang)
- Mutasi
- Pergantian Generasi

Output dari sistem adalah kromosom terbaik dan nilai  $x$  dan  $y$  hasil **decode** kromosom terbaik tersebut.

Beberapa langkah yang harus ditentukan dalam menjawab permasalahan terkait hal yang diobservasi serta proses yang harus dibangun:

## 1. Representasi Individu (Kromosom)

Kromosom merupakan kumpulan parameter yang mewakili solusi yang mungkin dalam suatu permasalahan yang direpresentasikan dengan nilai genotype dan phenotype. Nilai genotype adalah nilai kromosom dari tiap gen, sedangkan nilai phenotype adalah nilai asli dari kromosom. Pada algoritma genetika menggunakan nilai genotype dari kromosom sehingga memerlukan proses penerjemahan dari nilai genotype ke nilai phenotype (encoding).

Macam-macam desain kromosom:

### a. Binary Encoding

Kromosom biner terdiri dari gen-gen bernilai 1 atau 0. Kromosom ini termasuk model standar dengan tingkat keberhasilan yang tinggi karena jumlah gen pada kromosom biner menunjukkan tingkat ketelitian yang diharapkan.

Genotype	Phenotype
Binary Encoding using 3 bits (3 gens)	
$  \begin{array}{c}  x_1 \qquad \qquad x_2 \\  \boxed{0} \ \boxed{1} \ \boxed{0} \ \boxed{1} \ \boxed{1} \ \boxed{0}  \end{array}  $	
$x = r_{min} + \frac{r_{max} - r_{min}}{\sum_{i=1}^N 2^{-i}} (g_1 * 2^{-1} + g_2 * 2^{-2} + \dots + g_N * 2^{-N})$	$x_1 = -0.57$
$x_1 = -2 + \frac{3 - (-2)}{(2^{-1} + 2^{-2} + 2^{-3})} (0 * 2^{-1} + 1 * 2^{-2} + 0 * 2^{-3})$	$x_2 = 2.29$
$x_2 = -2 + \frac{3 - (-2)}{(2^{-1} + 2^{-2} + 2^{-3})} (1 * 2^{-1} + 1 * 2^{-2} + 0 * 2^{-3})$	
<small>sumber : responsi pengantar kecerdasan buatan Telkom University</small>	

Gambar 2 Kromosom binary encoding

Rumus binary encoding:

$$x = r_{min} + \frac{r_{max} - r_{min}}{\sum_{i=1}^N 2^{-i}} (g_1 * 2^{-1} + g_2 * 2^{-2} + \dots + g_N * 2^{-N})$$

b. Integer Encoding

Kromosom integer terdiri dari gen-gen dimulai dengan nilai 0 sampai 9.

Genotype	Phenotype				
<p>Integer Encoding using 3 gens</p> <table> <tr> <td><math>x_1</math></td><td><math>x_2</math></td></tr> <tr> <td>2   0   5</td><td>7   6   9</td></tr> </table> $x = r_{min} + \frac{r_{max} - r_{min}}{\sum_{i=1}^N 9 * 10^{-i}} (g_1 * 10^{-1} + g_2 * 10^{-2} + \dots + g_N * 10^{-N})$ $x_1 = -2 + \frac{3 - (-2)}{9 * (10^{-1} + 10^{-2} + 10^{-3})} (2 * 10^{-1} + 0 * 10^{-2} + 5 * 10^{-3})$ $x_2 = -2 + \frac{3 - (-2)}{9 * (10^{-1} + 10^{-2} + 10^{-3})} (7 * 10^{-1} + 6 * 10^{-2} + 9 * 10^{-3})$ <p><small>sumber : responsi pengantar kecerdasan buatan Telkom University</small></p>	$x_1$	$x_2$	2   0   5	7   6   9	<p><math>x_1 = -0.97</math></p> <p><math>x_2 = 1.84</math></p>
$x_1$	$x_2$				
2   0   5	7   6   9				

Gambar 3 Kromosom integer encoding

Rumus integer encoding:

$$x = r_{min} + \frac{r_{max} - r_{min}}{\sum_{i=1}^N 10^{-i}} (g_1 * 10^{-1} + g_2 * 10^{-2} + \dots + g_N * 10^{-N})$$

c. Real Encoding

Kromosom real terdiri dari gen-gen dengan rentang nilai dari 0 sampai 1. Kromosom ini termasuk model kromosom yang memiliki jumlah parameter yang banyak sehingga tingkat keberhasilan dari bentuk kromosom ini rendah dalam kecepatan (jumlah generasi).

Genotype	Phenotype				
<p>Real Encoding using 3 gens</p> <table> <tr> <td><math>x_1</math></td><td><math>x_2</math></td></tr> <tr> <td>0.47   0.08   0.13</td><td>0.73   0.92   0.66</td></tr> </table> $x = r_{min} + \frac{r_{max} - r_{min}}{N} (g_1 + g_2 + \dots + g_N)$ $x_1 = -2 + \frac{3 - (-2)}{3} (0.47 + 0.08 + 0.13)$ $x_2 = -2 + \frac{3 - (-2)}{3} (0.73 + 0.92 + 0.66)$ <p><small>sumber : responsi pengantar kecerdasan buatan Telkom University</small></p>	$x_1$	$x_2$	0.47   0.08   0.13	0.73   0.92   0.66	<p><math>x_1 = -0.87</math></p> <p><math>x_2 = 1.85</math></p>
$x_1$	$x_2$				
0.47   0.08   0.13	0.73   0.92   0.66				

Gambar 4 Kromosom real encoding

Pada tugas pemrograman 01, kami menggunakan desain kromosom bilangan biner agar memiliki keberhasilan yang tinggi karena pada permasalahan ini kami menggunakan 8 gen setiap atribut supaya tingkat ketelitiannya yang cukup baik sehingga pada satu kromosom terdiri dari 16 bit atau gen (8 gen dari x dan 8 gen dari y).

```
#DECODE CHROMOSOME (BINARY ENCODING)
def keDesimal(Des):
    return int("".join(map(str,Des)),2) #mengubah bin ke desimal
def decodeChromosome(chrome):
    JUMLAH_GEN = round(len(chrome)/2)
    x = keDesimal(chrome[:JUMLAH_GEN])
    y = keDesimal(chrome[JUMLAH_GEN:])
    pembagi = 2**((len(chrome)/2)-1) #rumus pembagi/sigma n binary encoding
    xbin = -1+((2-(-1))/pembagi)*x #rumus bin encoding variabel x
    ybin = -1+((1-(-1))/pembagi)*y #rumus bin encoding variabel y
    return (xbin,ybin)
```

*Gambar 5 Screenshot code decode chromosome binary encoding*

## 2. Jumlah Populasi dan Generasi

### a. Jumlah Populasi

Jumlah populasi mempengaruhi kinerja dan efektifitas dari Genetic Algorithms. Jumlah populasi kecil maka populasi tidak menyediakan cukup materi untuk mencakup ruang permasalahan, sehingga pada umumnya kinerja yang diberikan Genetic Algorithms menjadi buruk. Selain itu penggunaan populasi yang besar dapat mencegah terjadinya konvergensi wilayah lokal, banyak aplikasi Genetic Algorithms menggunakan populasi pada range 50-100.

Di sini kami menggunakan jumlah populasi sebanyak 50 populasi sesuai dengan batasan umum populasi agar dapat mencakup materi solusi yang cukup bagi ruang permasalahan serta mencegah konvergensi wilayah lokal akibat populasi yang terlalu besar.

### b. Jumlah Generasi

Jumlah generasi merupakan sebuah jumlah perulangan (iterasi) dilakukannya rekombinasi dan seleksi. Jumlah generasi ini mempengaruhi kestabilan output dan lama iterasi. Jumlah generasi yang besar dapat mengarahkan ke arah solusi yang optimal, namun akan membutuhkan waktu yang lama. Sedangkan jika jumlah generasinya terlalu sedikit maka solusi akan terjebak pada lokal optimum.

```
JUMLAH_POPULASI = 50
MUTASI_RATE = 0.01
CROSSOVER_RATE = 0.65
JUMLAH_GEN = 16
JUMLAHGENERASI = 20
```

*Gambar 6 Screenshot inisialisasi jumlah populasi, probabilitas mutasi, probabilitas crossover, jumlah gen, dan jumlah generasi*

```
def bangkitkanPopulasi(n,JUMLAH_GEN):
    populasi = [[int(random.choice([1,0])) for i in range(JUMLAH_GEN)] for j in range(n)]
    return populasi
```

Gambar 7 Screenshot code pembangkitan populasi

Pada kasus ini, kami melakukan jumlah generasi sebanyak 20 generasi. Jumlah tersebut dipertimbangkan tidak terlalu kecil untuk menghindari terjebak di lokal optimum. Selain itu, kami menghindari jumlah generasi yang terlalu banyak supaya mengefektifkan *running time* dan mengurangi beban komputasi pada komputer. Kami pun mengatur beberapa parameter lainnya, seperti probabilitas mutasi sebesar 0.01 sesuai dengan probabilitas mutasi yang mendekati 1%, probabilitas crossover sebesar 0.65 menyesuaikan dengan probabilitas crossover yang terjadi saat mendekati 60% sampai 70%, dan jumlah gen sebesar 16 gen pada satu kromosom atau 8 bit (gen) setiap atribut supaya tingkat ketelitiannya yang cukup baik sehingga pada satu kromosom terdiri dari 16 bit atau gen (8 gen dari x dan 8 gen dari y).

### 3. Fitness

Nilai fitness merupakan ukuran yang menunjukkan baik atau tidaknya suatu solusi. Nilai fitness terbagi menjadi dua:

- Nilai fitness maksimum  
 $f = h$  ;  $h$  adalah fungsi objektif
- Nilai fitness minimum

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

Pada pemecahan masalah ini, kita menggunakan nilai fitness maksimum karena berdasarkan permasalahan yang disebutkan bahwa kita harus mencari nilai maksimum dari fungsi tersebut

```
#PERHITUNGAN FITNESS
def h(x,y):
    return((math.cos(x**2))*(math.sin(y**2)))+(x+y) #fungsi h(x,y) = (cos x^2 * sin y^2) + (x+y)
def hitungFitness(populasi):
    fitness = []
    for i in populasi:
        x,y = decodeChromosome(i)
        fitness.append(h(x,y)) #karena mencari nilai maksimum maka f=h #inverse kalau minimum
    return fitness
```

Gambar 8 Screenshot code perhitungan nilai fitness

### 4. Seleksi Parent

- Tournament
  - Menentukan jumlah individu sebanyak  $n$ .
  - Memilih individu dari populasi sebanyak  $n$  secara random.
  - Menyeleksi nilai fitness terbaik (nilai fitness paling maksimum) di antara individu-individu yang telah dipilih.

### Algoritma tournament

```
function Tournament(populasi, n);
    fbest = null;
    for i = 1 to n {
        indiv = populasi(random(1,n));
        if (fbest = null) or ((fitness(indiv) > fitness(fbest))) {
            fbest = indiv;
        }
    }
    return fbest
}
```

#### b. Roulette Wheel

1. Menghitung probabilitas setiap kromosom.
2. Menginisialisasi titik temu (t) secara acak.
3. Melakukan perulangan dengan pengurangan antara t dan nilai probabilitas.

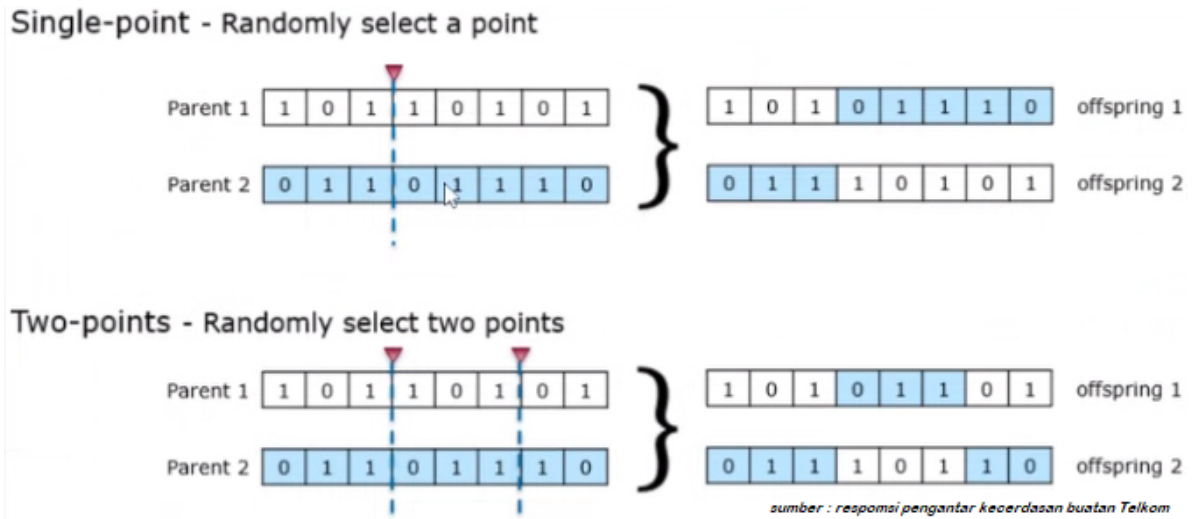
```
Function Roulettewheel(fitness);
    total = 0;
    for i = 1 to n{
        total += fitness[indiv]
    }
    t = random()
    indiv = 1
    while (t > 0) do
        t -= fitness[indiv]/total //probabilitas kromosom
        indiv++
    return indiv
```

Program kami terhadap pemecahan masalah ini menggunakan metode Tournament pada selection parent. Hal tersebut kami pertimbangkan karena metode Tournament memiliki keuntungan yang lebih sesuai dalam permasalahan ini. Inayati(2010), kelebihan metode Tournament, yakni *time complexity* yang cepat, membutuhkan memori yang lebih sedikit dibandingkan metode Roulette Wheel, dan dapat menemukan solusi yang lebih optimal berdasarkan banyak proses yang dilakukan daripada metode Roulette Wheel.

```
#SELEKSI ORANG TUA (TOURNAMENT)
def tournament(populasi,fitness,n):
    acak = random.sample(range(n),round(n/2)) #membuat bilangan acak untuk memilih kandidat turnamen
    kandidat = [[fitness[acak[i]] for i in range(round(n/2))]]
    rank = sorted(zip(kandidat,acak), key=lambda k: k[0], reverse=True) #mengurutkan rank berbasis dari fitness
    return populasi[rank[0][1]], fitness[rank[0][1]] #mengembalikan individu terbaik dari populasi berbasis fitness rank
```

Gambar 9 Screenshot code implementasi Tournament pada selection parent

## 5. Crossover



Gambar 10 Ilustrasi single-point crossover dan two-points crossover

Crossover terjadi saat nilai probabilitas mendekati 60% - 70%. Proses ini dilakukan dengan menentukan 2 posisi awal gen dan akhir gen pada individu yang hendak dilakukan crossover (kawin silang) secara acak. Lalu, nilai nilai parent 1 dan parent 2 ditukar dari posisi awal dan akhir sehingga menghasilkan child 1 dan child 2. Beberapa macam crossover pada algoritma genetika:

a. Single-point Crossover

Single-point adalah pindah silang satu titik potong. Letak titik potong dilakukan secara random.

b. Two-point Crossover

Two-point adalah pindah silang dua titik potong. Letak titik potong dilakukan secara random.

Kami menggunakan single-point crossover agar memudahkan proses kawin silang antar kromosom sehingga menghemat penggunaan memori dan *time complexity* yang lebih cepat.

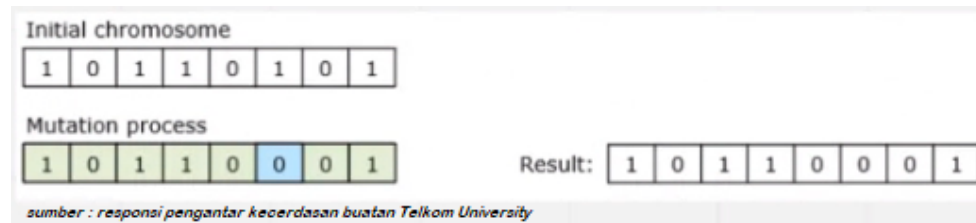
```
#CROSSOVER (SINGLE POINT)
def crossover(parent1,parent2,rate):
    if random.random() <= rate:
        titik_potong = random.choice(range(round(len(parent1)/2),len(parent1)-1))
        temp = parent1[titik_potong:]
        parent1[titik_potong:] = parent2[titik_potong:]
        parent2[titik_potong:] = temp
    return parent1,parent2
```

Gambar 11 Screenshot code implementasi crossover



## 6. Mutation

Mutation berfungsi untuk menjaga variasi populasi dari tiap generasi. Probabilitas dari mutation hanya mendekati 1% sehingga tidak selalu terjadi pada setiap generasi.



Gambar 12 Ilustrasi mutasi pada algoritma genetika

Pertama, kita harus menentukan rasio probabilitas mutasi. Kemudian, kita tentukan titik terjadinya mutasi pada gen secara acak. Jika kita menggunakan desain kromosom biner, saat titik mutasi menunjuk suatu gen yang bernilai 1, maka nilai gen tersebut akan berubah menjadi 0 dan berlaku juga sebaliknya.

```
#MUTASI
def mutasi(chrome,rate):
    m = chrome.copy()
    for i in range(len(m)):
        if m[i] == 1:
            m[i] = 0
        elif m[i] == 0:
            m[i] = 1
    return m
```

Gambar 13 Screenshot code implementasi mutasi

## 7. Survivor Selection

### a. Generational replacement

Hal-hal yang dilakukan survivor selection jenis generational replacement:

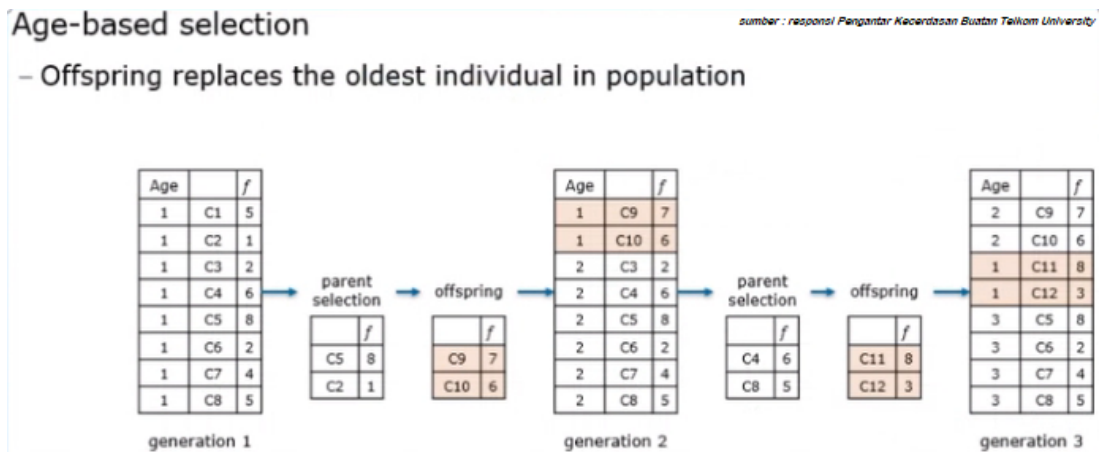
1. Mengganti seluruh individu pada satu generasi.
2. Memiliki elitisme atau proses menjamin bahwa dari satu generasi ke generasi selanjutnya selalu mengalami evolusi terhadap solusi terbaik (menyalin fitness terbaik ke generasi selanjutnya).

### b. Steady-State

Survivor selection jenis steady-state melakukan pergantian sebagian individu pada satu generasi.

Contoh :

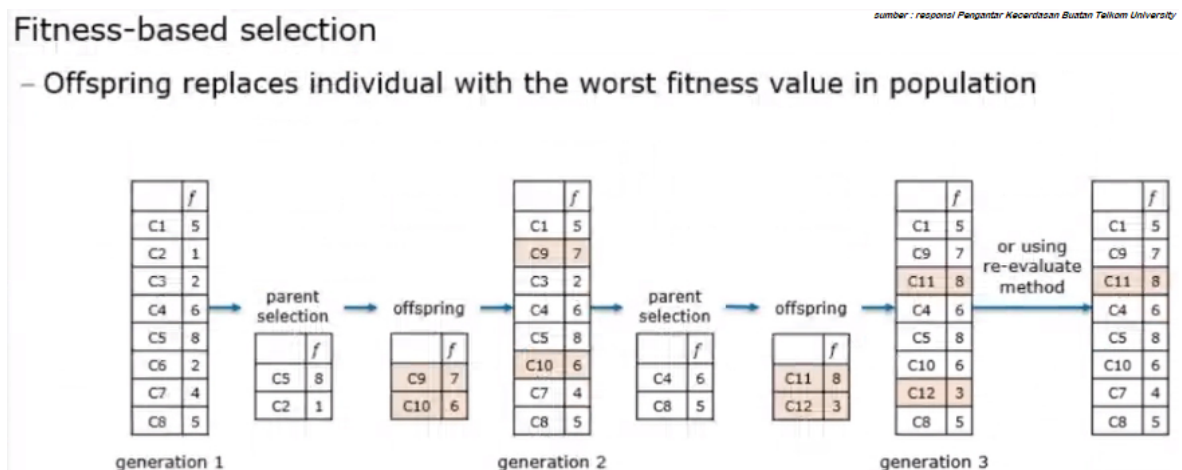
Steady-state pada seleksi umur



Gambar 14 Ilustrasi steady state berdasarkan seleksi umur

Steady-state pada seleksi umur melakukan penggantian kromosom yang memiliki fitness terburuk dengan kromosom offspring. Hal tersebut dilakukan secara sekuensial.

Steady-state berdasarkan fitness



Gambar 15 Ilustrasi steady state berdasarkan fitness-based selection

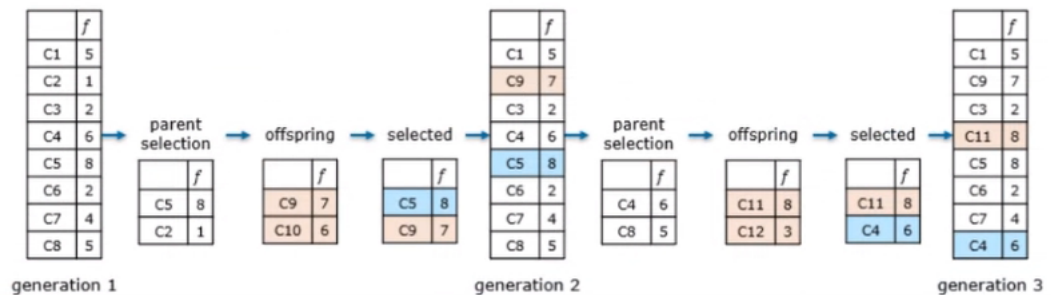
Steady-state berdasarkan fitness melakukan penggantian kromosom yang memiliki fitness terburuk dengan kromosom hasil seleksi parent yang menjadi offspring. Hal ini berlanjut sampai mencapai jumlah generasi maksimum atau nilai fitness terbaik melebihi batasan yang sudah ditentukan.

Steady-state berdasarkan local fitness

### Local Fitness-based selection

sumber : responsi Pengantar kecerdasan Buatan Telkom University

- Select the best individuals form a pair of parents and its offspring to replace one or all the parents



Gambar 16 Ilustrasi steady state berdasarkan local fitness-based selection

Steady-state berdasarkan local fitness mengganti kromosom yang memiliki fitness terburuk dengan kromosom hasil seleksi kromosom parent dan offspring yang memiliki fitness terbaik. Survivor selection ini diteruskan sampai mencapai jumlah generasi maksimum atau nilai fitness terbaik melebihi batasan yang sudah ditentukan.

Kelompok kami menggunakan metode generational replacement pada survivor selection karena steady state cenderung lebih kompleks yang dapat memakan waktu lebih banyak saat *running time*.

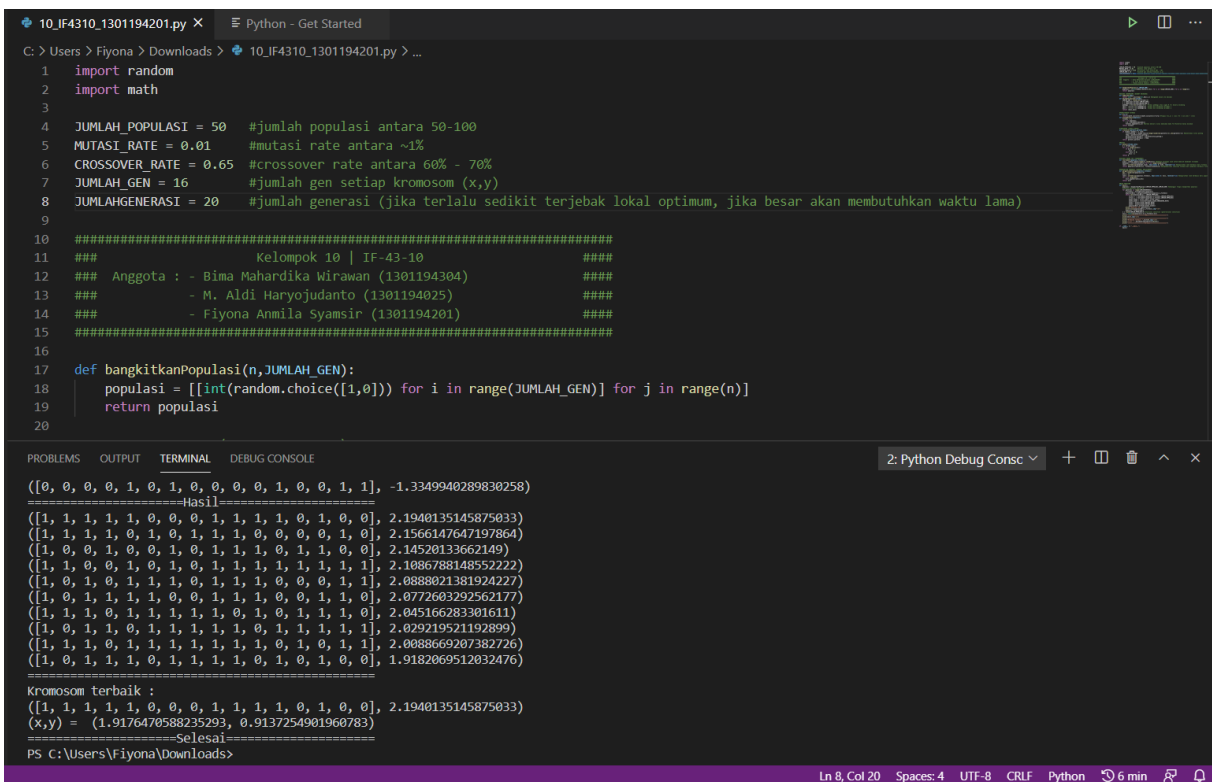
```
#PERGANTIAN GENERASI (GENERAL REPLACEMENT)
def pergantianGenerasi(populasi,fitness):
    max = round(len(populasi)/5)
    elit = []
    rank = sorted(zip(populasi,fitness), key=lambda k: k[1], reverse=True)
    for i in range(max):
        elit.append(rank[i][0])
    return elit
```

Gambar 17 Screenshot code survivor selection menggunakan generational replacement

Setelah menyelesaikan langkah survivor selection di iterasi pertama, algoritma dilanjutkan pada langkah nomor 2 sampai survivor selection selama jumlah generasi belum mencapai kriteria penghentian evolusi, yaitu jumlah generasi maksimum. Kemudian, program akan menampilkan kromosom terbaik alias solusi terbaik diikuti nilai elitisme serta nilai *x* dan *y* dari kromosom tersebut saat jumlah generasi maksimum tercapai.

## Kesimpulan

Berdasarkan analisis dan strategi yang sudah disampaikan, kami dapat menarik kesimpulan bahwa permasalahan mencari nilai maksimum dari fungsi  $h(x, y) = (\cos x^2 * \sin y^2) + (x + y)$  dapat dirampungkan dengan algoritma genetika. Penyelesaian masalah ini dituntaskan oleh algoritma genetika yang diimplementasikan pada suatu program komputer dengan bahasa python. Fungsi dari permasalahan tersebut diberikan batasan pada nilai x dan y, yakni nilai x dengan batasan  $-1 \leq x \leq 2$  dan nilai y dengan batasan  $-1 \leq y \leq 1$ . Parameter kontrol dari permasalahan ini, yaitu jumlah generasi sebanyak 20 generasi, probabilitas crossover (Pc) sebesar 0,65, dan probabilitas mutasi (Pm) sebesar 0,01. Kriteria berhenti yang digunakan oleh kami adalah jumlah generasi = 20 dengan 10 kali percobaan algoritma genetika. Hasil analisis dari 10 kali percobaan mendapatkan suatu nilai yang konvergen ke nilai maksimum dengan nilai x mendekati 2 dan nilai y mendekati 1, tentunya setiap menjalankan program ini akan menghasilkan nilai x dan nilai y yang berbeda (random) sesuai cara kerja komputer.



```
10_IF4310_1301194201.py X Python - Get Started
C: > Users > Fiyona > Downloads > 10_IF4310_1301194201.py > ...
1 import random
2 import math
3
4 JUMLAH_POPULASI = 50 #jumlah populasi antara 50-100
5 MUTASI_RATE = 0.01 #mutasi rate antara ~1%
6 CROSSOVER_RATE = 0.65 #crossover rate antara 60% - 70%
7 JUMLAH_GEN = 16 #jumlah gen setiap kromosom (x,y)
8 JUMLAHGENERASI = 20 #jumlah generasi (jika terlalu sedikit terjebak lokal optimum, jika besar akan membutuhkan waktu lama)
9
10 #####
11 ##### Kelompok 10 | IF-43-10 #####
12 ##### Anggota : - Bima Mahardika Wirawan (1301194304) #####
13 ##### - M. Aldi Haryojudanto (1301194025) #####
14 ##### - Fiyona Annila Syamsir (1301194201) #####
15 #####
16
17 def bangkitkanPopulasi(n,JUMLAH_GEN):
18     populasi = [[int(random.choice([1,0])) for i in range(JUMLAH_GEN)] for j in range(n)]
19     return populasi
20
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE 2: Python Debug Consc
([0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1], -1.3349940289830258)
=====Hasil=====
([1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0], 2.1940135145875833)
([1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0], 2.1566147647197864)
([1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0], 2.14520133662149)
([1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], 2.1086788148552222)
([1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1], 2.0888021381924227)
([1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0], 2.0772603292562177)
([1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0], 2.045166283301611)
([1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0], 2.020219521192899)
([1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1], 2.002219521192899)
([1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1], 2.0088669207382726)
([1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0], 1.9182069512032476)
=====
Kromosom terbaik :
([1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0], 2.1940135145875833)
(x,y) = (1.9176470588235293, 0.9137254901960783)
=====Selesai=====
PS C:\Users\Fiyona\Downloads>
```

Gambar 18 Screenshot hasil eksekusi program algoritma genetika

Berdasarkan gambar 18, kita dapat melihat bahwa kromosom (solusi) terbaik memiliki nilai x = 1,917 dan nilai y = 0,913 mendekati range yang sudah ditetapkan sehingga terbukti bahwa algoritma genetika mampu memberikan solusi terbaik dalam memecahkan permasalahan nilai maksimum fungsi melalui program ini.

## Referensi

Materi Algoritma Genetika dari Responsi Pengantar Kecerdasan Buatan Telkom University

[http://repository.uin-suska.ac.id/10898/1/2010\\_2010124TIF.pdf](http://repository.uin-suska.ac.id/10898/1/2010_2010124TIF.pdf), diakses 24 Maret 2021

<https://github.com/raisoturu/genetic-algorithm-playground/tree/master/ga-max-function>  
(referensi source code algoritma genetika), diakses 23 Maret 2021

## Link video presentasi (video berbeda antara mahasiswa)

Bima Mahardika Wirawan (1301194304)

<https://youtu.be/EbKfuWx4ctM>

*(Algoritma Genetika Mencari Nilai Maksimum dari Suatu Fungsi | AI | IF-43-10 | Kelompok 10)*

Fiyona Anmila Syamsir (1301194201)

<https://youtu.be/xjDU8xwkfik>

*(Algoritma Genetika Mencari Nilai Maksimum dari Suatu Fungsi | AI | IF-43-10 | Kelompok 10)*

M. Aldi Haryojudanto (1301194025)

[https://youtu.be/LVaCdBq\\_ySw](https://youtu.be/LVaCdBq_ySw)

*(Algoritma Genetika Mencari Nilai Maksimum dari Suatu Fungsi | AI | IF-43-10 | Kelompok 10)*