**Homework3**

|  |  |  |
| --- | --- | --- |
| 202283890036 | 黄家睿(Jerry Huang) | Internet of Things |

The report analyzes and summarizes a Python implementation code based on the Beluga Whale Optimization algorithm (abbreviated as BWO). The goal is to find the optimal solution for a given objective function by optimizing the algorithm. The code implements two class structures: BWO\_CLASS and BWO\_solution, which inherits from it, containing the initialization and execution logic of the algorithm.

## The Class of the evolutionary algorithm

### class BWO\_CLASS

BWO\_CLASS is the base class of the algorithm, mainly responsible for initializing parameters and providing simple printing functions for target function information.

Parameters:

1. population\_size: Population size, default is 300.
2. max\_iterations: Maximum number of iterations, default is 500.
3. objective\_function: The target function identifier defaults to "F9".

Function：

1. \_\_init\_\_：Complete parameter initialization.
2. BWO\_function：

Retrieve information about the objective function from the benchmarks module, including the bounds of the search space (lb, ub), the dimension (dim), and the specific handle of the objective function (fobj). Also, print out information about the objective function.

The code:

class BWO\_CLASS:

def \_\_init\_\_(self, population\_size=300, max\_iterations=500, objective\_function="F9"):

*# 初始化 BWO 类。*

*#*

*# 参数：*

*# population\_size (int): 种群数量，默认为 300。*

*# max\_iterations (int): 最大迭代次数，默认为 500。*

*# objective\_function (callable): 目标函数，默认为 F9函数。*

self.population\_size = population\_size

self.max\_iterations = max\_iterations

self.objective\_function = objective\_function

def BWO\_function(self):

func\_details1 = benchmarks.getFunctionSet(self.objective\_function)

lb = func\_details1[1]

ub = func\_details1[2]

dim = func\_details1[3]

fobj = getattr(benchmarks, self.objective\_function) *# 获取函数求解*

print(str(self.objective\_function) + " " +

str(self.max\_iterations) + " " +

str(self.population\_size) + " " +

str(ub) + " " +

str(lb) + " " +

str(dim) + " " +

str(fobj))

### 1.2 Class BWO\_solution

The BWO\_solution class inherits from BWO\_CLASS and extends the practical solving logic of the optimization algorithm on the basis of the parent class's functionality.

This class is responsible for implementing the main functions of the BWO algorithm, and on the basis of inheriting from the parent class, it realizes the calculation of the optimal solution.

**BWO\_function：**

1. Get the target function information from the benchmarks module (consistent with the parent class logic).
2. Invoke the BWO method in the BWO module, passing in the target function and related parameters, to perform the optimization calculation.

The code:

class BWO\_solution(BWO\_CLASS):

def BWO\_function(self):

func\_details1 = benchmarks.getFunctionSet(self.objective\_function)

lb = func\_details1[1]

ub = func\_details1[2]

dim = func\_details1[3]

fobj = getattr(benchmarks, self.objective\_function) *# 获取函数求解*

x = BWO.BWO(fobj, lb, ub, dim, self.population\_size, self.max\_iterations)

## 2.The primary members

### 2.1 Definition Primary member

In the \_\_init\_\_ constructor method of BWO\_CLASS, population\_size, max\_iterations, and objective\_function are defined as class attributes and assigned based on the initialization parameters.

class BWO\_CLASS:

def \_\_init\_\_(self, population\_size=300, max\_iterations=500, objective\_function="F9"):

*# 初始化主要成员*

self.population\_size = population\_size *# 种群数量*

self.max\_iterations = max\_iterations *# 最大迭代次数*

self.objective\_function = objective\_function *# 目标函数名称*

### 2.2 The use of Primary Member

def BWO\_function(self):

func\_details1 = benchmarks.getFunctionSet(self.objective\_function) *# 使用目标函数*

lb = func\_details1[1] *# 搜索下界*

ub = func\_details1[2] *# 搜索上界*

dim = func\_details1[3] *# 问题维度*

fobj = getattr(benchmarks, self.objective\_function) *# 获取目标函数句柄*

print(str(self.objective\_function) + " " +

str(self.max\_iterations) + " " +

str(self.population\_size) + " " +

str(ub) + " " +

str(lb) + " " +

str(dim) + " " +

str(fobj))

Here, self.objective\_function and other member variables are referenced multiple times, ensuring that the method logic remains consistent with the main members of the class.

## 3.inheritance

The BWO\_solution class inherits from the BWO\_CLASS, therefore the primary members defined in BWO\_CLASS are also inherited. The subclass methods can directly use the properties of the parent class:

class BWO\_solution(BWO\_CLASS):

def BWO\_function(self):

*# 直接调用父类的 primary members*

func\_details1 = benchmarks.getFunctionSet(self.objective\_function)

lb = func\_details1[1]

ub = func\_details1[2]

dim = func\_details1[3]

fobj = getattr(benchmarks, self.objective\_function)

*# 使用 primary members 调用核心算法*

x = BWO.BWO(fobj, lb, ub, dim, self.population\_size, self.max\_iterations)

## 4.polymorphism

In my code, polymorphism is achieved through method overriding. The subclass BWO\_solution inherits from the parent class BWO\_CLASS and redefines the BWO\_function method of the parent class, implementing different behaviors, thus achieving polymorphism.

### 4.1 The function in father class

The parent class BWO\_CLASS defines a generic BWO\_function method, whose main function is to print information related to the target function. This method is generic and does not execute optimization algorithms; it is only used for information display.

### 4.1 The reload of the child class

The subclass BWO\_solution inherits from BWO\_CLASS and overrides the BWO\_function method, extending its functionality by directly calling the optimization algorithm to perform the actual computation.

### 4.3 Implementation in the main method

if \_\_name\_\_ == '\_\_main\_\_':

bwo\_test = BWO\_CLASS() *# 创建父类对象*

bwo\_test.BWO\_function() *# 调用父类方法*

bwo = BWO\_solution() *# 创建子类对象*

bwo.BWO\_function() *# 调用子类方法*

***bwo\_test.BWO\_function():***

The method of the parent class is called, outputting the information of the target function, but not executing the optimization algorithm.

***bwo.BWO\_function():***  
The method of the subclass is called, executing the actual optimization algorithm.