

BIS LAB 5/09/2025

Sharada Koundinya (1BM23CS310)

CODE:

```
import random
```

```
import math
```

```
# Example:  $f(x) = x * \sin(10 * \pi * x) + 2$ 
```

```
def fitness_function(x):
```

```
    return x * math.sin(10 * math.pi * x) + 2
```

```
POPULATION_SIZE = 6
```

```
GENE_LENGTH = 10
```

```
MUTATION_RATE = 0.05
```

```
CROSSOVER_RATE = 0.8
```

```
GENERATIONS = 20
```

```
DOMAIN = (-1, 2)
```

```
def random_gene():
```

```
    return random.uniform(DOMAIN[0], DOMAIN[1])
```

```
def create_chromosome():
```

```
    return [random_gene() for _ in range(GENE_LENGTH)]
```

```
def initialize_population(size):
```

```
    return [create_chromosome() for _ in range(size)]
```

```
def evaluate_population(population):
```

```
    return [fitness_function(express_gene(chrom)) for chrom in population]
```

```
def express_gene(chromosome):  
    return sum(chromosome) / len(chromosome)
```

```
def select(population, fitnesses):  
    total_fitness = sum(fitnesses)  
    pick = random.uniform(0, total_fitness)  
    current = 0  
    for individual, fitness in zip(population, fitnesses):  
        current += fitness  
        if current > pick:  
            return individual  
    return random.choice(population)
```

```
def crossover(parent1, parent2):  
    if random.random() < CROSSOVER_RATE:  
        point = random.randint(1, GENE_LENGTH - 1)  
        child1 = parent1[:point] + parent2[point:]  
        child2 = parent2[:point] + parent1[point:]  
        return child1, child2  
    return parent1[:], parent2[:]
```

```
def mutate(chromosome):  
    new_chromosome = []  
    for gene in chromosome:  
        if random.random() < MUTATION_RATE:  
            new_chromosome.append(random_gene())  
    else:
```

```
        new_chromosome.append(gene)
    return new_chromosome
```

```
def gene_expression_algorithm():
    population = initialize_population(POPULATION_SIZE)
    best_solution = None
    best_fitness = float("-inf")

    for generation in range(GENERATIONS):
        fitnesses = evaluate_population(population)

        for i, chrom in enumerate(population):
            if fitnesses[i] > best_fitness:
                best_fitness = fitnesses[i]
                best_solution = chrom[:]

        print(f"Generation {generation+1}: Best Fitness = {best_fitness:.4f}, Best x = {express_gene(best_solution):.4f}")

    new_population = []
    while len(new_population) < POPULATION_SIZE:
        parent1 = select(population, fitnesses)
        parent2 = select(population, fitnesses)
        offspring1, offspring2 = crossover(parent1, parent2)
        offspring1 = mutate(offspring1)
        offspring2 = mutate(offspring2)
        new_population.extend([offspring1, offspring2])
```

```

population = new_population[:POPULATION_SIZE]

print("\nBest solution found:")

print(f"Genes: {best_solution}")

x_value = express_gene(best_solution)

print(f"x = {x_value:.4f}")

print(f"f(x) = {fitness_function(x_value):.4f}")

if __name__ == "__main__":

    gene_expression_algorithm()

```

OUTPUT:

```

Generation 1: Best Fitness = 2.3125, Best x = 0.4262
Generation 2: Best Fitness = 2.3125, Best x = 0.4262
Generation 3: Best Fitness = 2.3125, Best x = 0.4262
Generation 4: Best Fitness = 2.3125, Best x = 0.4262
Generation 5: Best Fitness = 2.3125, Best x = 0.4262
Generation 6: Best Fitness = 2.3125, Best x = 0.4262
Generation 7: Best Fitness = 2.3125, Best x = 0.4262
Generation 8: Best Fitness = 2.4233, Best x = 0.6237
Generation 9: Best Fitness = 2.4233, Best x = 0.6237
Generation 10: Best Fitness = 2.4233, Best x = 0.6237
Generation 11: Best Fitness = 2.4233, Best x = 0.6237
Generation 12: Best Fitness = 2.4233, Best x = 0.6237
Generation 13: Best Fitness = 2.4233, Best x = 0.6237
Generation 14: Best Fitness = 2.4233, Best x = 0.6237
Generation 15: Best Fitness = 2.4233, Best x = 0.6237
Generation 16: Best Fitness = 2.4233, Best x = 0.6237
Generation 17: Best Fitness = 2.4395, Best x = 0.4594
Generation 18: Best Fitness = 2.4395, Best x = 0.4594
Generation 19: Best Fitness = 2.4395, Best x = 0.4594
Generation 20: Best Fitness = 2.4395, Best x = 0.4594

Best solution found:
Genes: [0.6948405045559576, -0.647173288232043, -0.3813499383055478, 1.631627548910124, 0.9271637073163099, 0.0324867196364278, -0.3565755055362756, 1.5226396608397925, 1.0654293190513275]
x = 0.4594
f(x) = 2.4395

```