Laporan – Observasi Tugas Pararel 1
Kecerdasan Buatan 2019/2020
Hariadi Adha Firmansyah – 1301174252 – IF-41-10

- Desain kromosom dan Teknik mendekodekannya

```
In [3]: def getIndividu(nIndividu):
            box = []
            for i in range(0,nIndividu):
                box.append(np.random.randint(0,
            return box
```

Fungsi tersebut digunakan untuk membuat satu individu dengan mengambil nilai alel random

```
In [4]: def getPopulasi(nPopulasi, nIndividu):
            boxPop = []
            for i in range(0,nPopulasi):
                boxPop.append(getIndividu(nIndiv
            return boxPop
```

fungsi tersebut digunakan untuk membuat populasi yang didalamnya terdapat beberapa individu dengan nilai alel random

```
In [5]: def genPhenotype(x1,x2, individu):
            cell = nIndividu//2
            tempCell = cell+1
            pangkat = 0
            for i in range(1,tempCell):
                pangkat = pangkat + 10**(-i)
            blok1 = []
            for i in range(0,cell):
                blok1.append(individu[i])
            blok2 = []
            for i in range(cell,nIndividu):
                blok2.append(individu[i])

            jum1 = 0
            jum2 = 0
            for i in range(1,tempCell):
                jum1 = jum1 + blok1[i-1]*(10**(-i))
            #print("jum1",jum1)
            for i in range(1,tempCell):
                jum2 = jum2 + blok2[i-1]*(10**(-i))
            #print("jum2",jum2)

            nx1 = x1[0] + ((x1[1] - x1[0]) / (9 * pangkat)
            nx2 = x2[0] + ((x2[1] - x2[0]) / (9 * pangkat)

            return nx1,nx2
```

Fungsi genPhenotype adalah fungsi dari decode yang menggunakan rumus decode bilangan integer. Didalam decode tersebut saya membagi tiap hitungan mulai dari memecah individu jadi 2 bagian, kemudian dihitung dengan batas yang telah ditentukan di soal.

- Ukuran Populasi

```
In [23]: #main Program
         nPopulasi = 150
         nIndividu = 25
```

ukuran populasi pada program yang saya buat adalah 150 populasi. Dikarenakan untuk mendapatkan kemungkinan nilai terbaik agar lebih akurat dan setelah beberapa kali percobaan.

- Teknik Pemilihan calon orangtua

```
In [8]: # Parent Selection
        def RouletteWheel(fit):
            total = 0
            for i in range(len(fit)):
                total = total + fit[i]

            r = np.random.uniform(0,1)
            individu = 0
            i = 0
            while r>0:
                r = r - (fit[i]/total)
                individu = individu+1
                i = i+1
            return individu-1
```

Pada Teknik pemilihan orangtua, saya memilih metode roullettewheel, pada fungsi tersebut diperlukan nilai fitness dari satu populasi. Kemudian hasil hitungan pada fungsi roulettewheel akan mengoutputkan nilai indeks, maka indeks yang dikeluarkan oleh rouletewheel lah yang akan menjadi calon orangtua.

- Pemilihan dan Teknik operasi genetic (crossover dan mutasi)

```
In [9]: #crossOver (pindah silang)
        def crossOver(tab1, tab2, probCrossOver):
            if probCrossOver >= np.random.uniform(0,1):
                cellRandom = np.random.randint(2,6)
                cell1 = cellRandom
                temp = tab1[cellRandom:]
                tab1[cellRandom:] = tab2[cellRandom:]
                tab2[cellRandom:] = temp
            return tab1, tab2
```

Pada crossover, algoritma yang saya buat adalah pertama saya membuat probabilitas terjadinya mutase terlebih dahulu. Jika memenuhi syarat, maka saya akan menukar dengan cara mengambil alel setelah indeks ke N random(cellRandom) ke variable temp. kemudian nilai dari tab1 setelah indeks ke N random(cellRandom) akan digantian oleh alel dari tab2 setelah indeks ke N random. Setelah itu nilai dari tab2

setelah indeks ke N random, akan di isi atau di replace oleh temp.

```
In [10]: #mutasi
def probMutation(tabel1, probMut):
    for i in range(len(tabel1)):
        if probMut>=np.random.uniform(0,1):
            getValue1 = np.random.randint(0,9)
#             getValue1 = abs(tabel1[i]-9)
            tabel1[i] = getValue1
    return tabel1
```

pada mutasi, algoritma yang saya buat adalah melakukan perulangan sebanyak Panjang individu yang disimpan pada tabel1. Kemudian akan dilakukan pengecekan probabilitas. Jika lolos, maka saya akan meminta nilai random dari 0 sampai 9 yang nantinya akan dijadikan nilai mutase. Dan semua alel jika probabilitas tinggi akan memiliki peluang untuk terganti nilainya dengan angka random.

- Nilai probabilitas operasi genetic (Pc dan Pm)

```
probCrossOver = 0.9999
probMut = 0.6
```

Untuk nilai probabilitas pc saya men set dengan 0,9999 dan probabilitas 0,6. Dikarenakan setelah dilakukan berbagai percobaan dengan banyak angka, kedua angka tersebut adalah angka yang paling mendekati atau memiliki kedekatan yang cukup dekat dengan hasil akhir.

- Metode pemilihan generasi baru

```
In [26]: #main Program
nPopulasi = 150
nIndividu = 25
probCrossOver = 0.9999
probMut = 0.6
mainPopulation = getPopulasi(nPopulasi, nIndividu)
maxGen = 1000
for i in range(maxGen):
    mainFit = calculateFit(mainPopulation)
    newPopulation = elitisme(mainFit,mainPopulation)
    while len(newPopulation) < nPopulasi:
        p1 = mainPopulation[RouletteWheel(mainFit)].copy()
        p2 = mainPopulation[RouletteWheel(mainFit)].copy()
        newChild1, newChild2 = crossOver(p1, p2, probCross
        newChild1 = probMutation(p1, probMut)
        newChild2 = probMutation(p2, probMut)
        newPopulation.append(newChild1)
        newPopulation.append(newChild2)
#     clear_output(wait=True)
#     print("gen ",i)
#     print(*list(zip(mainPopulation, mainFit)),sep="\n")
    mainPopulation = newPopulation
# #     print(*mainFit,sep="\n")
hasilfit = calculateFit(mainPopulation)
x1=[-3,3]
x2=[-2,2]
print(genPhenotype(x1,x2,mainPopulation[0]),hasilfit[0])
```

Pada metode pemilihan generasi baru, saya menggunakan metode generational replacement. Pada metode tersebut pertama saya mengenerate semua fungsi yang telah saya buat, kemudian saya masukan kedalam perulangan sebanyak gen yang telah saya set, yaitu 1000 generasi. Ditiap perulangan saya mereplace hasil populasi baru ke newPopulation. Dan menghitung hasil fitness terakhir, nilai x1 dan x2 nya.

- Kriteria pemberhentian generasi

```
([0, 0, 8, 3, 3, 7, 7, 2, 0, 0], -93.12556330971756)
([1, 3, 7, 7, 5, 4, 7, 7, 3, 8], -7.337149529382617)
(-0.08955089550895501, 0.7105471054710555) 1.0315924096280253
```

Kriteria pemberhentian jika telah selesai melakukan perulangan sebanyak 1000 gen, hasil akhir dari program ini berbeda beda namun angkanya selalu mendekati, untuk x1 = ±-0,08955 untuk x2 = ±0,71054 dan nilai fitness terbaiknya = ±1,03159.