PSO 演算法

613k0007c 余品誼

報告使用 PSO 優化問題，需包含實驗結果、分析及討論，需包含所使用的參數、準確性、實驗結果的一致性(和 GA 比較)、可獲致滿意結果的維度（D=10、D=30）等。

目標：

1. 優化單峰問題(Shere function)
2. 優化雙峰問題(Schwefel function)

名詞解釋：

1. PSO

   PSO 是指 Particle Swarm Optimization (粒子群最佳化) 演算法。這是一種基於群體智慧的最佳化方法，模擬了鳥群覓食或魚群游動等自然現象的行為。它透過群體內的粒子互相合作與競爭，來尋找全局最佳解。

   運作原理：

   粒子 (Particle)：每個粒子代表一個候選解，粒子具有位置和速度。在每次迭代中，粒子的位置會根據當前位置、速度以及它個體的最佳經驗和群體的最佳經驗進行更新。

   個體最佳位置 (Personal Best)：每個粒子會記錄它自身所探索過的最佳位置。

   全局最佳位置 (Global Best)：粒子群中所有粒子中找到的最佳位置，會影響所有粒子的位置更新。

PSO 實驗方法：

1. Shere function

   同下

2. Schwefel function

   - 初始化

     ```
     pso = PSO(population_size=100, dimensions=30, max_iter=200000)
     ```

   - 迭代尋找最佳

- 更新個體和全局最佳位置

```
for particle in self.particles:
    # 更新個體的最佳位置
    particle.update_personal_best()

    # 更新全局最佳位置
    if particle.best_fitness < self.global_best_fitness:
        self.global_best_fitness = particle.best_fitness
        self.global_best_position = particle.best_position.copy()
```

- 更新速度和位置

```
# 更新速度和位置
for particle in self.particles:
    inertia = self.w * particle.velocity
    cognitive = self.c1 * np.random.rand(self.dimensions) * (particle.best_position - particle.position)
    social = self.c2 * np.random.rand(self.dimensions) * (self.global_best_position - particle.position)

    particle.velocity = inertia + cognitive + social
    particle.position += particle.velocity

    # 限制粒子位置範圍在 [-500, 500] 之間
    particle.position = np.clip(particle.position, -500, 500)
```

慣性部分 (inertia)：保持粒子之前的速度方向，這有助於粒子在沒有顯著吸引力的方向上繼續保持動量。

認知部分 (cognitive)：這部分讓粒子傾向於靠近它自身的最佳位置。

社會部分 (social)：這部分讓粒子傾向於靠近全局最佳位置。

particle.velocity：這一行綜合了慣性、個體影響和全局影響，更新了粒子的速度。

particle.position += particle.velocity：更新粒子的位置。

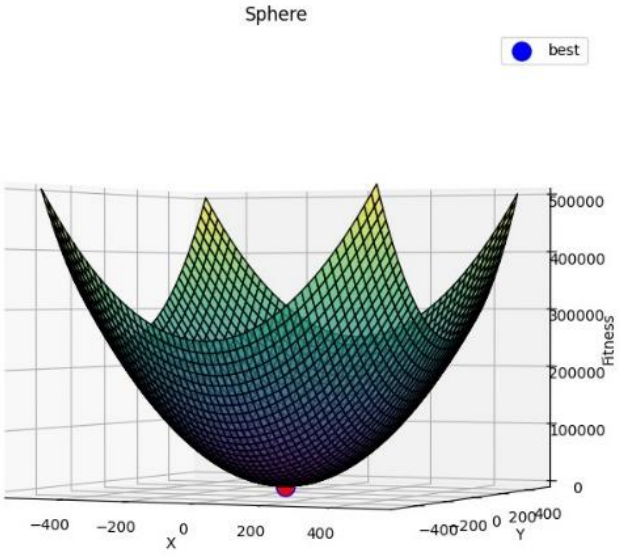
- 記錄全局最佳適應度

實驗結果：

1. Shere function

[-0.00, 0.00, 28.41, 500.00, 67.45,
  449.34, 107.12, -9.63, -146.27, 31.85,
  -7.00, -51.85, 26.09, -271.91, -316.90,
 -425.92, 238.08, 84.15, 500.00, 87.39,
 -88.42, 148.42, 19.56, -474.43, -500.00,
  162.85, -484.52, 273.05, -5.75, 220.10]

全局最佳適應度：3.07e-18 = 0.00000000000000000307

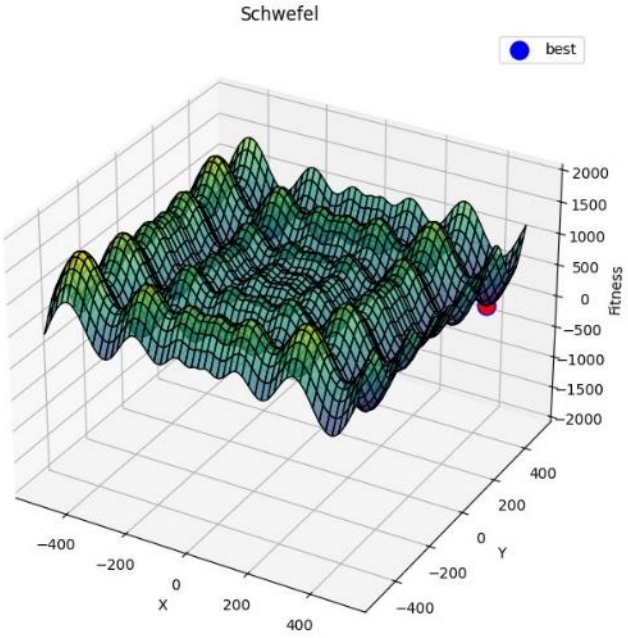


2. Schwefel function

最終最佳位置：

[420.97, 420.97, -6.73, -299.78, -326.21,
  -73.33, 437.56, -401.20, -316.77, -288.13,
   67.81, 418.19, -218.50, 28.16, 116.47,
 -249.59, -271.70, -434.03, -54.05, -319.08,
  143.56, 453.51, -485.83, 338.87, -196.40,
 -387.44, 164.97, -496.00, 85.80, -21.95]

全局最佳適應度：2.5455132458773733e-05 =
0.000025455132458773733

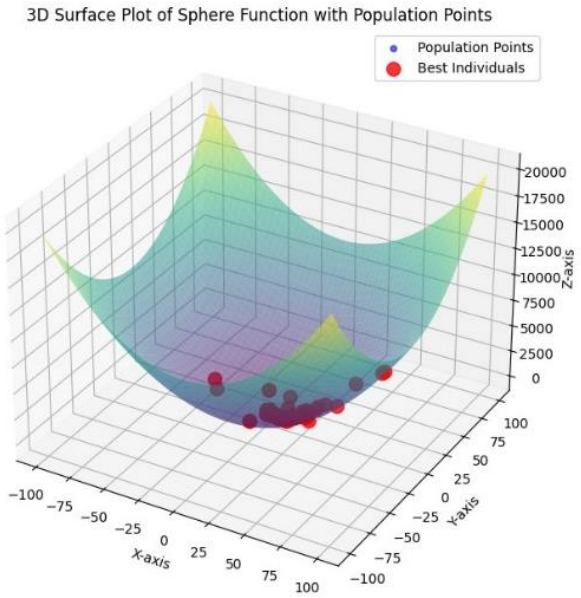


GA 實驗結果：

實驗一：

1. Shere function

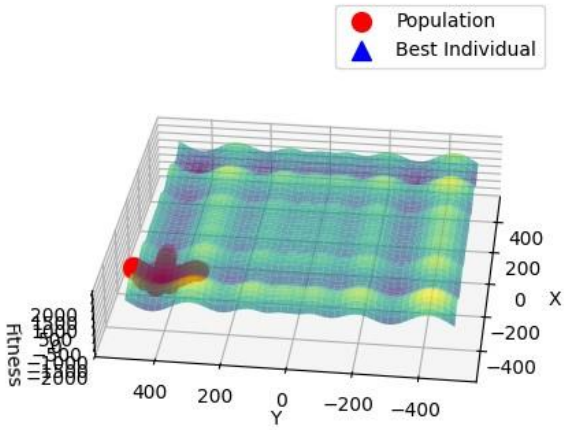
   最優適應度: 0.00037732148285163884119178257O



1. Schwefel function

全局最佳適應度: 2467.9004302955036

[-302.52, 420.59, 420.97, -302.51, 204.34, 717.06,
203.80, -302.52, 420.96, -560.64, -302.52, 420.96,
-302.53, -302.84, 420.97, 420.78, -302.52, 420.97,
203.81, 203.71, 203.81, 203.82, 420.98, -302.49,
203.81, 203.81, 204.44, 420.96, 420.97, 420.97]



實驗結果比較：

單峰問題（Sphere function）：

- PSO：在 Sphere function 中，PSO 演算法表現非常好，最終找到了接近全局最小值的位置，且收斂速度較快。
- GA：GA 在 Sphere function 中也能找到較優解，但相比 PSO，其收斂速度較慢，適應度值稍高。

多峰問題（Schwefel function）：

- PSO：在 Schwefel function 中，PSO 演算法能夠找到較接近全局最優的解，但 Schwefel 函數的多峰結構讓 PSO 容易陷入局部最優解。
- GA：GA 在 Schwefel function 中表現較好，通過基因操作能夠跳出局部最優解，最終找到了更低的適應度值。

總結：

PSO 的優勢：

- PSO 對於連續的、光滑的適應度表面（如 Sphere function）具有-較好的收斂性，特別是在高維優化問題中，PSO 的性能優於 GA。
- PSO 的運算速度較快，因為沒有基因操作，粒子直接根據速度和位置更新。

GA 的優勢：

- GA 能夠處理具有複雜適應度表面（如多峰問題）的優化問題。由於 GA 的基因操作具有隨機性，能夠跳出局部最優解，更適合處理具有多個局部極值的問題。
- GA 更靈活，能夠處理離散、組合問題，適用範圍廣泛。