

```

import numpy as np
def cuckoo_search(func, bounds, n_nests=25, n_iterations=10, pa=0.25):
    dim = len(bounds)

    nests = initialize_nests(bounds, n_nests)

    fitness = evaluate_fitness(nests, func)

    best_idx = np.argmin(fitness)
    best_solution = nests[best_idx]
    best_fitness = fitness[best_idx]

    for iteration in range(n_iterations):
        new_nests = generate_new_solutions(nests)
        new_fitness = evaluate_fitness(new_nests, func)

        for i in range(n_nests):
            if np.random.rand() < pa:
                new_nests[i] = generate_random_solution(bounds)

        for i in range(n_nests):
            if new_fitness[i] < fitness[i]:
                nests[i] = new_nests[i]
                fitness[i] = new_fitness[i]
        best_idx = np.argmin(fitness)
        if fitness[best_idx] < best_fitness:
            best_solution = nests[best_idx]
            best_fitness = fitness[best_idx]

        print(f"Iteration {iteration + 1}/{n_iterations}, Best Fitness: {best_fitness}")

    return best_solution, best_fitness

def initialize_nests(bounds, n_nests):
    nests = np.random.rand(n_nests, len(bounds))
    for i in range(len(bounds)):
        nests[:, i] = bounds[i][0] + (bounds[i][1] - bounds[i][0]) * nests[:, i]
    return nests

def evaluate_fitness(nests, func):
    return np.array([func(nest) for nest in nests])

def generate_new_solutions(nests):
    new_nests = np.copy(nests)
    levy_flight = np.random.normal(0, 1, size=nests.shape) * np.abs(np.random.normal(0, 1, size=nests.shape))
    new_nests += levy_flight
    return new_nests

def generate_random_solution(bounds):
    return np.array([np.random.uniform(bounds[i][0], bounds[i][1]) for i in range(len(bounds))])

```

```
def sphere_function(x):  
    return np.sum(x**2)  
  
bounds = [(-5.0, 5.0), (-5.0, 5.0)]  
  
best_solution, best_fitness = cuckoo_search(sphere_function, bounds)  
  
print("Best Solution:", best_solution)  
print("Best Fitness:", best_fitness)
```



```
Iteration 1/10, Best Fitness: 2.1519347281478205  
Iteration 2/10, Best Fitness: 0.173944468763009  
Iteration 3/10, Best Fitness: 0.173944468763009  
Iteration 4/10, Best Fitness: 0.173944468763009  
Iteration 5/10, Best Fitness: 0.173944468763009  
Iteration 6/10, Best Fitness: 0.173944468763009  
Iteration 7/10, Best Fitness: 0.173944468763009  
Iteration 8/10, Best Fitness: 0.173944468763009  
Iteration 9/10, Best Fitness: 0.173944468763009  
Iteration 10/10, Best Fitness: 0.173944468763009  
Best Solution: [-0.05758749  0.4130716 ]  
Best Fitness: 0.173944468763009
```