LAPORAN OBSERVASI TUGAS PARALEL 1 : GENETIC ALGORITHM

CLARISA HASYA YUTIKA | 1301174256 | IF 41 02

1. Deskripsi Masalah

Genetic algorithm adalah algoritma yang terinspirasi oleh biologi evolusi seperti warisan, mutasi, seleksi, dan crossover. Salah satu fungsi dari genetic algorithm adalah untuk mencari nilai minimum dari fungsi matematika yang sulit untuk dipecahkan oleh manusia

2. Fungsi yang dibangun

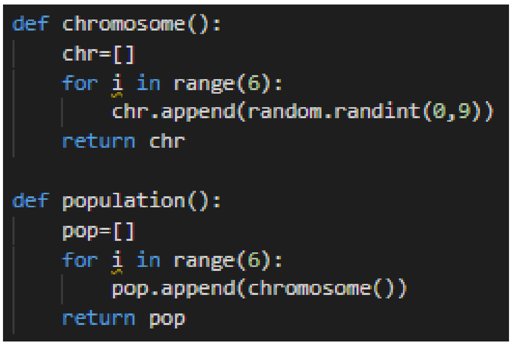
$$f(x_1, x_2) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$$

dengan batasan $-3 \leq x_1 \leq 3$ dan $-2 \leq x_2 \leq 2$.

3. Analisis Algoritma

- Dekode Kromosom

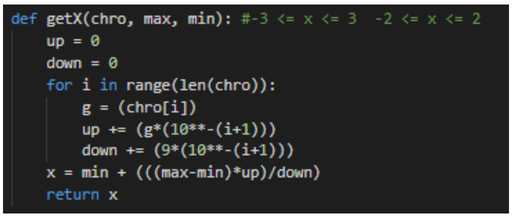
  Membuat list kromosom dengan panjang 6 dengan bilangan integer random antara 0 sampai 9. Lalu membuat list populasi sebanyak 6 kromosom



```
def chromosome():
    chr=[]
    for i in range(6):
        chr.append(random.randint(0,9))
    return chr

def population():
    pop=[]
    for i in range(6):
        pop.append(chromosome())
    return pop
```

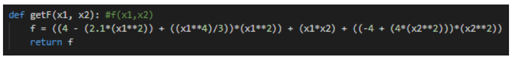
  Lalu decode kromosom untuk x1 dan x2



```
def getX(chro, max, min): #-3 <= x <= 3  -2 <= x <= 2
    up = 0
    down = 0
    for i in range(len(chro)):
        g = (chro[i])
        up += (g*(10**-(i+1)))
        down += (9*(10**-(i+1)))
    x = min + (((max-min)*up)/down)
    return x
```

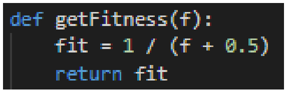
- Perhitungan Fitness

  Untuk mendapatkan nilai fitness, harus menghitung nilai fungsi x1 x2 terlebih dahulu



```
def getF(x1, x2): #f(x1,x2)
    f = ((4 - (2.1*(x1**2)) + ((x1**4)/3))*(x1**2)) + (x1*x2) + ((-4 + (4*(x2**2)))*(x2**2))
    return f
```

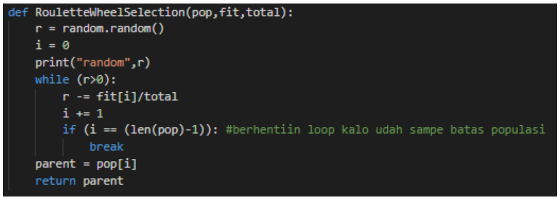
  Lalu masukkan ke dalam rumus fitness 1/(f+h), dengan h=0.5



```
def getFitness(f):
    fit = 1 / (f + 0.5)
    return fit
```

- Pemilihan Orang Tua

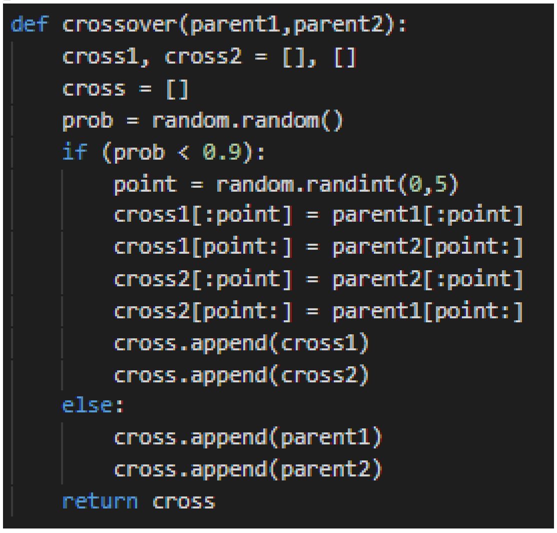
  Dalam pemilihan orang tua dapat menggunakan beberapa cara, salah satunya Roulette Wheel Selection



```
def RouletteWheelSelection(pop,fit,total):
    r = random.random()
    i = 0
    print("random",r)
    while (r>0):
        r -= fit[i]/total
        i += 1
        if (i == (len(pop)-1)): #berhentiin loop kalo udah sampe batas populasi
            break
    parent = pop[i]
    return parent
```

- Crossover

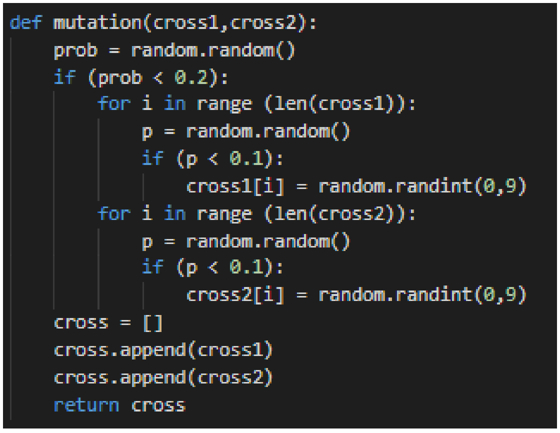
  Hasil dari pemilihan orang tua kemudian di crossover dengan probabilitas 0.7



```
def crossover(parent1,parent2):
    cross1, cross2 = [], []
    cross = []
    prob = random.random()
    if (prob < 0.9):
        point = random.randint(0,5)
        cross1[:point] = parent1[:point]
        cross1[point:] = parent2[point:]
        cross2[:point] = parent2[:point]
        cross2[point:] = parent1[point:]
        cross.append(cross1)
        cross.append(cross2)
    else:
        cross.append(parent1)
        cross.append(parent2)
    return cross
```

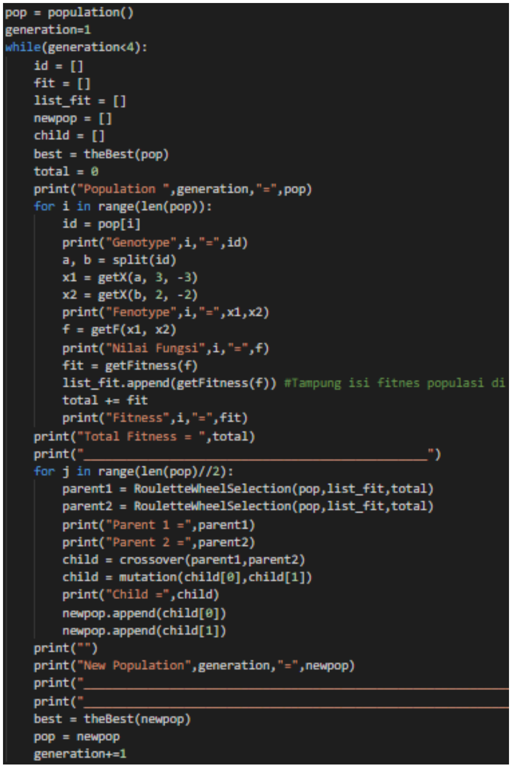
- Mutasi

  Hasil dari crossover kemudian di mutasi dengan probabilitas 0.2, kemudian akan di cek per allel dengan probabilitas 0.1 untuk nilai nya diganti dengan bilangan integer random antara 0 sampai 9



```
def mutation(cross1,cross2):
    prob = random.random()
    if (prob < 0.2):
        for i in range (len(cross1)):
            p = random.random()
            if (p < 0.1):
                cross1[i] = random.randint(0,9)
        for i in range (len(cross2)):
            p = random.random()
            if (p < 0.1):
                cross2[i] = random.randint(0,9)
    cross = []
    cross.append(cross1)
    cross.append(cross2)
    return cross
```

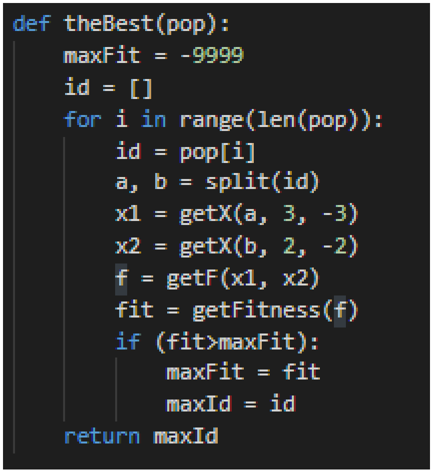
- Pergantian Generasi



```
pop = population()
generation=1
while(generation<4):
    id = []
    fit = []
    list_fit = []
    newpop = []
    child = []
    best = theBest(pop)
    total = 0
    print("Population ",generation,"=",pop)
    for i in range(len(pop)):
        id = pop[i]
        print("Genotype",i,"=",id)
        a, b = split(id)
        x1 = getX(a, 3, -3)
        x2 = getX(b, 2, -2)
        print("Fenotype",i,"=",x1,x2)
        f = getF(x1, x2)
        print("Nilai Fungsi",i,"=",f)
        fit = getFitness(f)
        list_fit.append(getFitness(f)) #Tampung isi fitnes populasi di
        total += fit
        print("Fitness",i,"=",fit)
    print("Total Fitness = ",total)
    print("________________________________________________")
    for j in range(len(pop)//2):
        parent1 = RouletteWheelSelection(pop,list_fit,total)
        parent2 = RouletteWheelSelection(pop,list_fit,total)
        print("Parent 1 =",parent1)
        print("Parent 2 =",parent2)
        child = crossover(parent1,parent2)
        child = mutation(child[0],child[1])
        print("Child =",child)
        newpop.append(child[0])
        newpop.append(child[1])
    print("")
    print("New Population",generation,"=",newpop)
    print("____________________________________________________________
    print("____________________________________________________________
    best = theBest(newpop)
    pop = newpop
    generation+=1
```

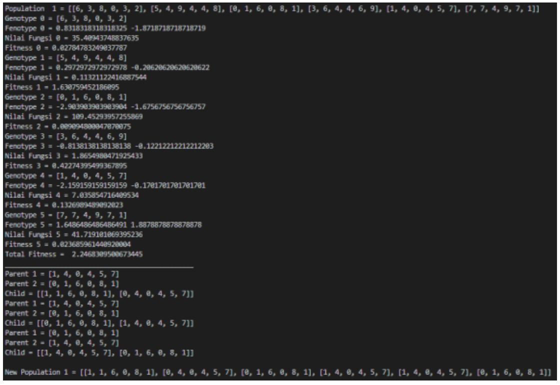
- Kromosom Terbaik

Untuk mendapatkan kromosom terbaik dari semua generasi yang didapatkan adalah membandingkan nilai fitness paling besar dari seluruh kromosom, dan didapatkan nilai fungsi minimal nya



```
def theBest(pop):
    maxFit = -9999
    id = []
    for i in range(len(pop)):
        id = pop[i]
        a, b = split(id)
        x1 = getX(a, 3, -3)
        x2 = getX(b, 2, -2)
        f = getF(x1, x2)
        fit = getFitness(f)
        if (fit>maxFit):
            maxFit = fit
            maxId = id
    return maxId
```

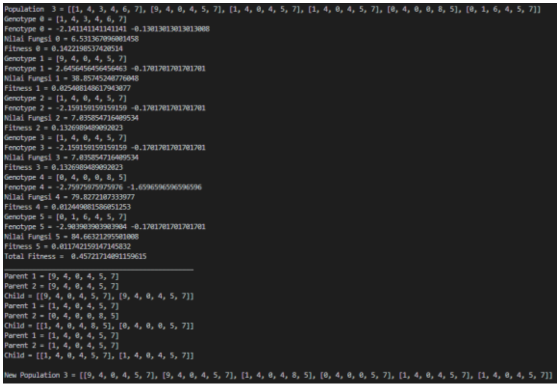
4. Kesimpulan



```
Population  1 = [[6, 3, 8, 0, 3, 2], [5, 4, 9, 4, 4, 8], [0, 1, 6, 0, 8, 1], [3, 6, 4, 4, 6, 9], [1, 4, 0, 4, 5, 7], [7, 7, 4, 9, 7, 1]]
Genotype 0 = [6, 3, 8, 0, 3, 2]
Fenotype 0 = 0.8318318318318325 -1.8718718718718719
Nilai Fungsi 0 = 35.40943748837635
Fitness 0 = 0.02784783249037787
Genotype 1 = [5, 4, 9, 4, 4, 8]
Fenotype 1 = 0.2972972972972978 -0.20620620620620622
Nilai Fungsi 1 = 0.11321122416887544
Fitness 1 = 1.630759452186095
Genotype 2 = [0, 1, 6, 0, 8, 1]
Fenotype 2 = -2.903903903903904 -1.6756756756756757
Nilai Fungsi 2 = 109.45293957255869
Fitness 2 = 0.009094800047070075
Genotype 3 = [3, 6, 4, 4, 6, 9]
Fenotype 3 = -0.8138138138138138 -0.12212212212212203
Nilai Fungsi 3 = 1.8654980471925433
Fitness 3 = 0.42274395499367895
Genotype 4 = [1, 4, 0, 4, 5, 7]
Fenotype 4 = -2.159159159159159 -0.1701701701701701
Nilai Fungsi 4 = 7.035854716409534
Fitness 4 = 0.1326989489092023
Genotype 5 = [7, 7, 4, 9, 7, 1]
Fenotype 5 = 1.6486486486486491 1.8878878878878878
Nilai Fungsi 5 = 41.719101069395236
Fitness 5 = 0.023685961440920004
Total Fitness =  2.2468309500673445
_______________________________________________
Parent 1 = [1, 4, 0, 4, 5, 7]
Parent 2 = [0, 1, 6, 0, 8, 1]
Child = [[1, 1, 6, 0, 8, 1], [0, 4, 0, 4, 5, 7]]
Parent 1 = [1, 4, 0, 4, 5, 7]
Parent 2 = [0, 1, 6, 0, 8, 1]
Child = [[0, 1, 6, 0, 8, 1], [1, 4, 0, 4, 5, 7]]
Parent 1 = [0, 1, 6, 0, 8, 1]
Parent 2 = [1, 4, 0, 4, 5, 7]
Child = [[1, 4, 0, 4, 5, 7], [0, 1, 6, 0, 8, 1]]

New Population 1 = [[1, 1, 6, 0, 8, 1], [0, 4, 0, 4, 5, 7], [0, 1, 6, 0, 8, 1], [1, 4, 0, 4, 5, 7], [1, 4, 0, 4, 5, 7], [0, 1, 6, 0, 8, 1]]
```

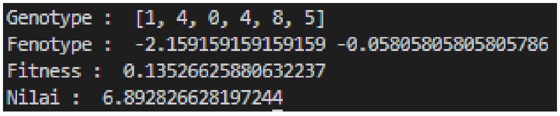
Output dari generasi pertama



```
Population  3 = [[1, 4, 3, 4, 6, 7], [9, 4, 0, 4, 5, 7], [1, 4, 0, 4, 5, 7], [1, 4, 0, 4, 5, 7], [0, 4, 0, 0, 8, 5], [0, 1, 6, 4, 5, 7]]
Genotype 0 = [1, 4, 3, 4, 6, 7]
Fenotype 0 = -2.141141141141141 -0.13013013013013008
Nilai Fungsi 0 = 6.531367096001458
Fitness 0 = 0.1422198537420514
Genotype 1 = [9, 4, 0, 4, 5, 7]
Fenotype 1 = 2.6456456456456463 -0.1701701701701701
Nilai Fungsi 1 = 38.85745240776048
Fitness 1 = 0.025408148617943077
Genotype 2 = [1, 4, 0, 4, 5, 7]
Fenotype 2 = -2.159159159159159 -0.1701701701701701
Nilai Fungsi 2 = 7.035854716409534
Fitness 2 = 0.1326989489092023
Genotype 3 = [1, 4, 0, 4, 5, 7]
Fenotype 3 = -2.159159159159159 -0.1701701701701701
Nilai Fungsi 3 = 7.035854716409534
Fitness 3 = 0.1326989489092023
Genotype 4 = [0, 4, 0, 0, 8, 5]
Fenotype 4 = -2.75975975975976 -1.6596596596596596
Nilai Fungsi 4 = 79.8272107333977
Fitness 4 = 0.012449081586051253
Genotype 5 = [0, 1, 6, 4, 5, 7]
Fenotype 5 = -2.903903903903904 -0.1701701701701701
Nilai Fungsi 5 = 84.66321295501008
Fitness 5 = 0.011742159147145832
Total Fitness =  0.45721714091159615
_______________________________________________
Parent 1 = [9, 4, 0, 4, 5, 7]
Parent 2 = [9, 4, 0, 4, 5, 7]
Child = [[9, 4, 0, 4, 5, 7], [9, 4, 0, 4, 5, 7]]
Parent 1 = [1, 4, 0, 4, 5, 7]
Parent 2 = [0, 4, 0, 0, 8, 5]
Child = [[1, 4, 0, 4, 8, 5], [0, 4, 0, 0, 5, 7]]
Parent 1 = [1, 4, 0, 4, 5, 7]
Parent 2 = [1, 4, 0, 4, 5, 7]
Child = [[1, 4, 0, 4, 5, 7], [1, 4, 0, 4, 5, 7]]

New Population 3 = [[9, 4, 0, 4, 5, 7], [9, 4, 0, 4, 5, 7], [1, 4, 0, 4, 8, 5], [0, 4, 0, 0, 5, 7], [1, 4, 0, 4, 5, 7], [1, 4, 0, 4, 5, 7]]
```

Output dari generasi ketiga



```
Genotype :  [1, 4, 0, 4, 8, 5]
Fenotype :  -2.159159159159159 -0.05805805805805786
Fitness :  0.13526625880632237
Nilai :  6.892826628197244
```

Output dari kromosom terbaik dari 3 generasi

Kesimpulnnya adalah genetic algorithm adalah salah satu metode pencarian nilai minimum dari fungsi matematika, semakin banyak generasi yang dibuat, semakin pasti nilai minimum dari fungsi matematika