

Name: Dharnidhar Thakur

Roll No: 23

Find the maximum solution of $f(x,y,z) = x^2 - xy + z$ with the given criteria $-5 < x < 5$, $0 < y < 2$, $-2 < z < 2$ set following parameter in advance ,

- population size 100
- probability of crossover 0.9
- probability of mutation 0.15
- fitness function is $f(x,y,z)$.

```
import numpy as np

# Function to calculate fitness for an individual (x, y, z)
def fitness(x, y, z):
    return x**2 - x*y + z

# Function to initialize a population of size n
def initialize_population(n, x_min=-5, x_max=5, y_min=0, y_max=2, z_min=-2, z_max=2):
    population = []
    for _ in range(n):
        x = np.random.uniform(x_min, x_max)
        y = np.random.uniform(y_min, y_max)
        z = np.random.uniform(z_min, z_max)
        population.append((x, y, z))
    return population

# Function to perform selection using tournament selection
def tournament_selection(population, fitness_values, tournament_size=3):
    selected_indices = []
    for _ in range(len(population)):
        tournament = np.random.choice(len(population), tournament_size, replace=False)
        tournament_fitness = [fitness_values[i] for i in tournament]
        selected_indices.append(tournament[tournament_fitness.index(max(tournament_fitness))])
    return selected_indices

# Function to perform crossover (single-point crossover) between 2 parent
def crossover(parent1, parent2):
    crossover_point = np.random.randint(1, len(parent1))
    child1 = parent1[:crossover_point] + parent2[crossover_point:]
    child2 = parent2[:crossover_point] + parent1[crossover_point:]
    return child1, child2

# Function to perform mutation (random perturbation)
def mutate(individual, mutation_prob=0.15, x_min=-5, x_max=5, y_min=0, y_max=2, z_min=-2, z_max=2):
    mutated_individual = list(individual)
    for i in range(3):
        if np.random.rand() < mutation_prob:
            if i == 0:
                mutated_individual[i] = np.random.uniform(x_min, x_max)
            elif i == 1:
                mutated_individual[i] = np.random.uniform(y_min, y_max)
            elif i == 2:
                mutated_individual[i] = np.random.uniform(z_min, z_max)
    return tuple(mutated_individual)

# Genetic Algorithm
def genetic_algorithm(population_size=100, num_generations=100, crossover_prob=0.9, mutation_prob=0.15):
    # Initialize the population
    population = initialize_population(population_size)

    for generation in range(num_generations):
        # Calculate fitness for each individual
        fitness_values = [fitness(*individual) for individual in population]

        # Selection
        selected_indices = tournament_selection(population, fitness_values)

        # to perform Crossover
        new_population = []
        for i in range(0, population_size, 2):
            if np.random.rand() < crossover_prob:
                child1, child2 = crossover(population[selected_indices[i]], population[selected_indices[i + 1]])
                new_population.extend([child1, child2])
            else:
                new_population.extend([population[selected_indices[i]], population[selected_indices[i + 1]]])

        #for Mutation
        new_population = [mutate(ind) for ind in new_population]
```

```
# Replace the old population with the new population
population = new_population

# Calculate fitness for the final population
fitness_values = [fitness(*individual) for individual in population]

# Find the maximum fitness and corresponding individual
max_fitness = max(fitness_values)
max_individual = population[fitness_values.index(max_fitness)]

return max_individual, max_fitness

# Run the genetic algorithm
best_individual, best_fitness = genetic_algorithm()
print("Best Individual:", best_individual)
print("Best Fitness:", best_fitness)
```

```
➤ Best Individual: (-4.99130863261848, 1.9966524724848473, 1.999378034936691)
Best Fitness: 36.878448623241106
```

[Colab paid products](#) - [Cancel contracts here](#)

✓ Connected to Python 3 Google Compute Engine backend

