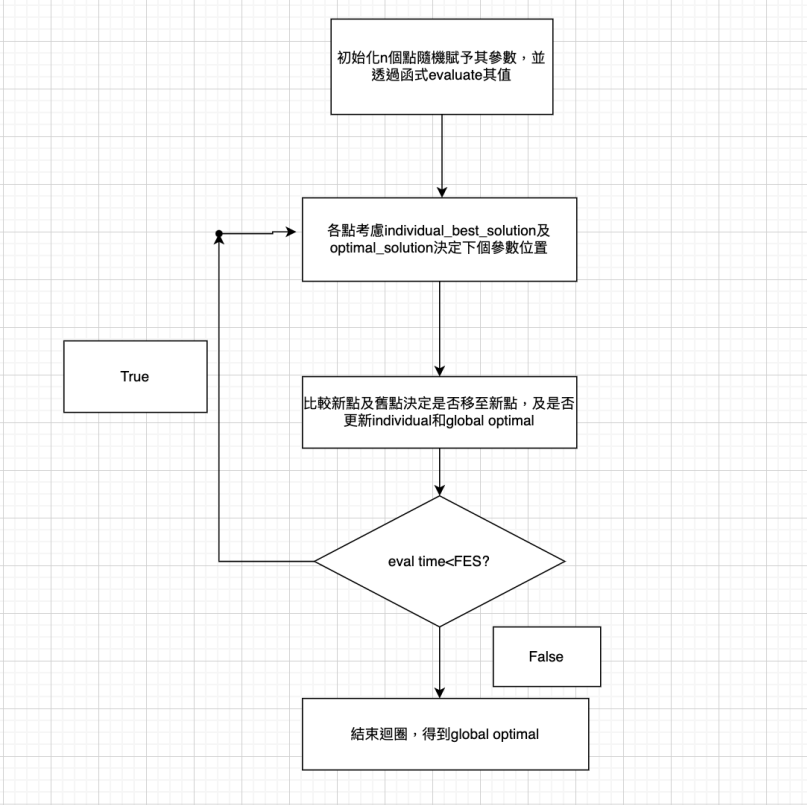
Hw5 code review report

111062503 吳冠志

1. 程式流程
   - 本次實作演算法：Particle Swarm



2. 程式

```
def __init__(self, target_func):
    super().__init__(target_func) # must have this init to work normally

    self.lower = self.f.lower(target_func)
    self.upper = self.f.upper(target_func)
    self.dim = self.f.dimension(target_func)

    self.target_func = target_func

    self.eval_times = 0
    self.optimal_value = float("inf") # 正無限大實體物件
    self.optimal_solution = np.empty(self.dim)

    # Particle Swarm Optimization實作
    self.pop_size=10
    self.solutions=[]
    self.individual_best_solution=[]
    self.individual_objective_value=[]

    # C1、C2、W
    self.cognition_factor=3
    self.social_factor=6
    self.reach=0
```

參數介紹

Self.lower、self.upper：函式參數的上下界

Self.dim：參數個數

Self.eval_times：呼叫 self.f.evaluate 次數

Self.optimal_value：此演算法找到著 minimum

Self.optimal_solution：minimum 對應的參數

Self.pop_size：使用幾個點來找 minimum

Self.solution：用於點參數的更新，看是否能超越 individual_best_solution

Self.best_solution：此點目前找到的最佳解對應的參數

Self.individual_objective_value：此點目前找到的最佳解

Self.cognition_factor：決定往 individual_best_solution 更新的比例

Self.social_factor：決定往 global_optimal 更新的比例

Self.reach：當呼叫 func 次數超過限制時 reach 設 1、幫助離開迴圈

```
35      def initialize(self):
36          min_index=0
37          min_val=float("inf")
38          for i in range(self.pop_size):
39              solution=np.random.uniform(np.full(self.dim, self.lower), np.full(self.dim, self.upper), self.dim)
40              self.solutions.append(solution)
41              self.individual_best_solution.append(solution)
42              objective=self.f.evaluate(func_num, solution)
43              self.eval_times += 1
44              self.individual_objective_value.append(objective)
45
46              if objective<min_val:
47                  min_index=i
48                  min_val=objective
49          self.optimal_solution=self.solutions[min_index].copy()
50          self.optimal_value =min_val
```

隨機設定參數初始化 solutions

設定 individual_best_solution、optimal_solution 以利日後更新

```
51      def move(self):
52          for i,solution in enumerate(self.solutions):
53              C1=self.cognition_factor*random.random()
54              C2=self.social_factor*random.random()
55              for d in range(self.dim):
56                  v=C1*(self.individual_best_solution[i][d]-self.solutions[i][d])+C2*(self.optimal_solution[d]-self.solutions[i][d])
57                  self.solutions[i][d]=0.7*self.solutions[i][d]+v
58                  self.solutions[i][d]=min(self.solutions[i][d],self.upper)
59                  self.solutions[i][d]=max(self.solutions[i][d],self.lower)
```

以 self.cognition_faceor 及 self.social_faceor 決定往 individual_best_solutiom 和 optimal_solution 前進的比例

```
60      def update_solution(self):
61          for i,solution in enumerate(self.solutions):
62              obj_val=self.f.evaluate(func_num,solution)
63              self.eval_times += 1
64              if obj_val == "ReachFunctionLimit":
65                  print("ReachFunctionLimit")
66                  self.reach=1
67                  break
68              if(obj_val<self.individual_objective_value[i]):
69                  self.individual_best_solution[i]=solution
70                  self.individual_objective_value[i]=obj_val
71                  if obj_val<self.optimal_value :
72                      self.optimal_solution=solution
73                      self.optimal_value =obj_val
```

決定各點之 solution 是否成為 individual 和 optimal solution

3. 成果

PSO.py_function1.txt

```
2.4156914633702493e-38
-4.433430347002611e-26
1.2218805573933632e-25
1.3849967003584417e-25
4.379606148715857e-25
-5.811587150068876e-25
6.706969486981888e-76
```

My result<CMAES result

PSO.py_function2.txt

```
-8.747043529205493e-23
-7.550674345278281e-23
1.976358929641572
```

My result>RS result

PSO.py_function3.txt

```
6.192030514476615e-39
3.203688563405289e-39
-2.5415631333517775e-39
-6.288529822403999e-39
1.812021669945228e-38
4.440892098500626e-16
```

My result<CMAES result

PSO.py_function4.txt

```
3.7514901146151025e-37
2.57252310222069e-37
5.080121201256584e-38
1.1638016178398928e-37
1.4073018916066517e-37
-5.798342960829138e-39
1.4933135122169916e-37
4.201772773469827e-37
-1.86710050891935e-37
3.744051743133646e-37
1.1685282821546326
```

CMAES result <My result<RS result

4. 結論

本次嘗試 Particle Swarm 演算法，可以看到成效並不是這麼好，func 2 的成果比 RS 還差，表示其很容易落在 local minimum，接著就卡住無法繼續更新直到呼叫 function 次數達到限制，呼叫次數有限也是其中一個無法達到很好效果的其中一個原因，使用 surrogate model 是一個可考慮的方向。