import numpy as np

def gwo(obj\_function, dim, search\_agents, max\_iter, lb, ub):

# Initialize alpha, beta, and delta positions

Alpha\_pos = np.zeros(dim)

Beta\_pos = np.zeros(dim)

Delta\_pos = np.zeros(dim)

Alpha\_score = float("inf")

Beta\_score = float("inf")

Delta\_score = float("inf")

# Initialize the positions of search agents

positions = np.random.uniform(lb, ub, (search\_agents, dim))

for iteration in range(max\_iter):

for i in range(search\_agents):

# Constrain positions within search space

positions[i] = np.clip(positions[i], lb, ub)

# Evaluate the fitness of each agent

fitness = obj\_function(positions[i])

# Update Alpha, Beta, and Delta

if fitness < Alpha\_score:

Alpha\_score, Alpha\_pos = fitness, positions[i].copy()

elif fitness < Beta\_score:

Beta\_score, Beta\_pos = fitness, positions[i].copy()

elif fitness < Delta\_score:

Delta\_score, Delta\_pos = fitness, positions[i].copy()

# Print the current best score at each iteration

print(f"Iteration {iteration + 1}/{max\_iter}, Best Score: {Alpha\_score:.6f}")

# Update the position of each search agent

a = 2 - iteration \* (2 / max\_iter) # Linearly decreases from 2 to 0

for i in range(search\_agents):

for j in range(dim):

r1, r2 = np.random.rand(), np.random.rand()

A1, C1 = 2 \* a \* r1 - a, 2 \* r2

D\_alpha = abs(C1 \* Alpha\_pos[j] - positions[i, j])

X1 = Alpha\_pos[j] - A1 \* D\_alpha

r1, r2 = np.random.rand(), np.random.rand()

A2, C2 = 2 \* a \* r1 - a, 2 \* r2

D\_beta = abs(C2 \* Beta\_pos[j] - positions[i, j])

X2 = Beta\_pos[j] - A2 \* D\_beta

r1, r2 = np.random.rand(), np.random.rand()

A3, C3 = 2 \* a \* r1 - a, 2 \* r2

D\_delta = abs(C3 \* Delta\_pos[j] - positions[i, j])

X3 = Delta\_pos[j] - A3 \* D\_delta

positions[i, j] = (X1 + X2 + X3) / 3 # Average of Alpha, Beta, Delta

return Alpha\_pos, Alpha\_score

# Example: Optimization of the Sphere function

def sphere\_function(x):

return np.sum(x\*\*2)

# Parameters

dim = 5 # Dimensionality

search\_agents = 30 # Number of wolves

max\_iter = 50 # Maximum iterations

lb, ub = -10, 10 # Search space boundaries

best\_position, best\_score = gwo(sphere\_function, dim, search\_agents, max\_iter, lb, ub)

print("Best Position:", best\_position)

print("Best Score:", best\_score)