LAPORAN OBSERVASI GENETIC ALGORITHM

Pendahuluan

Genetic Algorithm (Algoritma Genetik) adalah metode pencarian *heuristic* yang terinspirasi dari teori Charles Darwin tentang evolusi natural. Algoritma ini merefleksikan proses dari seleksi natural di mana individu-individu yang cocok akan dipilih untuk mereproduksi kembali kromosom generasi selanjutnya.

Deskripsi Kegiatan

Kegiatan yang akan dilakukan dalam observasi ini adalah melakukan analisis, desain, dan mengimplementasikan algoritma genetika (*Genetic Algorithm*) ke dalam suatu program komputer untuk menemukan **nilai minimum** dari fungsi:

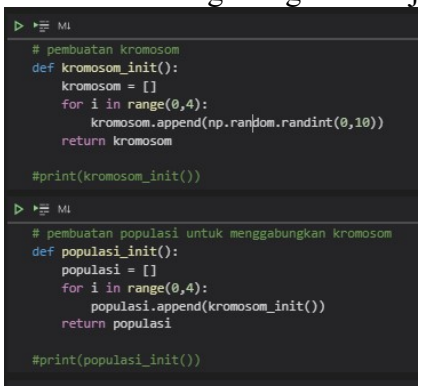
$$h(x_1, x_2) = \cos(x_1)\sin(x_2) - \frac{x_1}{(x_2^2 + 1)}$$

dengan batasan $-1 \leq x_1 \leq 2$ dan $-1 \leq x_2 \leq 1$.

Metode Penyelesaian

Bahasa pemrograman yang digunakan adalah Python dengan menggunakan Jupyter dan Visual Studio Code sebagai *text editor*-nya. Metode penyelesaian yang digunakan untuk menemukan nilai minimum dari fungsi di atas terbagi menjadi beberapa tahap, yaitu:

A. Pembuatan Kromosom dan Populasi
Langkah pertama adalah membuat kromosom lalu digabungkan menjadi suatu populasi.



```
# pembuatan kromosom
def kromosom_init():
    kromosom = []
    for i in range(0,4):
        kromosom.append(np.random.randint(0,10))
    return kromosom

#print(kromosom_init())
```

```
# pembuatan populasi untuk menggabungkan kromosom
def populasi_init():
    populasi = []
    for i in range(0,4):
        populasi.append(kromosom_init())
    return populasi

#print(populasi_init())
```

B. Dekode Kromosom
Dekode kromosom menggunakan rumus khusus untuk kromosom yang berisikan *integer*.

```
def dekode_kromosom(kromosom,r_min,r_max):
    sigma = 0
    sigma_k = 0

    for i in range(len(kromosom_a)-1):
        sigma += 10**-i
        sigma_k += kromosom[i] * sigma
    hasil_dekode = r_min + ((r_max - r_min)/9 * sigma) * sigma_k
    return hasil_dekode
```

```
def div_kromosom(kromosom):
    kromosom_a = []
    kromosom_b = []
    for i in range(0,2):
        kromosom_a.append(kromosom[i])
        kromosom_b.append(kromosom[i+2])
    return kromosom_a, kromosom_b
```

C. Penghitungan *Fitness*
Penghitungan untuk mencari nilai kecocokan kromosom terhadap suatu permasalahan.

```
def fitness(x1,x2):
    return math.cos(x1) * math.sin(x2) - x1/(x2**2 + 1)
```

D. Pemilihan Orangtua
Pemilihan *parent* (orangtua) menggunakan metode *stochastic roulette*.

```
def pilih_parent(semua_fitness):
    # stochastic roulette
    cek = True
    fitness_maks = max(semua_fitness)
    while(cek):
        indeks = round(np.random.uniform()*len(semua_fitness)-1)
        r = np.random.uniform()
        if (r < semua_fitness[indeks]/fitness_maks):
            cek = False
    return indeks;
```

E. *Crossover* (Pindah Silang)
Proses *crossover* (pindah silang) untuk membentuk dua kromosom anak baru dari dua *parent* yang terpilih.

```
def crossover(parent_1, parent_2):
    kromosom_a, kromosom_b = div_kromosom(parent_1)
    kromosom_c, kromosom_d = div_kromosom(parent_2)

    child_1 = []
    child_2 = []

    child_1.append(kromosom_a+kromosom_d)
    child_2.append(kromosom_c+kromosom_b)

    return child_1, child_2
```

F. Mutasi
Mutasi dilakukan untuk mengganti suatu gen dengan gen yang baru.

```
def mutasi(child_1, child_2):
    probs = np.random.randint(0,100)
    if probs == 1:
        indeks = np.random.randint(0,len(child_1)-1)
        child_1[indeks] = np.random.randint(0,10)
    elif probs == 2:
        indeks = np.random.randint(0,len(child_2)-1)
        child_2[indeks] = np.random.randint(0,10)
```

G. Pergantian Generasi
Proses penggantian generasi (menggunakan metode seleksi *fitness based selection*) untuk memilih kromosom terbaik untuk selanjutnya akan dijadikan populasi untuk generasi berikutnya.

```
def generasi_baru(semua_fitness, populasi, semua_child):
    # proses ini menggunakan metode seleksi fitness based selection
    kromosom_baru = []
    populasi_baru = []

    if (semua_fitness[0] < semua_fitness[1]):
        min_a = 0
        min_b = 1
    else:
        min_a = 1
        min_b =0

    for i in range(2, len(semua_fitness)):
        if (semua_fitness[i] < semua_fitness[min_b]):
            if(semua_fitness[i] < semua_fitness[min_a]):
                min_b = min_a
                min_a = i
            else:
                min_b = i

    for j in range(len(populasi)):
        if (j == min_a):
            kromosom_baru.append(semua_child[0])
            pop = []
            child_a, child_b = div_kromosom(semua_child[0])
            pop.append(dekode_kromosom(child_a,r1_min,r1_max))
            pop.append(dekode_kromosom(child_b,r2_min,r2_max))
            populasi_baru.append(pop)

        elif (j == min_b):
            kromosom_baru.append(semua_child[1])
            pop = []
            child_a, child_b = div_kromosom(semua_child[1])
            pop.append(dekode_kromosom(child_a,r1_min,r1_max))
            pop.append(dekode_kromosom(child_b,r2_min,r2_max))
            populasi_baru.append(pop)

        else:
            kromosom_baru.append(semua_child[j])
            pop = []
            child_a, child_b = div_kromosom(semua_child[j])
            pop.append(dekode_kromosom(child_a,r1_min,r1_max))
            pop.append(dekode_kromosom(child_b,r2_min,r2_max))
            populasi_baru.append(pop)

    return populasi_baru, kromosom_baru
    print(semua_child[1])
```

Kesimpulan

Kesimpulan dari *Genetic Algorithm* ini adalah algoritma ini dapat digunakan untuk pengoptimasian baik minimum ataupun maksimum pada suatu permasalahan dalam kehidupan sehari-hari.