LAPORAN TUGAS MATA KULIAH
PENGANTAR KECERDASAN BUATAN

Disusun oleh:

Muhammad Arya Fikriansyah 1301204066

Rifky Fahrizal Ubaidillah 1301204054

PROGRAM STUDI S1 INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
BANDUNG

2022

DAFTAR ISI

PENDAHULUAN

Algoritma Genetika merupakan suatu algoritma yang pendekatannya dilakukan melalui kajian sistem genetika kehidupan. Dimana berbagai kegiatan genetik seperti persilangan dan mutasi merupakan salah satu operasi pada Algoritma Genetika ini. Memanfaatkan teknik randomisasi dan persilangan serta mutasi, algoritma genetik menjadi suatu teknik heuristic yang dapat menemukan suatu solusi dengan cukup cepat.

John Holland dari Universitas Michigan pertama kali memperkenalkan Algoritma Genetika di Michigan pada awal 1970 dengan tulisannya berjudul *Adapted in Natural and Artificial System* yang cara kerjanya berdasarkan pada seleksi dan genetika alam. Sedangkan aplikasi pertamanya pada *manufacturing control* dikemukakan oleh L Davids dalam Proceedings of an International Conference on Genetic Algorithm and their Application Hillsdale 1985. Algoritma Genetika bekerja dari satu populasi bukan dari satu titik dan mencari nilai optimum secara keseluruhan.

Algoritma Genetika merepresentasikan individu sebagai sebuah kromosom. Algoritma ini menyelesaikan permasalahan dalam pencarian kromosom yang terbaik dengan memanipulasi isi di dalam kromosom tanpa tahu permasalahan yang sedang diselesaikan seperti yang terjadi di alam. Informasi yang diberikan hanya evaluasi dari tiap kromosom yang dihasilkan, dan digunakan untuk membelokkan seleksi dari kromosom sehingga kromosom yang terbaik yang terpelihara untuk dikembangkan lebih banyak.

Algoritma genetika merupakan algoritma yang berdasarkan pada mekanisme dan seleksi alam dan mempunyai 5 komponen yaitu :

1. Representasi genetika untuk solusi potensial permasalahan.
2. Metode untuk membuat populasi awal dari solusi potensial.
3. Nilai untuk parameter yang bervariasi : jumlah kromosom, banyaknya gen dalam kromosom, laju mutasi, dan laju *crossover*.
4. Operator-operator genetika : mutasi dan perkawinan silang (*crossover*).
5. Evaluasi.

ANALISIS DAN DESAIN

1. Desain kromosom dan metode dekode-nya

Panjang kromosom yang digunakan dalam tugas pemrograman ini adalah 8 gen, tersusun dari x dan y. Keduanya memiliki panjang masing-masing 4 gen. Desain Kromosom ialah pemilihan kandidat secara probabilistik dengan tujuan guna direproduksi, sementara itu gen tiap orang tua akan diturunkan ke anaknya.

Prosedur pengkodean yang dipakai pada tugas pemrograman ini merupakan representasi biner. *Decoding* (pengkodean) berguna untuk mengkodekan gen-gen pembentuk individu agar nilainya tidak melebihi *range* yang telah ditentukan dan sekaligus menjadi nilai variabel yang akan dicari sebagai solusi permasalahan.

2. Ukuran populasi

Populasi adalah sekumpulan kromosom. Dalam satu populasi, akan terdapat N buah kromosom dengan nilai N merupakan suatu parameter yang telah ditetapkan. Pada tugas pemrograman ini, kami memakai populasi sebanyak 100 kromosom per generasi yang hendak diambil 2 buah kromosom selaku orang tua.

3. Metode pemilihan orangtua

Pemilihan Orang Tua ialah seleksi orang tua yang hendak diseleksi untuk proses *crossover* serta mutasi dengan tujuan untuk mendapatkan calon induk yang terbaik, sehingga bisa menciptakan keturunan yang baik. Operator reproduksi yang dipakai dalam tugas pemrograman ini adalah operator reproduksi yang berdasarkan *Roulette Wheel selection*, *Roulette Wheel* 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.

4. Metode operasi genetik (pindah silang dan mutasi)

Crossover merupakan proses mengkombinasikan dua individu untuk memperoleh individu-individu baru yang diharapkan mempunyai nilai *fitness* lebih baik. Tidak semua pasangan induk mengalami proses *crossover*, banyaknya pasangan induk yang mengalami *crossover* ditentukan dengan nilai probabilitas *crossover*. Pada tugas pemrograman ini menggunakan *crossover* biner satu titik, dimana satu titik potong akan dipilih secara acak.

Mutasi adalah proses penggantian gen dengan nilai intervensinya. Pada tugas pemrograman ini kami menggunakan mutasi biner. Mutasi biner dilakukan dengan cara sederhana, yaitu dengan mengganti satu atau beberapa nilai gen dari kromosom.

5. Probabilitas operasi genetik (*Pc* dan *Pm*)

Probabilitas *crossover* ini digunakan untuk mengendalikan frekuensi operator *crossover*. Pada tugas pemrograman yang kami buat akan melakukan persilangan sebanyak 1 kali pada generasi ke 2 hingga generasi ke-n. Sehingga didapatkan bahwa probabilitas terjadinya *crossover* adalah 100%. Probabilitas mutasi berguna untuk menentukan berapa banyak kromosom yang harus bermutasi dalam satu generasi; tingkat mutasi berada dalam kisaran [0, 1]. Pada tugas pemrograman ini kami menggunakan *Pm* sebesar 0.5%.

6. Metode pergantian generasi (seleksi survivor).

Seleksi Survivor adalah proses pergantian kromosom menggunakan metode seleksi survivor, pada tugas pemrograman ini kami menggunakan *Steady State Model*. *Steady State Model* adalah metode dimana di dalam populasi, tidak semua kromosom diganti. Penggantian dilakukan hanya pada sejumlah kromosom tertentu. Kami membuat agar setiap generasi memiliki populasi yang sama yaitu 100 kromosom. Kami memilih *survivor* dengan cara membuang kromosom dengan nilai *fitness* terkecil di dalam populasi, sehingga menyisakan kromosom dengan nilai *fitness* terbaik.

7. Kriteria penghentian evolusi (*loop*).

Pada tugas pemrograman ini, evolusi akan berhenti pada saat kandidat yang sudah terpilih selalu sama dengan kandidat kromosom untuk seleksi orang tua dan generasi sudah mencapai generasi ke 100. Hal tersebut terjadi karena jika kromosom orang tua yang sudah terpilih selalu sama dengan kromosom yang menjadi kandidat orang tua, maka kode akan terus melakukan perulangan pada bagian *looping* dengan kondisi "*len(parent)!=k*".

IMPLEMENTASI

1. Dekode kromosom

Metode pengkodean yang digunakan pada tugas pemrograman ini adalah representasi biner, dimana kromosom berbentuk himpunan bilangan yang bernilai 0 atau 1. Bilangan-bilangan biner akan merepresentasikan nilai x dan nilai y yang akan dimasukkan ke dalam rumus

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

dengan domain (batas nilai) untuk x dan y: $-5 \leq x \leq 5$ dan $-5 \leq y \leq 5$. Untuk menghitung representasi biner menggunakan rumus representasi biner.

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

Kode program:

```
def Representation(self, ra, rb, g):
    '''
    Fungsi ini digunakan untuk menghitung nilai representasi
    Representasi yang digunakan yaitu Representasi biner 1 titik
    '''
    rn = [2**-i for i in range(1, len(g) + 1)]                          #rentang nilai
    return rb + ((ra - rb) / sum(rn) * sum([g[i] * rn[i] for i in range(len(g))]))  #Rumus representasi biner
```

2. Perhitungan *fitness*

Perhitungan nilai *fitness* pada program yang kami buat menggunakan fungsi objektif maksimasi dimana nilai fitness adalah nilai dari fungsi itu sendiri, fungsi fitness maksimasi dapat dituliskan sebagai :

$$f = h$$

Kode program:

```
def fitness(x, y):
    '''
    Fungsi ini digunakan untuk menghitung nilai fitness
    Fungsi fitness yang digunakan adalah maksimum f = h
    '''
    return heuristic(x, y)
```

3. Pemilihan orang tua

Operator reproduksi yang dipakai dalam tugas pemrograman ini adalah operator reproduksi yang berdasarkan *Roulette Wheel selection.*

Kode program:

```
def parentselection(k):
    '''
    Fungsi ini digunakan untuk seleksi orangtua
    Fungsi ini akan mereturn orangtua dari metode Roulette wheel
    '''
    parent = []                                                         #List atau array orangtua
    list_fitness = list(map(lambda c: fitness(c.x, c.y), populasi))     #Memakai fungsi lambda sebagai anonymous function
    list_weight = [list_fitness[i] / sum(list_fitness) for i in range(len(populasi))] #List atau array untuk menampung weight dari semua kromosom yang ada di populasi
    while len(parent) != k:
        kandidat = random.choices(populasi, weights=list_weight)[0]     #Parameter weight akan memberi berat kemungkinan pada setiap nilai
        if not exist(parent, kandidat):                                 #Sehingga setiap item untuk dipilih ditentukan oleh bobot relatifnya.
            parent.append(kandidat)
    return parent
```

4. *Crossover* (pindah silang)

Metode yang digunakan dalam program adalah metode *crossover* paling sederhana yaitu pindah silang satu titik potong (*1-point crossover*). Penentuan posisi titik potong dilakukan secara random.

Kode Program:

```
def Crossover(parent1, parent2):
    '''
    Fungsi crossover digunakan untuk menghasilkan anak
    '''
    pos = random.randint(1, len(parent1.biner) - 2)           #mengambil titik potong dari indeks kedua paling awal atau indek kedua paling akhir

    biner_child1 = parent1.biner[:pos] + parent2.biner[pos:]
    biner_child2 = parent2.biner[:pos] + parent1.biner[pos:]
```

5. Mutasi

Pada tugas pemrograman ini kami menggunakan mutasi biner. Mutasi biner dilakukan dengan cara sederhana, yaitu dengan mengganti satu atau beberapa nilai gen dari kromosom. Pada mutasi kami menggunakan probabilitas mutasi sebesar 0.5%.

Kode Program:

```
'''
Mutasi
'''
#Mutasi anak pertama
prob_mutasi = random.uniform(0, 100)                        #memilih angka random dari 0 sampai 100 prob_mutasi dalam persen
if prob_mutasi < 0.5:                                       #0.5% mutasi
    idx_mutasi = random.randint(0, len(biner_child1) - 1)
    if biner_child1[idx_mutasi] == 1:
        biner_child1[idx_mutasi] = 0
    else:
        biner_child1[idx_mutasi] = 1

#Mutasi anak ke 2
prob_mutasi = random.uniform(0, 100)                        #Memilih angka random dari 0 sampai 100 prob_mutasi dalam persen
if prob_mutasi < 0.5:                                       #0.5% mutasi
    idx_mutasi = random.randint(0, len(biner_child2) - 1)
    if biner_child2[idx_mutasi] == 1:
        biner_child2[idx_mutasi] = 0
    else:
        biner_child2[idx_mutasi] = 1

#Memasukan hasil crossover dan mutasi ke populasi
populasi.append(chromosome(biner_child1))
populasi.append(chromosome(biner_child2))
```

6. Pergantian Generasi

Proses pergantian generasi dimulai dari seleksi orangtua, yaitu memilih M kromosom untuk diletakkan ke dalam Mating Pool. Kemudian, memasangkan secara acak M kromosom orangtua di dalam Mating Pool sehingga dihasilkan M/2 pasangan orang tua. Kemudian, dengan probabilitas *Pc* yang telah ditentukan setiap pasangan akan direkombinasi. Hasil rekombinasi kemudian dimutasi berdasarkan probabilitas *Pm*. Setelah proses rekombinasi dan mutasi selesai, akan dihasilkan M kromosom baru. M kromosom baru akan menggantikan M kromosom lama yang dibuang jika nilai fitnessnya lebih buruk dari kromosom yang baru.

Kode program:

```
def survivorselection():
    '''
    Fungsi seleksi survivor berfungsi agar populasi terus sama
    '''
    populasi.sort(key=lambda c: heuristic(c.x, c.y), reverse=True)  #Memakai fungsi key dan fungsi lambda sebagai anonymous function di dalam fungsi sort descending

    while len(populasi) != 100:                                     #Mengatur jumlah populasi agar tetap 100
        populasi.pop()                                              #Membuang kromosom yang paling buruk di populasi
```

KESIMPULAN

Dari percobaan yang telah dilakukan pada tugas pemrograman ini kami mendapatkan kesimpulan bahwa *Genetic Algorithm* (GA) dapat menemukan nilai optimum suatu fungsi, namun dapat pula menghasilkan nilai yang tidak optimum apabila nilai parameternya tidak sesuai. Oleh karena itu, nilai-nilai parameter yang digunakan haruslah pas dan sesuai.

Untuk dapat mengetahui bahwa program *Genetic Algorithm* (GA) yang telah dibuat dapat menghasilkan nilai optimum yang diinginkan dibutuhkan beberapa kali percobaan. Dari percobaan beberapa kali yang telah dilakukan, nilai optimum yang dihasilkan oleh program terkadang berbeda. Hal tersebut disebabkan karena program memilih secara acak kromosom-kromosom yang akan direkombinasi atau dimutasi sehingga hasil yang didapatkan tidak akan selalu sama.

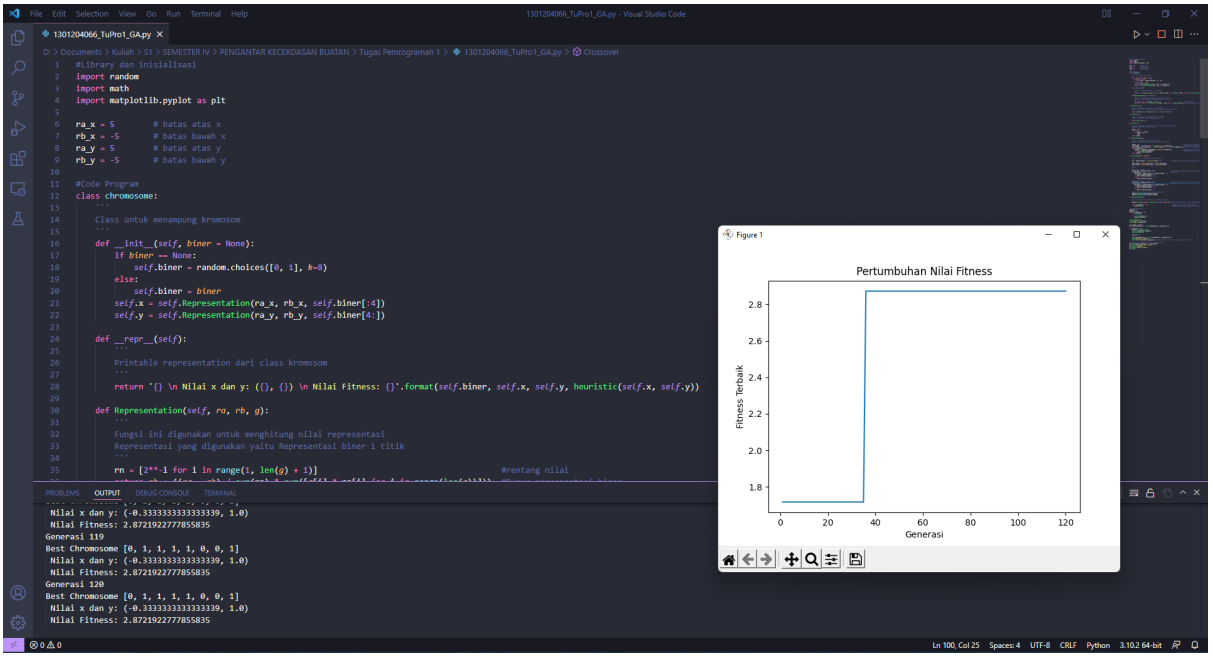
LAMPIRAN

Peran Anggota Kelompok

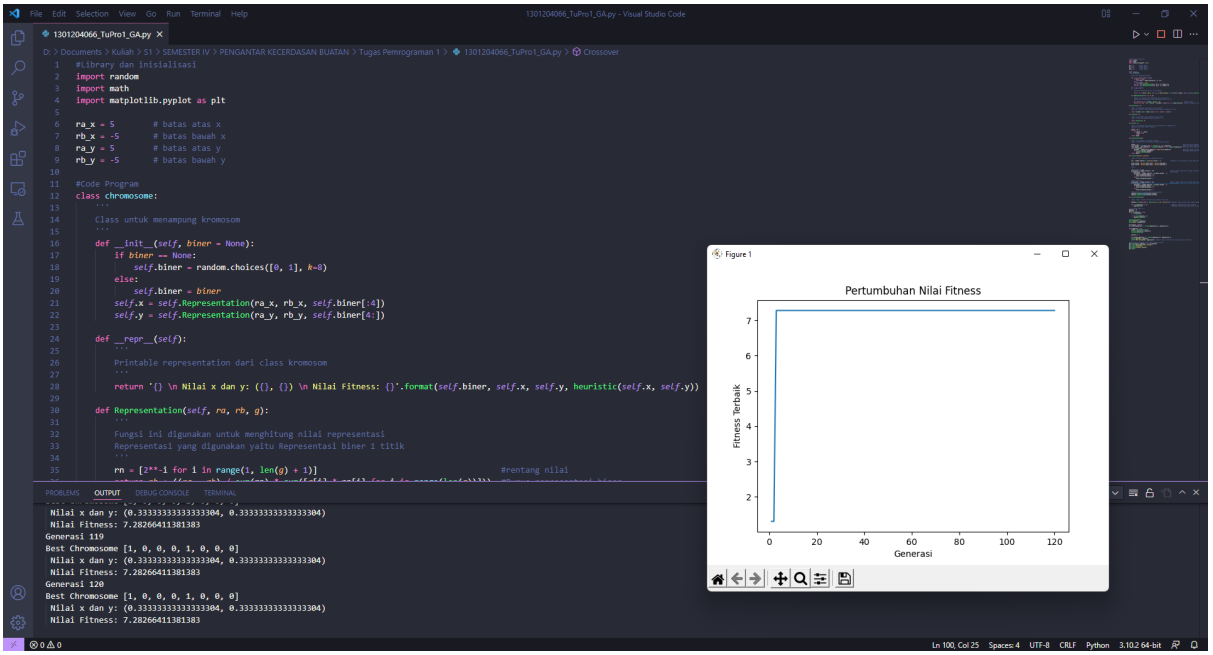
Nama	NIM	Peran
Muhammad Arya Fikriansyah	1301204066	Coding, Laporan bagian analisis desain dan implementasi.
Rifky Fahrizal Ubaidillah	1301204054	Coding, Laporan bagian pendahuluan dan kesimpulan.

Hasil Percobaan

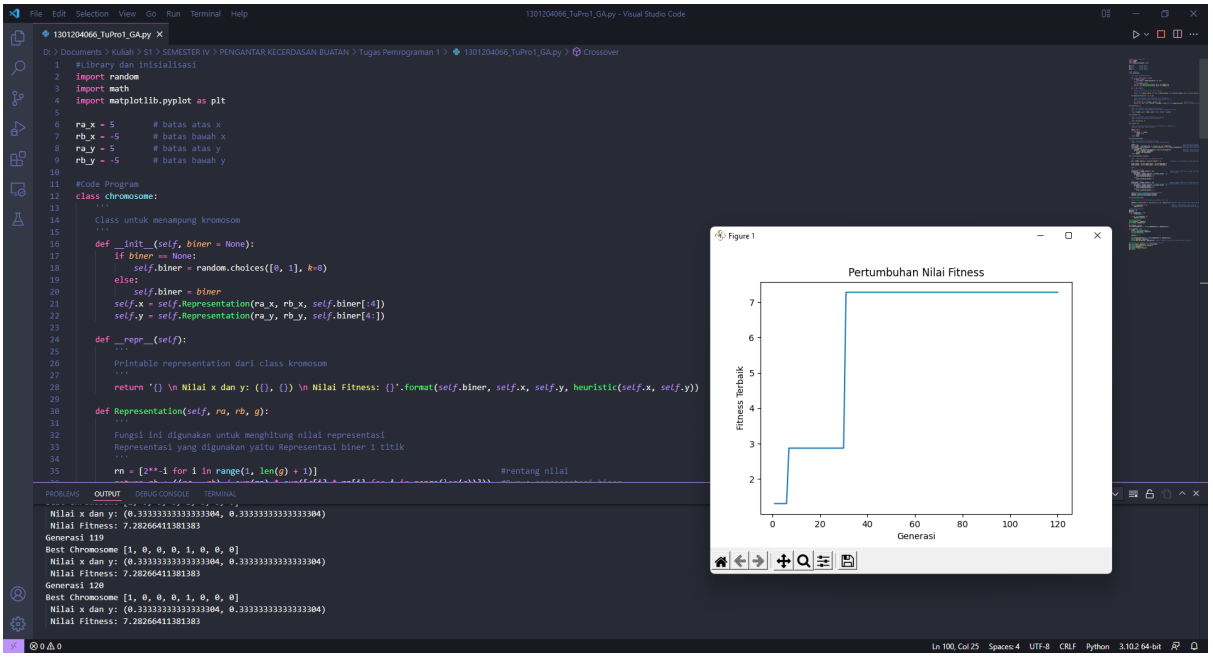
Percobaan 1:



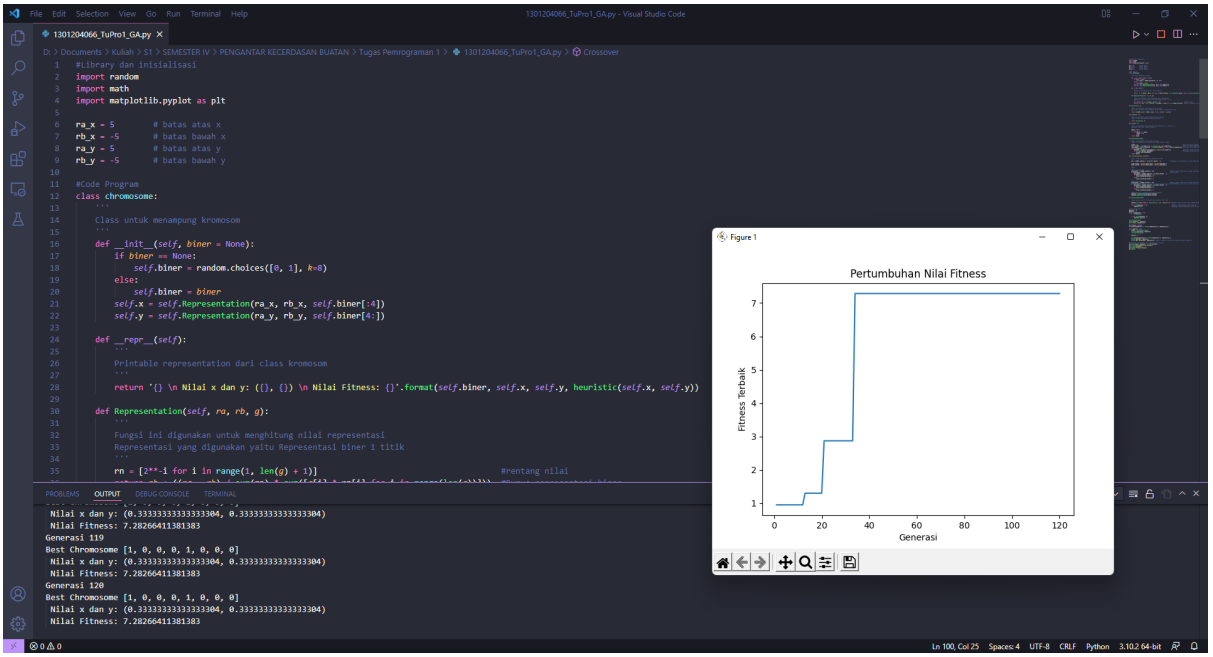
Percobaan 2:



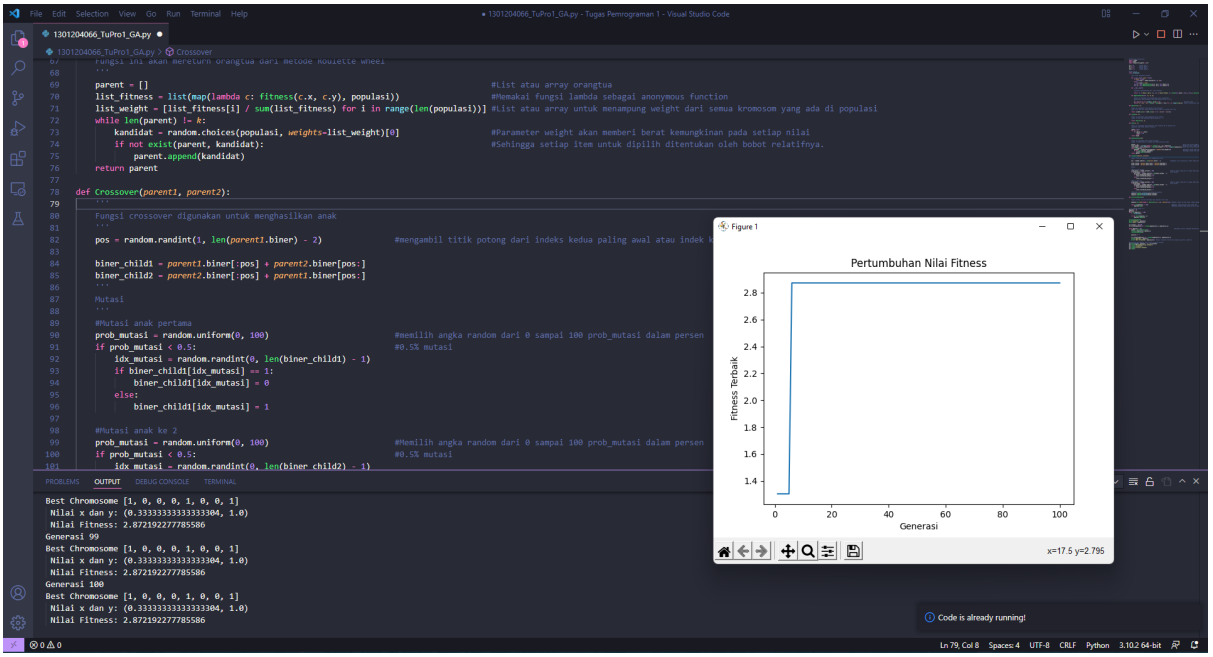
Percobaan 3:



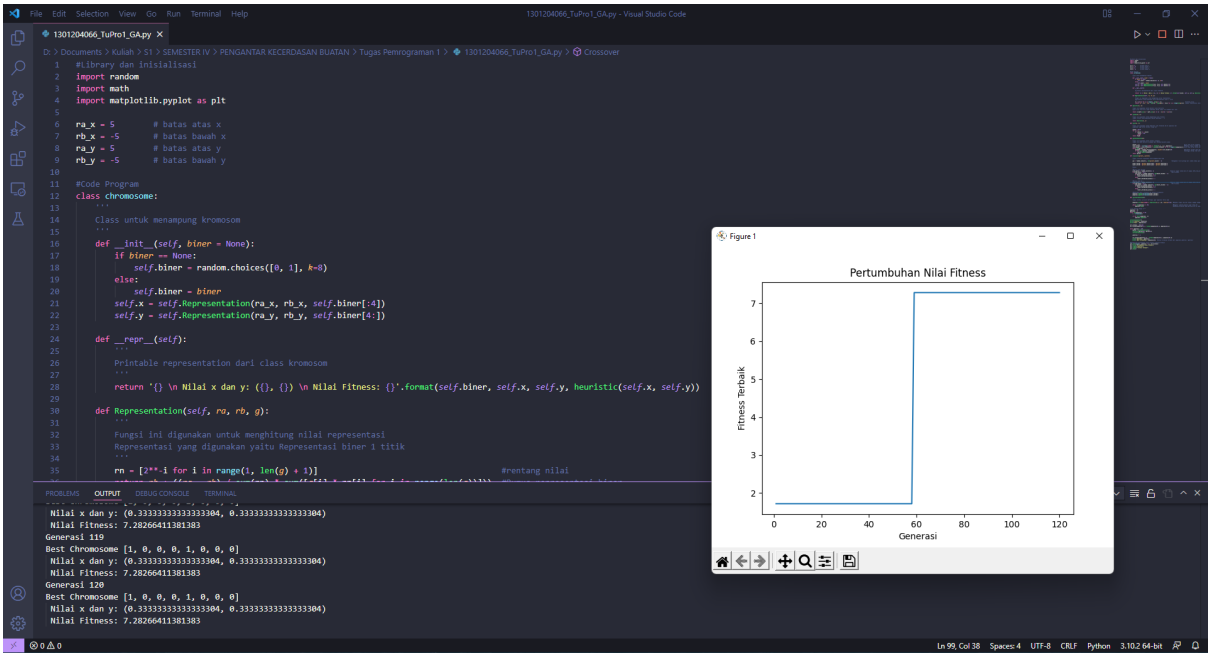
Percobaan 4:



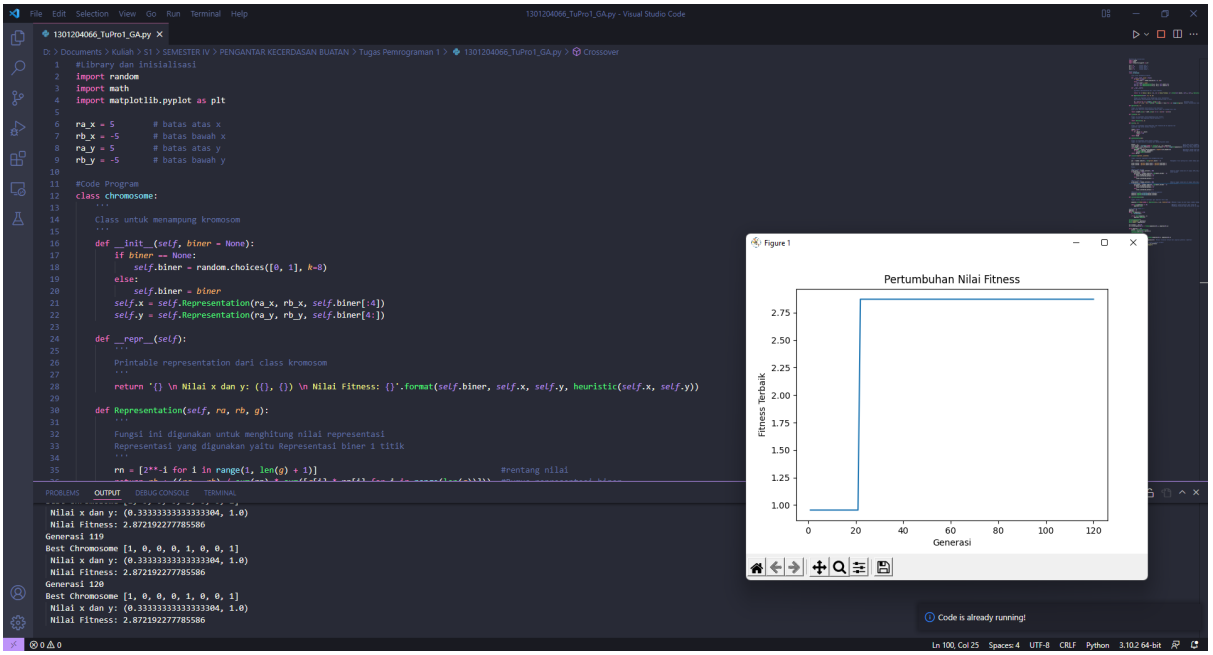
Percobaan 5:



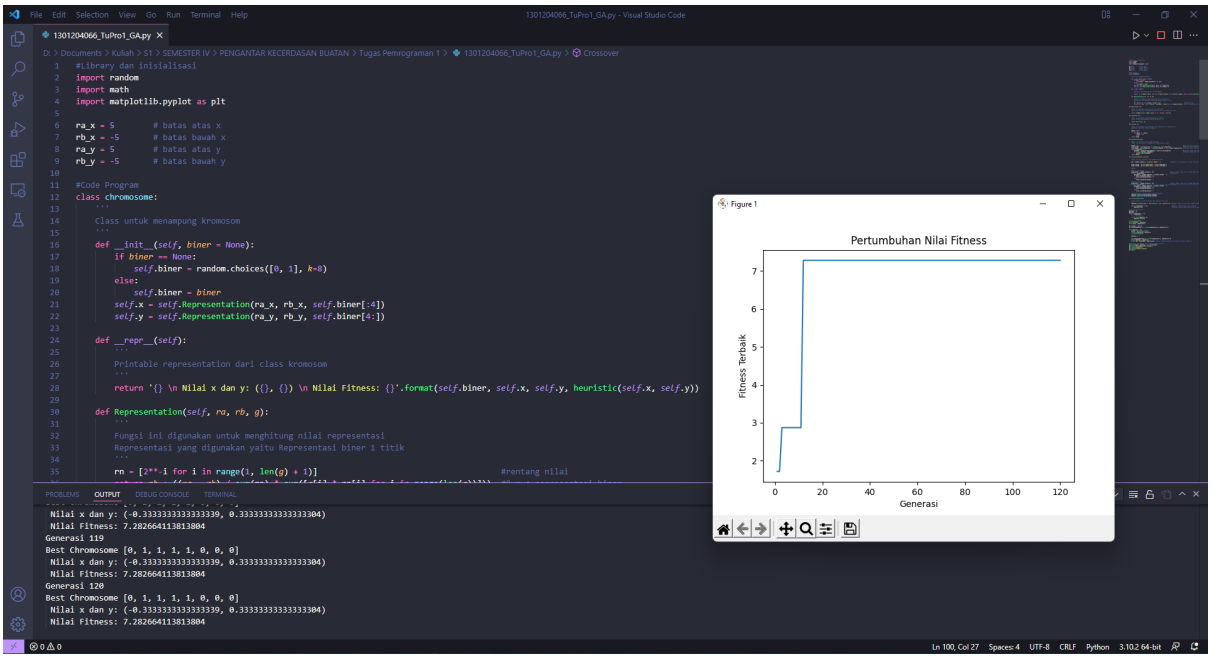
Percobaan 6:



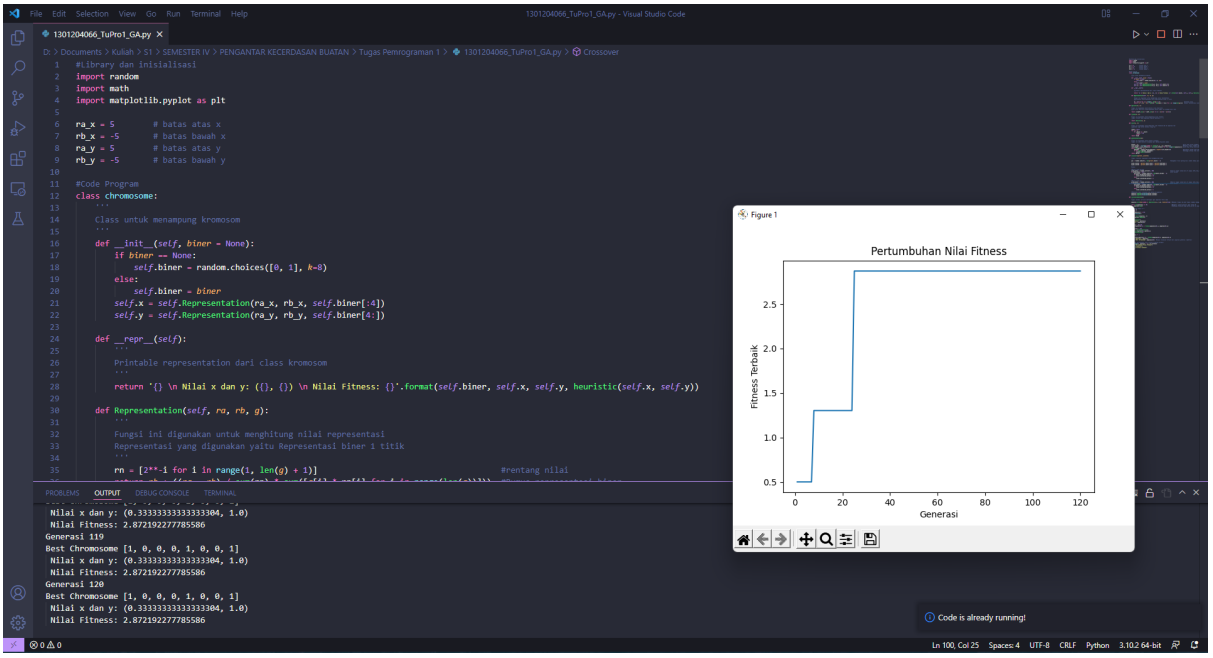
Percobaan 7:



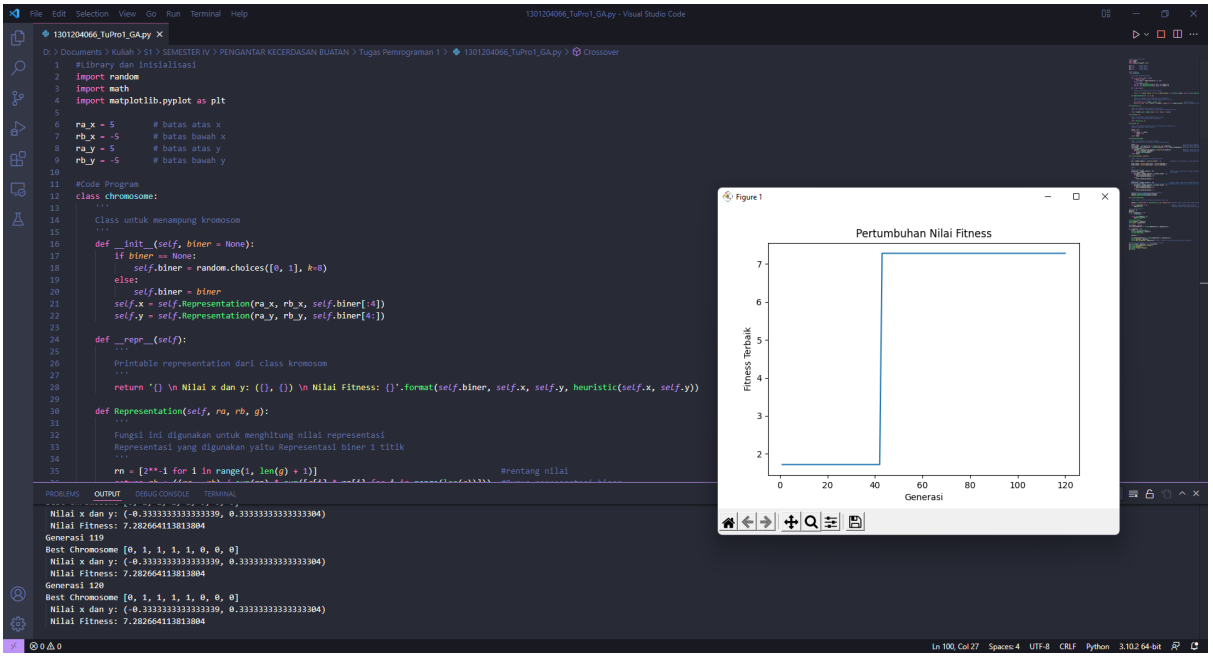
Percobaan 8:



Percobaan 9:



Percobaan 10:



DAFTAR PUSTAKA

Mitchell, 1999, *An Introduction to Genetic Algorithms*, A Bradford Book The MIT Press, Massachusetts.

Davis, Lawrence, 1991, *Handbook of Genetic Algorithms*, Von Nostrand Reinhold, New York.

Goldberg, David E, 1989, *Genetic Algorithm in Search*, Optimization and Machine Learning, Addison Wesley Publishing Company, MA.

Adhy, Kushartantya, D 2010, '*Penyelesaian Masalah Job Shop Menggunakan Algoritma Genetika*', Jurnal Masyarakat Informatika*,* Vol. 1, No. 1, hh 31-33.

Putra, I. M. S, 2018. '*Penerapan Algoritma Genetika Dan Implementasi Dalam MATLAB*', vol, 53, 1689-1699.

Ridwansyah, R, 2017. '*Penggunaan Algoritma Genetika untuk Desain Topologi Mesh pada Jaringan Fisik Telekomunikasi*', Jurnal Matematika, Statistika dan Komputasi, 14(1), 87-92.

Hosea, 2007. *Implementasi Algoritma Genetika dalam Penentuan Kandidat Model*. (Skripsi, Universitas Sanata Dharma, 2007)