```
import numpy as np

# Define the optimization problem
def fitness_function(x):
    # Example: minimize x^2
    return np.sum(x**2)

# Initialize parameters
N = 50  # Number of wolves
T = 100  # Number of iterations
dim = 2  # Number of dimensions in the search space
wolves = np.random.rand(N, dim)  # Random initial positions of wolves

# Initialize alpha, beta, delta wolves
alpha_position = np.zeros(dim)
beta_position = np.zeros(dim)
delta_position = np.zeros(dim)

alpha_fitness = float("inf")
beta_fitness = float("inf")
delta_fitness = float("inf")

# Start optimization
for t in range(T):
    for i in range(N):
        # Evaluate fitness of each wolf
        fitness_value = fitness_function(wolves[i])

        # Update alpha, beta, delta wolves based on fitness values
        if fitness_value < alpha_fitness:
            delta_fitness = beta_fitness
            delta_position = beta_position
            beta_fitness = alpha_fitness
            beta_position = alpha_position
            alpha_fitness = fitness_value
            alpha_position = wolves[i]

        elif fitness_value < beta_fitness:
            delta_fitness = beta_fitness
            delta_position = beta_position
            beta_fitness = fitness_value
            beta_position = wolves[i]

        elif fitness_value < delta_fitness:
            delta_fitness = fitness_value
            delta_position = wolves[i]

    # Update the positions of wolves
    A = 2 * np.random.rand(N, dim) - 1
    C = 2 * np.random.rand(N, dim)

    D_alpha = np.abs(C * alpha_position - wolves)
    D_beta = np.abs(C * beta_position - wolves)
    D_delta = np.abs(C * delta_position - wolves)

    wolves = wolves + A * (D_alpha + D_beta + D_delta) / 3  # Update position based on all 3 wolves

# Output the best solution found
print("Best Position:", alpha_position)
print("Best Fitness:", alpha_fitness)
```

```
Best Position: [-1.73665799e-11  7.69039851e-11]
Best Fitness: 6.215821017318649e-21
```