1BM22CS241

Sanjeet Prajwal Pandit

Genetic_Algorithm

```
import random

def fitness_function(x):
    return x ** 2

def initialize_population(size, lower_bound, upper_bound):
    return [random.uniform(lower_bound, upper_bound) for _ in range(size)]

def roulette_wheel_selection(population):
    total_fitness = sum(fitness_function(x) for x in population)
    selection_probs = [fitness_function(x) / total_fitness for x in
population]
    cumulative_probs = [sum(selection_probs[:i+1]) for i in
range(len(selection_probs))]

    random_value = random.uniform(0, 1)
    for i, cumulative in enumerate(cumulative_probs):
        if random_value <= cumulative:
            return population[i]

def crossover(parent1, parent2):
    return (parent1 + parent2) / 2  # Simple average crossover

def mutate(x, mutation_rate):
    if random.random() < mutation_rate:
        return x + random.uniform(-1, 1)  # Simple mutation
    return x

def genetic_algorithm(population_size, generations, mutation_rate,
lower_bound, upper_bound):
    population = initialize_population(population_size, lower_bound,
upper_bound)

    for generation in range(generations):
        new_population = []

        for _ in range(population_size):
            parent1 = roulette_wheel_selection(population)
            parent2 = roulette_wheel_selection(population)
            child = crossover(parent1, parent2)
            child = mutate(child, mutation_rate)
            new_population.append(child)

        population = new_population

    best_solution = max(population, key=fitness_function)
    return best_solution, fitness_function(best_solution)

# Parameters
population_size = 100
generations = 50
mutation_rate = 0.1
lower_bound = -10
upper_bound = 10
```

```python
best_x, best_fitness = genetic_algorithm(population_size, generations,
mutation_rate, lower_bound, upper_bound)
print(f"Best solution: x = {best_x}, f(x) = {best_fitness}")
```

```
Best solution: x = 9.5644631211605, f(x) = 91.47895479603926
```