Nama: Silitonga, Valentino Rossy

NIM: 2109076038

Mata Kuliah: Kecerdasan Buatan

**LAPORAN TUGAS**

**KECERDASAN BUATAN**

**1. *Listing Program***

import random

# Definisi batasan nilai variabel

min\_a = 0

max\_a = 30

min\_b = 0

max\_b = 10

min\_c = 0

max\_c = 10

min\_d = 0

max\_d = 10

# Definisi jumlah populasi

pop\_size = 6

# Definisi parameter crossover dan mutasi

crossover\_rate = 0.8

mutation\_rate = 0.1

# Fungsi evaluasi (fitness)

def evaluate(chromosome):

a = chromosome[0]

b = chromosome[1]

c = chromosome[2]

d = chromosome[3]

return abs(a + 4\*b + 2\*c + 3\*d -30)

# Inisialisasi populasi secara acak

def initialize\_population():

population = []

for \_ in range(pop\_size):

chromosome = [random.randint(min\_a, max\_a),

random.randint(min\_b, max\_b),

random.randint(min\_c, max\_c),

random.randint(min\_d, max\_d)]

population.append(chromosome)

return population

# Seleksi orang tua berdasarkan probabilitas fitness

def select\_parents(population):

fitness\_sum = sum(1 / (1 + evaluate(chromosome)) for chromosome in population)

probabilities = [(1 / (1 +evaluate(chromosome))) / fitness\_sum for chromosome in population]

cumulative\_probabilities = [sum(probabilities[:i+1]) for i in range(len(probabilities))]

parents = []

for \_ in range(pop\_size):

r = random.random()

for i in range(pop\_size):

if i == 0 and r < cumulative\_probabilities[i]:

parents.append(population[i])

break

elif cumulative\_probabilities[i-1] < r < cumulative\_probabilities[i]:

parents.append(population[i])

break

return parents

# Crossover dengan metode one-cut point

def crossover(parent1, parent2):

if random.random() < crossover\_rate:

cut\_point = random.randint(1, len(parent1) - 1)

child1 = parent1[:cut\_point] + parent2[cut\_point:]

child2 = parent2[:cut\_point] + parent1[cut\_point:]

return child1, child2

else:

return parent1, parent2

# Mutasi dengan mengganti satu gen secara acak

def mutate(chromosome):

if random.random() < mutation\_rate:

gene\_to\_mutate = random.randint(0, len(chromosome) -1)

if gene\_to\_mutate == 0 :

chromosome[gene\_to\_mutate] = random.randint(min\_a, max\_a)

elif gene\_to\_mutate == 1:

chromosome[gene\_to\_mutate] = random.randint(min\_b, max\_b)

elif gene\_to\_mutate == 2:

chromosome[gene\_to\_mutate] = random.randint(min\_c, max\_c)

else:

chromosome[gene\_to\_mutate] = random.randint(min\_d, max\_d)

return chromosome

# Algoritma Genetika

def genetic\_algorithm():

population = initialize\_population()

best\_chromosome = None

best\_fitness = float('inf')

for \_ in range(100): # Jumlah Generasi

parents = select\_parents(population)

next\_generation = []

for i in range(0,len(parents), 2):

parent1 = parents[i]

parent2 = parents[i+1] if i+1 < len(parents) else parents[0] # Wraparound crossover

child1, child2 = crossover(parent1, parent2)

child1 = mutate(child1)

child2 = mutate(child2)

next\_generation.extend([child1, child2])

population = next\_generation

# Evaluasi fitness tiap chromosome dalam populasi

for chromosome in population:

fitness = evaluate(chromosome)

if fitness < best\_fitness:

best\_chromosome = chromosome

best\_fitness = fitness

return best\_chromosome

# Menjalankan algoritma genetika untuk mencari solusi

best\_solution = genetic\_algorithm()

a, b, c, d = best\_solution

print(f"Solusi terbaik: a={a}, b={b}, c={c}, d={d}")

print(f"Hasil persamaan: {a} + 4\*{b} + 2{c} + 3\*{d} = {a + 4\*b + 2\*c + 3\*d}")

**2. Penjelasan atau Algoritma**

1. *Import* *library* adalah *library* yang menghasilkan *function* dengan nilai yang acak
2. Tentukan batasan nilai variabel a, b, c, dan d (a = 0 to 30, b, c, d = 0 to 10)
3. Tentukan jumlah populasi, probabilitas *crossover* (*crossover\_rate*), dan probabilitas mutasi (*mutation\_rate*) dengan nilai yang telah ditentukan
4. Inisialisasi populasi awal yaitu dengan nilai 6
5. Evaluasi chromosome (*function* yang ada pada soal) langkah pertama dalam algoritma genetika
6. Seleksi chromosome langkah kedua berdasarkan ke-*fitness* tiap chromosome
7. Crossover, dengan metode *cut point*
8. Menjalankan algoritma genetika untuk mencari solusi terbaik