

Metaheuristic Algorithms and Applications: Program Report III

Group Members: 葉宣佑 王力恩 張襄翊
2025/6/2

目錄

第 1 章 研究動機與目的	2
第 2 章 題目介紹	2
第 3 章 演算法介紹	2
3.1 基因演算法 (Genetic Algorithm, GA)	2
3.2 粒子群優化演算法(Particle Swarm Optimization, PSO).....	3
3.3 蟻群演算法(Ant Colony Optimization, ACO)	3
第 4 章 實驗參數	4
4.1 PART I 參數設定	4
4.2 PART II 參數設定	4
第 5 章 結果與分析	5
PART I.....	5
Experimental Results	5
Analysis	12
PART II	14
Experimental Results	14
Analysis	16
Table of The Part 1 statistic (12 Cities).....	17
Table of The Part 1 statistic (36 Cities).....	17
Table of The Part 1 statistic (52 Cities).....	17
Table of The Part 1 statistic (12 Cities).....	18
第 5 章 結論	18
第 6 章 實驗結果	19

第 1 章 研究動機與目的

本研究旨在探討與比較兩種常見元啟發式演算法：蟻群演算法（ACO）、粒子群優化演算法（PSO）與基因演算法（GA），於旅行商問題（TSP）上的表現。透過測試，分析各演算法在不同城市數量（12、36、52 個城市）下的最佳距離、平均距離、計算時間與適應函數表現，並進一步探討在特定限制條件（奇數城市不可連接）下的解決能力，期望能對不同問題特性選擇適當演算法提供依據。。

第 2 章 題目介紹

旅行推銷員問題（Traveling Salesman Problem, TSP）是一個經典的組合優化問題，其目標是找到一條最短的路徑，該路徑需經過一組城市並返回起始城市。TSP 在物流、交通規劃及路徑優化等領域具有廣泛應用。

第 3 章 演算法介紹

3.1 基因演算法 (Genetic Algorithm, GA)

基因演算法（Genetic Algorithm, GA）是一種模擬自然選擇與遺傳作用的群體式進化演算法。在本實作中，個體以實數向量表示，整體流程如下：

- 【初始化】：隨機產生 N 個個體，並使其落於定義域範圍內。
- 【適應度評估】：根據目標函數評估每個個體的適應度。
- 【選擇】：使用選擇較優的個體具有更高的機率被選中參與交配。
- 【交配】：採用多點交配策略，將父母染色體的不同區段進行交換，分段數與個體維度成比例，在離散題目 TSP 中我們使用順序交叉法（Order Crossover, OX），確保基因的穩定性。
- 【變異】：每個基因以固定機率進行變異，變異方式為將隨機路徑中的兩個位置進行交換。

3.2 粒子群優化演算法(Particle Swarm Optimization, PSO)

粒子群優化演算法（PSO）模擬群體中的社會行為，藉由結合個體與群體的最佳經驗，引導粒子進行搜尋與移動。其運作流程如下：

- 【初始化】：隨機初始化粒子 N 個位置與速度，由於 TSP 為離散行問題，因此我們將速度表示法改成一組交換操作（List of swap action）。
- 【速度更新】：每個粒子的操作會根據其個人歷史最佳位置與整體群體的最佳位置進行調整，新的操作組會根據原始操作組與他人操作組影響。
- 【位置更新】：粒子根據更新後的操作移動至新的位置，持續探索解空間。
- 【最佳解更新】：每次迭代中，更新個體的歷史最佳位置與群體目前的全域最佳解。

3.3 蟻群演算法(Ant Colony Optimization, ACO)

蟻群演算法（Ant Colony Optimization, ACO）模擬螞蟻群體尋找食物的行為，透過信息素的積累與揮發，引導螞蟻找到最短路徑。以下是其運作流程：

- 【初始化】：隨機初始化螞蟻的位置，並初始化信息素矩陣，所有邊的初始信息素值相等。
- 【路徑構建】：每隻螞蟻從起始城市出發，根據信息素濃度與距離計算選擇下一城市的機率。重複此過程，直到每隻螞蟻完成一條完整的路徑。
- 【適應值計算】：計算每條路徑的總距離，並根據距離計算適應值（通常為距離的倒數）。
- 【信息素更新】：
 - 信息素揮發：所有邊的現有信息素值按比例減少。
 - 信息素增強：根據螞蟻的路徑與適應值，增加信息素濃度，短路徑會獲得更多信息素。
- 【最佳解更新】：比較所有螞蟻的路徑，更新當前的最佳路徑與距離。
- 【迭代】：重複路徑構建、信息素更新與最佳解更新，直到達到最大迭代次數或收斂條件。

第 4 章 實驗參數

4.1 PART I 參數設定

城市數量：[12, 36, 52]

實驗次數：10 次

演算法	編號	其餘參數設定
GA	1	population_size=50, generations=1000, mutation_rate=0.2
GA	2	population_size=100, generations=500, mutation_rate=0.1
PSO	1	population_size=10000, generations=100, inertia_weight=1, cognitive_weight=2, social_weight=3
PSO	2	population_size=5000, generations=200, inertia_weight=0.5, cognitive_weight=1.5, social_weight=2
ACO	1	num_ants=10, num_iterations=200, alpha=1, beta=2, evaporation_rate=0.3
ACO	2	num_ants=20, num_iterations=100, alpha=1, beta=2, evaporation_rate=0.3
ACO	3	num_ants=10, num_iterations=200, alpha=1, beta=2, evaporation_rate=0.5

4.2 PART II 參數設定

城市數量：36（奇數編號城市之間無連接）

實驗次數：10 次

使用特製的距離演算，當有無法連接的城市連接時不直接將無限大的距離作為修正函式，而是增加距離處罰在越多違反規定的邊上。

演算法	編號	其餘參數設定
GA	1	population_size=50, generations=1500, mutation_rate=0.15
GA	2	population_size=100, generations=1000, mutation_rate=0.1
PSO	1	population_size=10000, generations=100, inertia_weight=1, cognitive_weight=2, social_weight=3
PSO	2	population_size=5000, generations=200, inertia_weight=0.5, cognitive_weight=1.5, social_weight=2
ACO	1	num_ants=10, num_iterations=300, alpha=1, beta=2.5, evaporation_rate=0.4
ACO	2	num_ants=25, num_iterations=200, alpha=1, beta=3, evaporation_rate=0.6

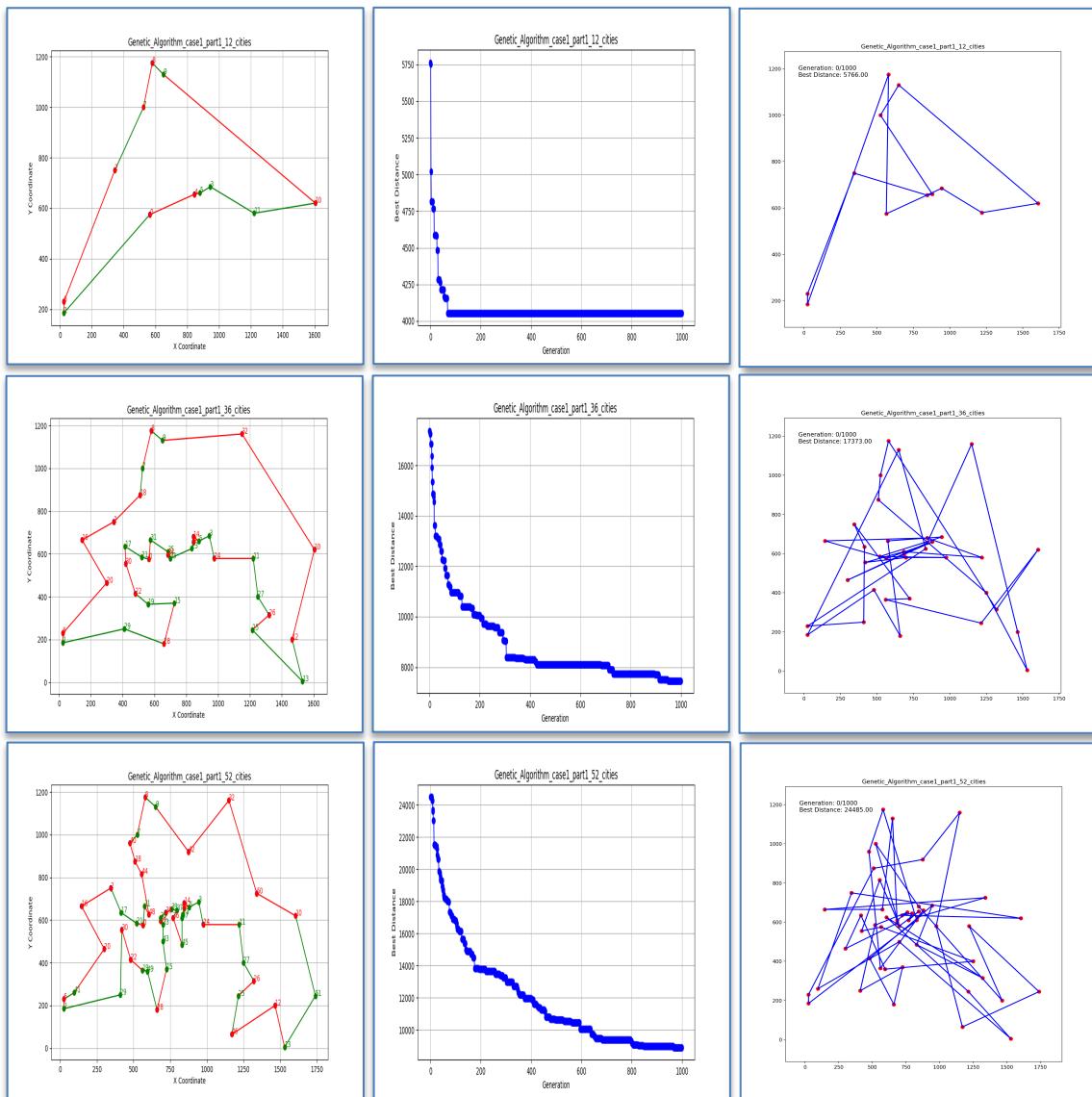
ACO	3	num_ants=20, num_iterations=400, alpha=1, beta=3.5, evaporation_rate=0.5
-----	---	--

第 5 章 結果與分析

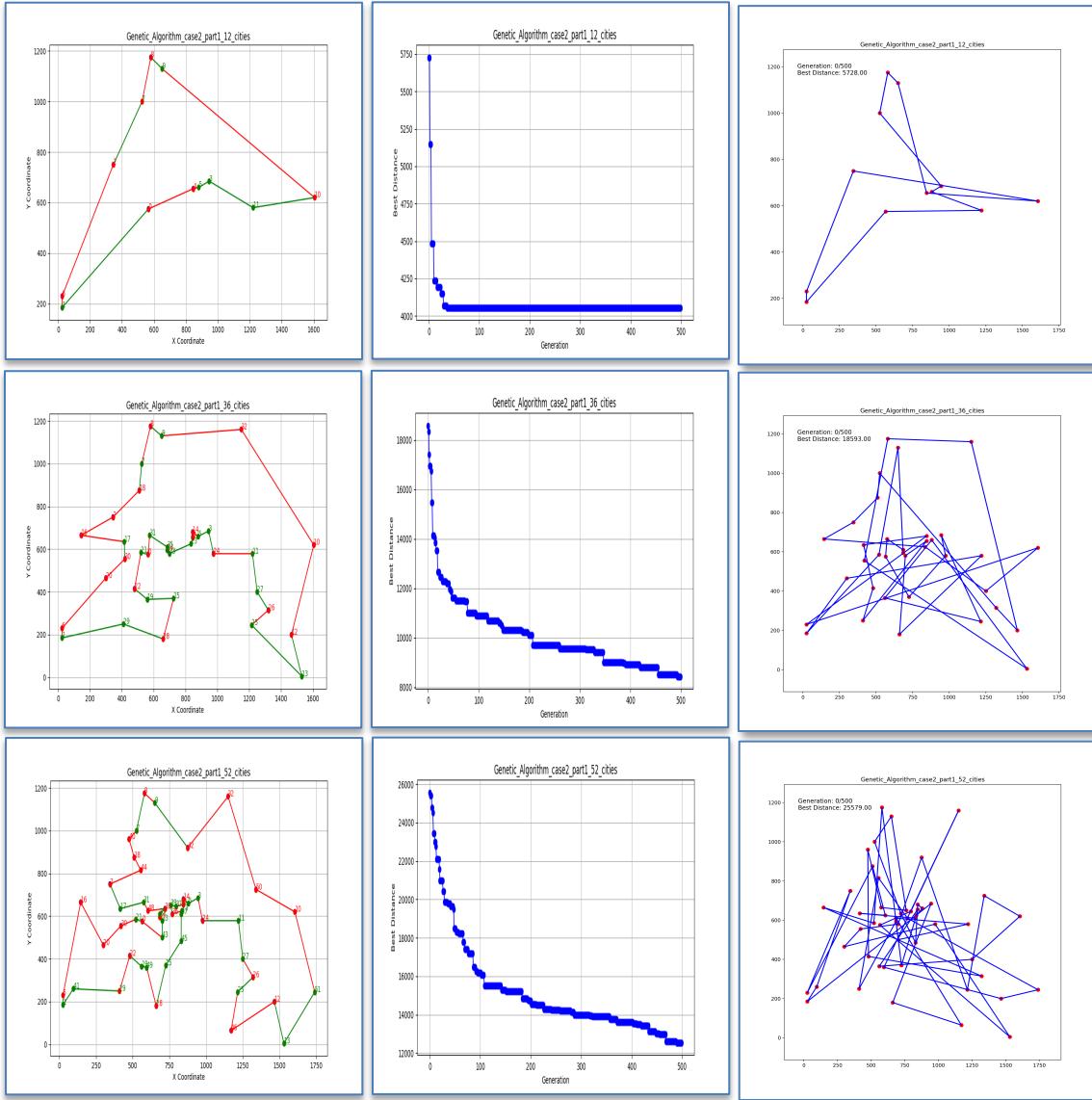
PART I

Experimental Results

GA, case1, 12/36/52 cities

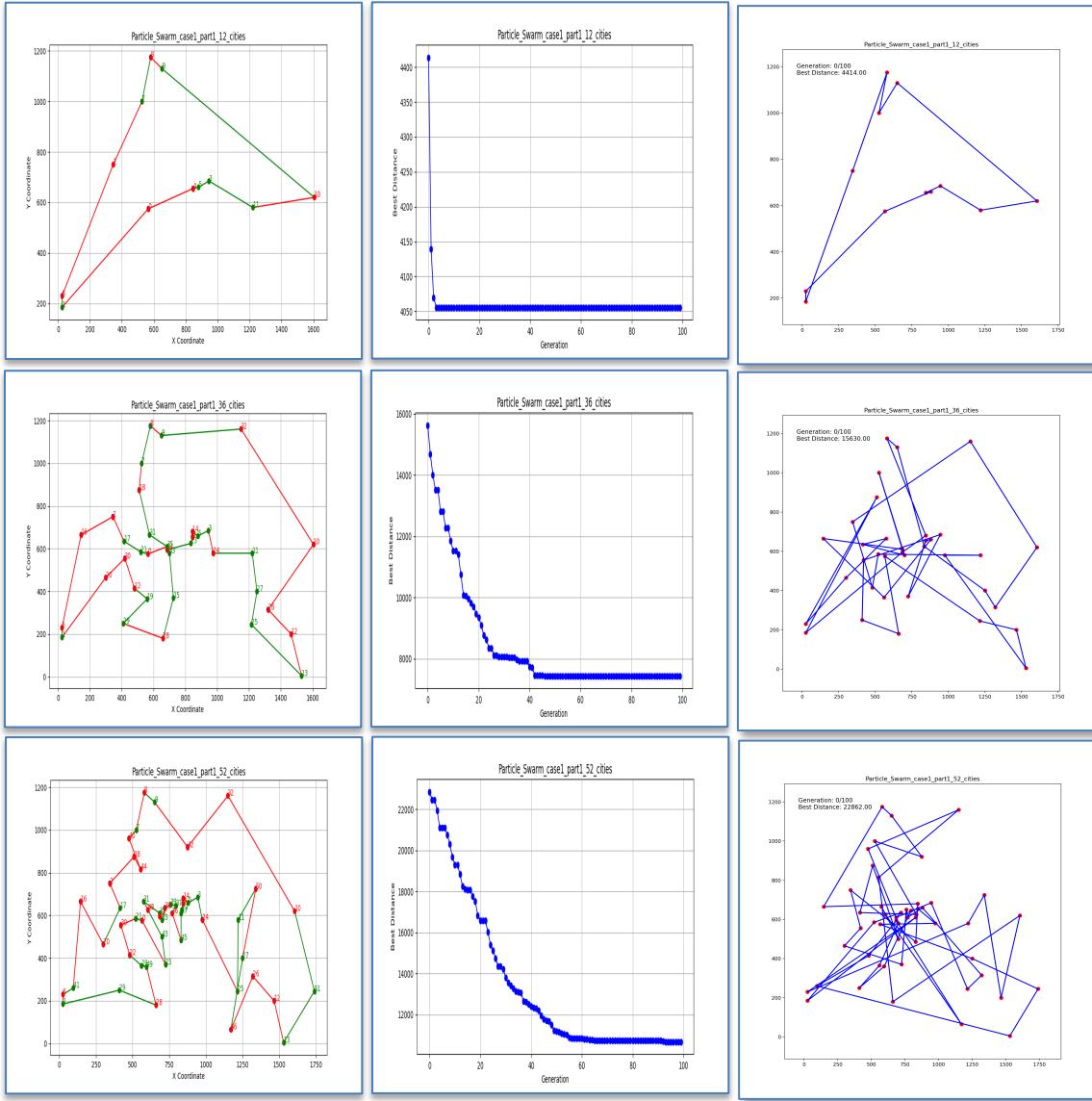


GA, case2, 12/36/52 cities

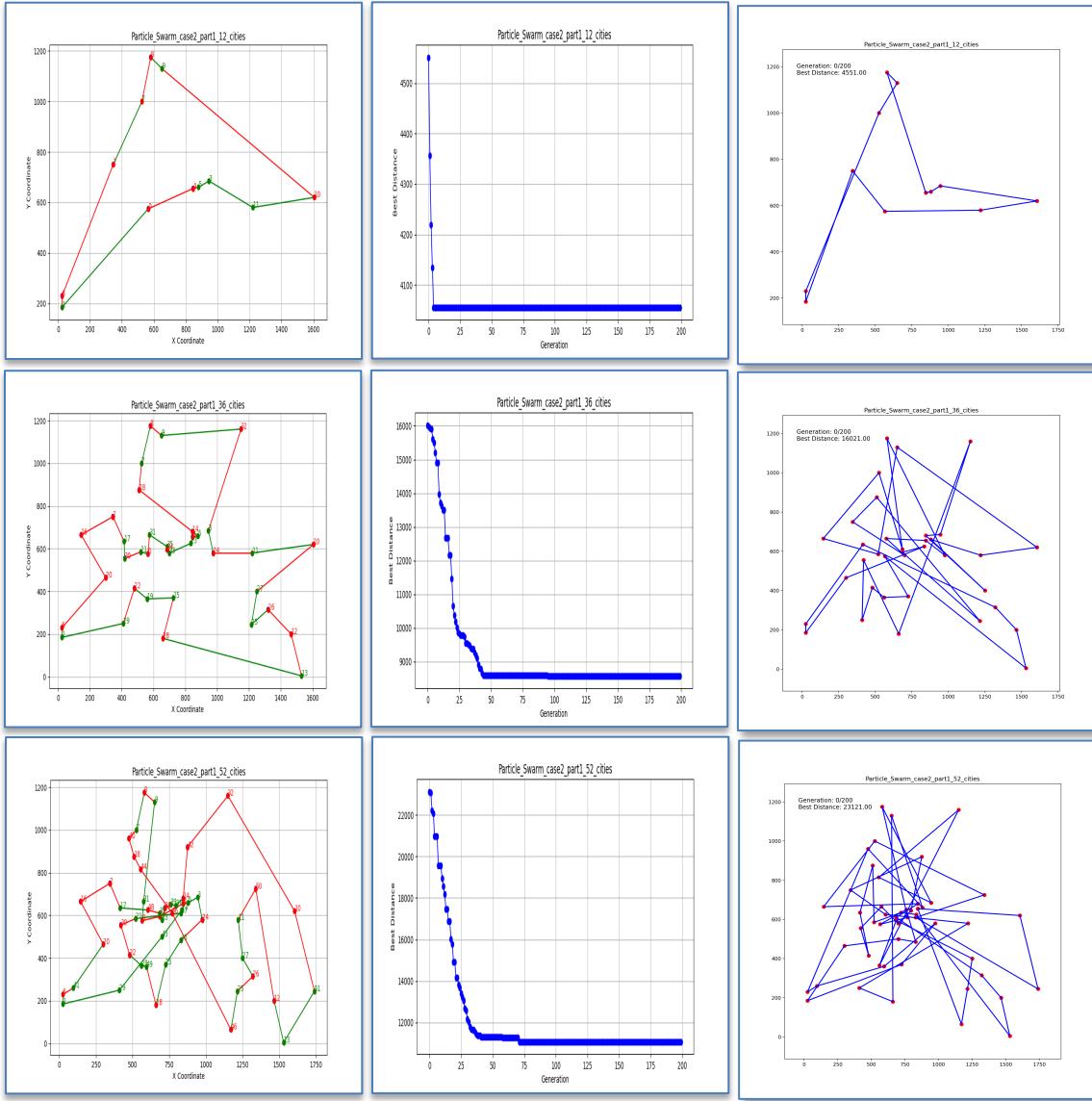


從歷史曲線以及動畫演示中可以觀察到 GA 演算法的可行性，以及其收斂的速度會隨著城市數量越多而越慢穩定。由於演算法會有一定機率發生突變導致城市編碼對調，因此可以在動畫中觀察到實驗最佳解的演化過程會有瞬間大幅度的路線交換。

PSO, case1, 12/36/52 cities

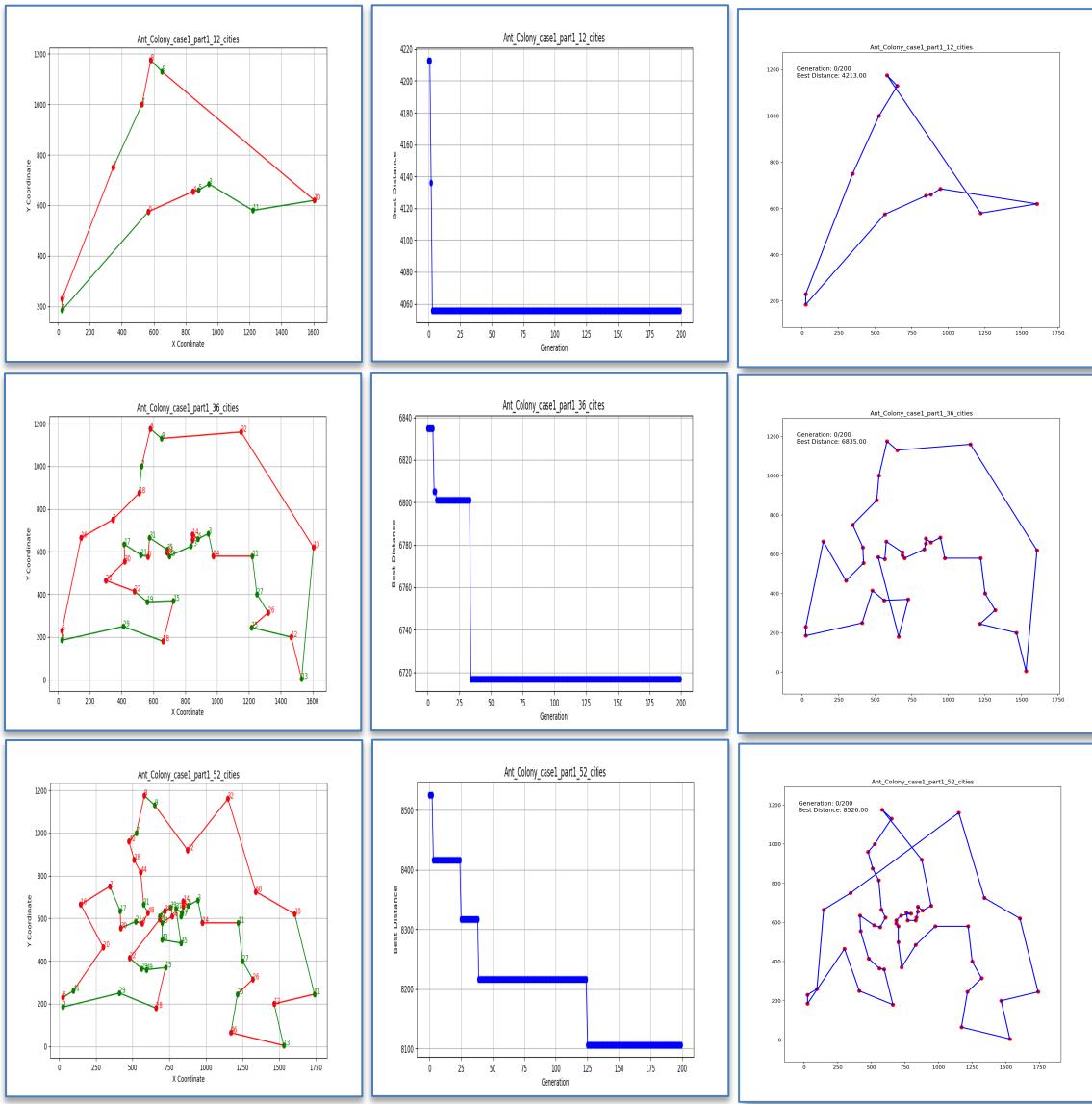


PSO, case2, 12/36/52 cities

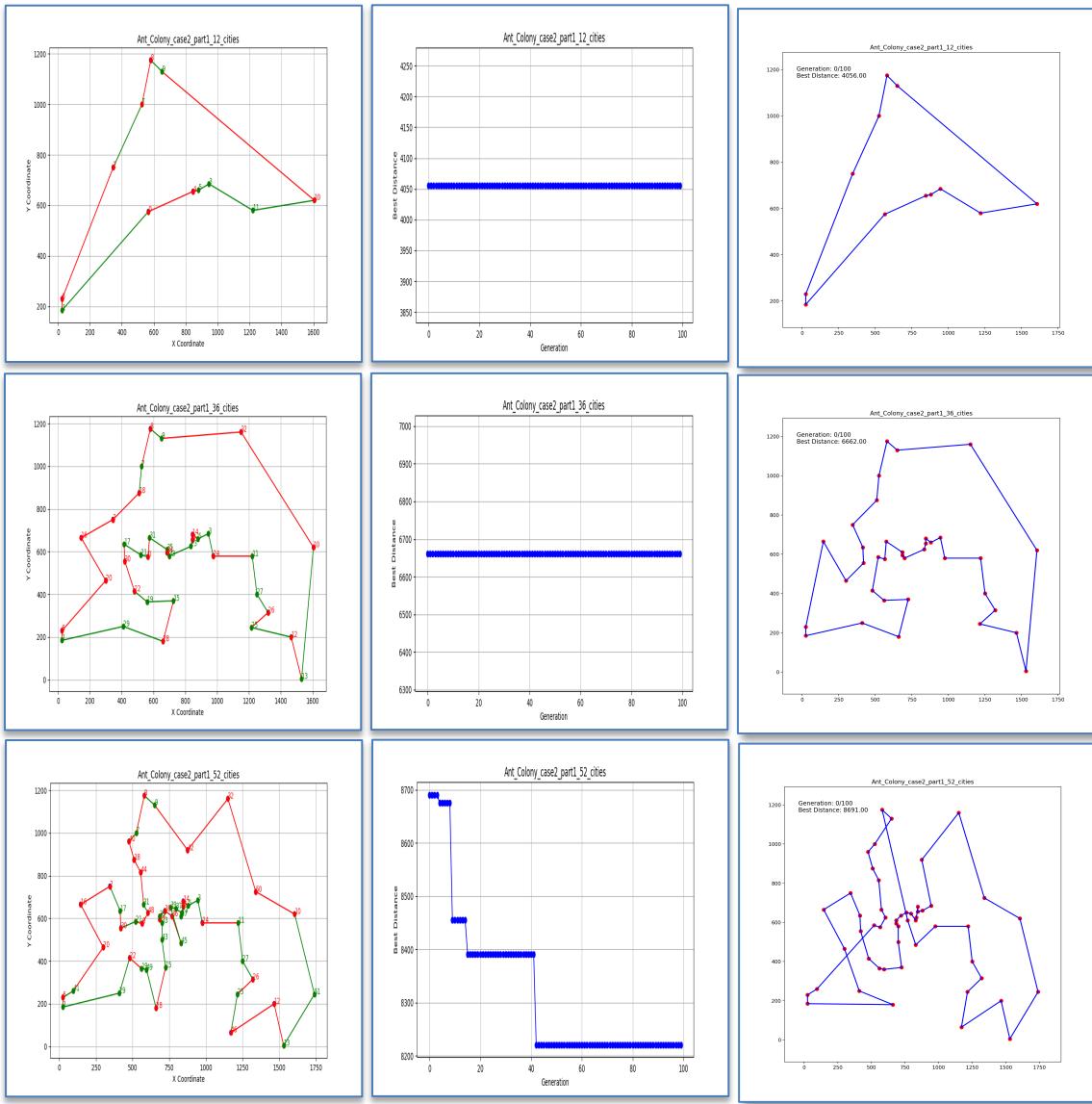


從歷史曲線以及動畫演示中可以觀察到 PSO 演算法的可行性，以及其收斂的速度會隨著城市數量越多而越慢穩定，但相較於 GA 演算法相對較快收斂，也比較容易陷入區域極小值。由於演算法因離散運算的關係將速度變化改為城市編碼對調操作，因此可以在動畫中觀察到實驗最佳解的演化過程會也有瞬間大幅度的路線交換，甚至比 GA 演算法更為頻繁。

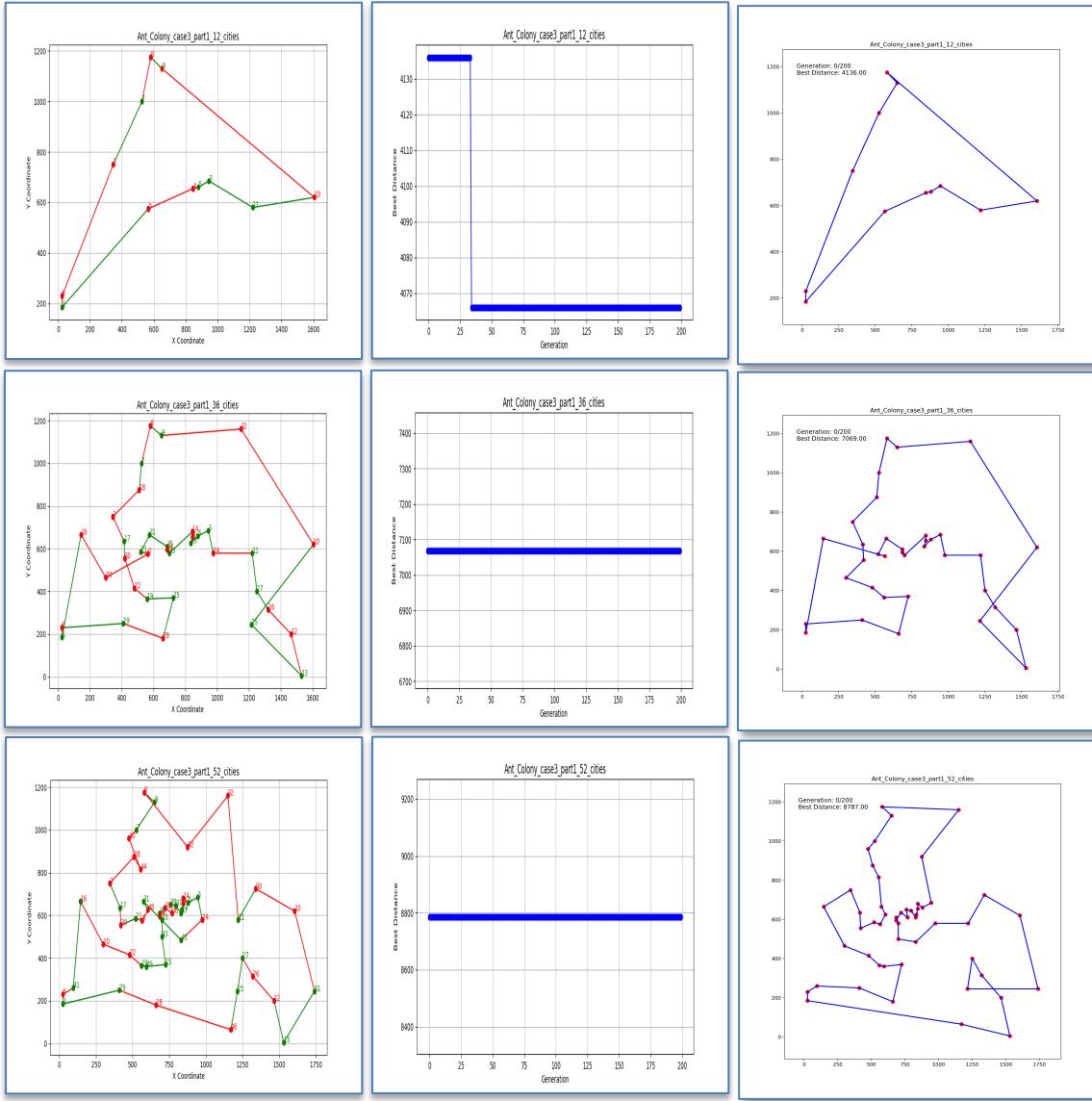
ACO, case1, 12/36/52 cities



ACO, case2, 12/36/52 cities



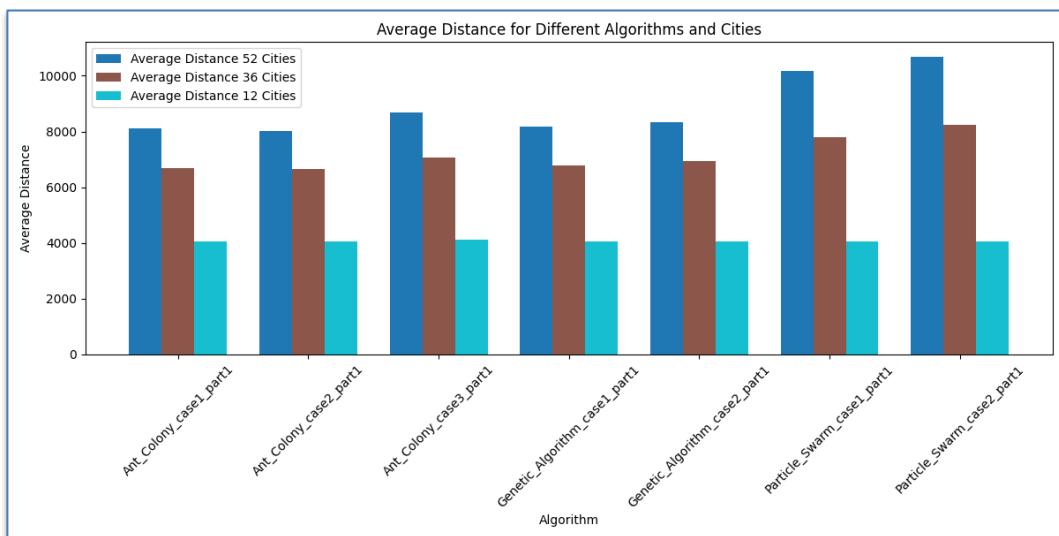
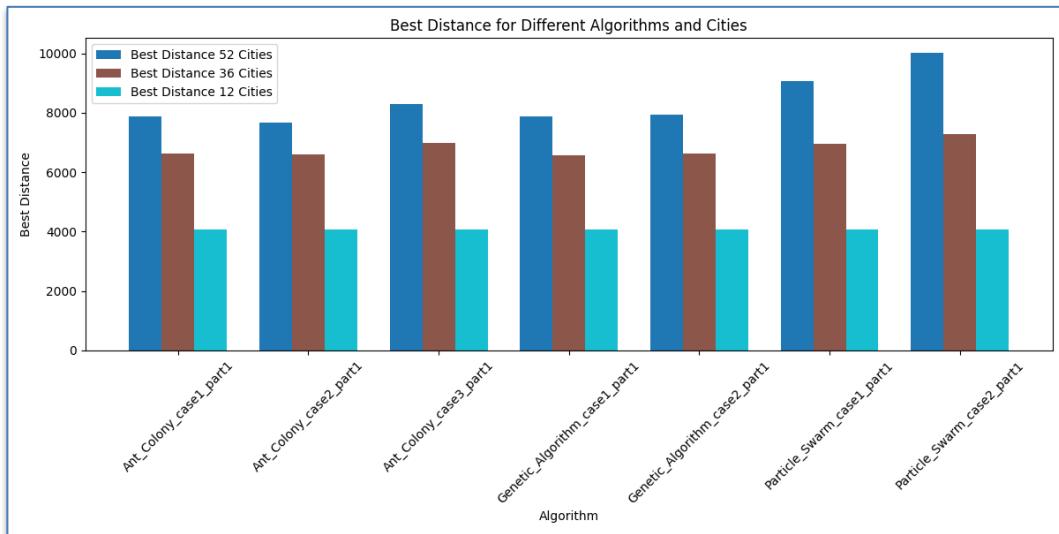
ACO, case3, 12/36/52 cities



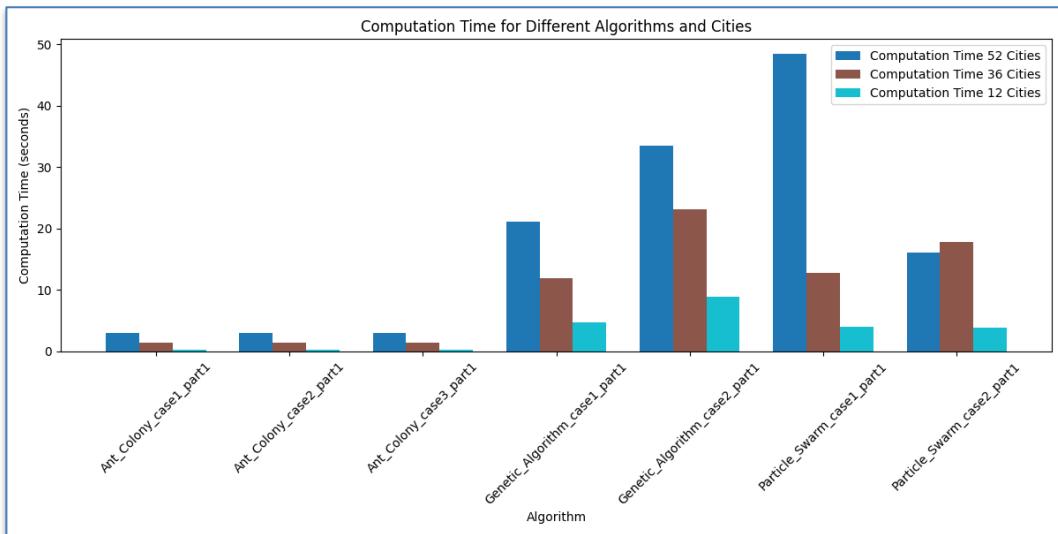
從歷史曲線以及動畫演示中可以觀察到 ACO 演算法的可行性，以及其收斂的速度非常的快。可以觀察到將螞蟻數量由 10 隻增加到 20 隻，或是將信息素的留存率增加，都會更加鞏固原有的路線訊息，因此容易快速陷入區域極小值，而將不再探索新的路線。

Analysis

以下是針對 part I 不同演算法以及參數下的模擬統計數據:



從上面兩張統計圖表可以觀察到，PSO 的穩定性最差，所計算出來的最小距離都偏大，收斂的不夠完整。再者 GA 的演算法雖然推演時間較 ACO 久，但計算出來的結果相對較佳以及穩定。

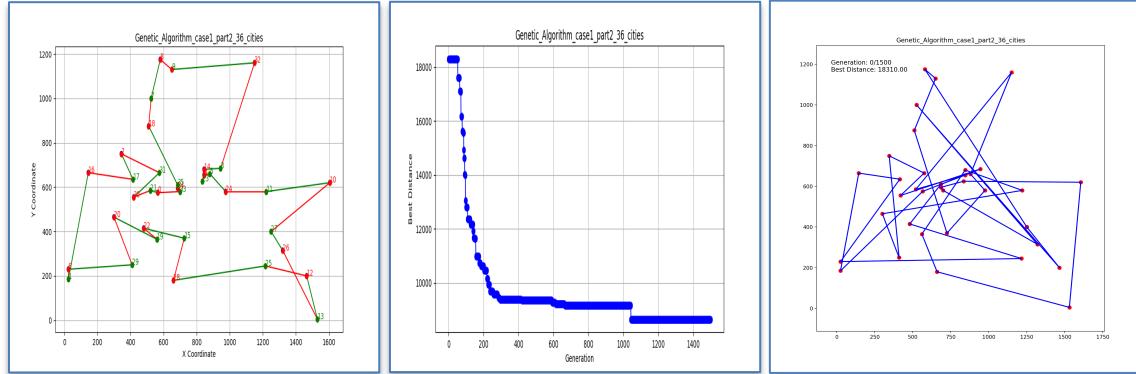


從計算時間上來看，ACO 演算法十分的快速，遠勝過其他兩種演算法。

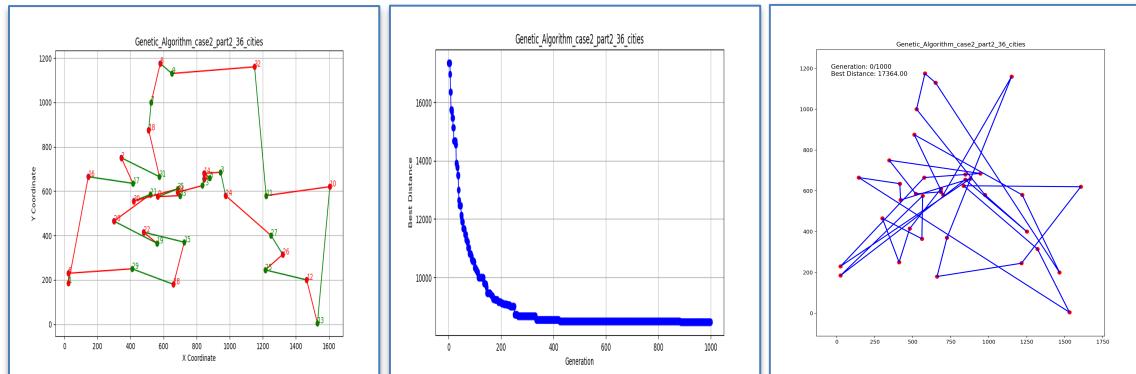
PART II

Experimental Results

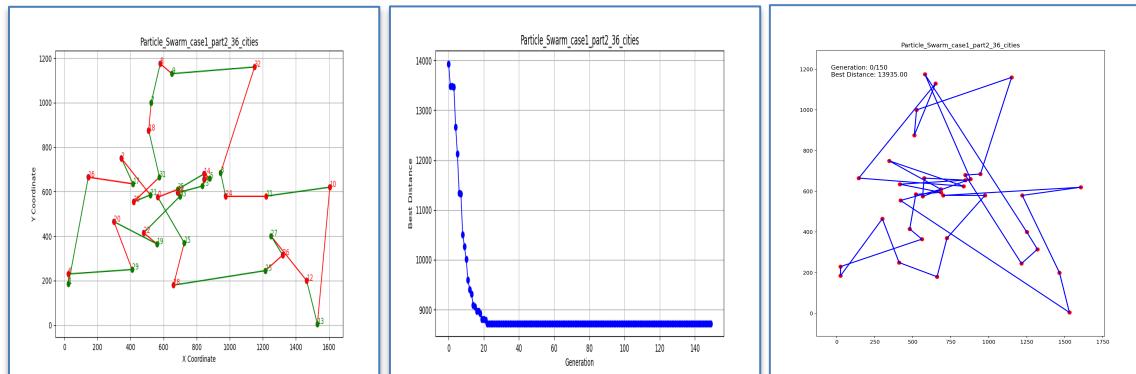
GA, case1, 36 cities



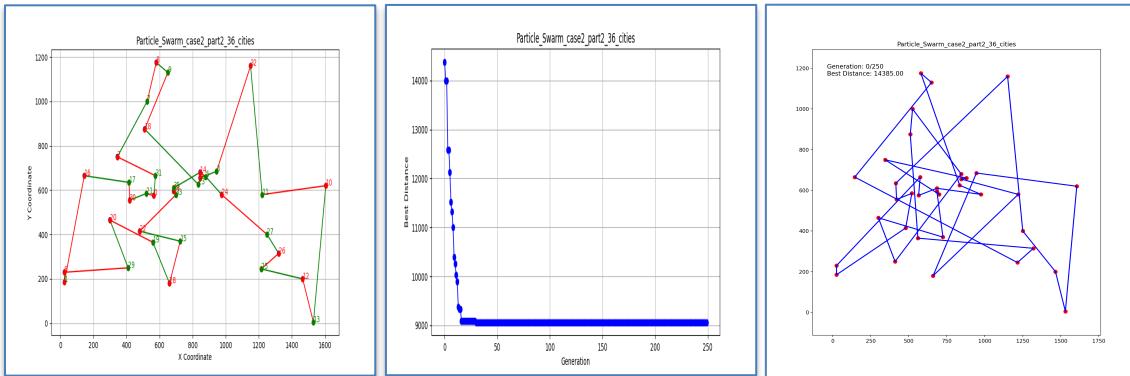
GA, case2, 36 cities



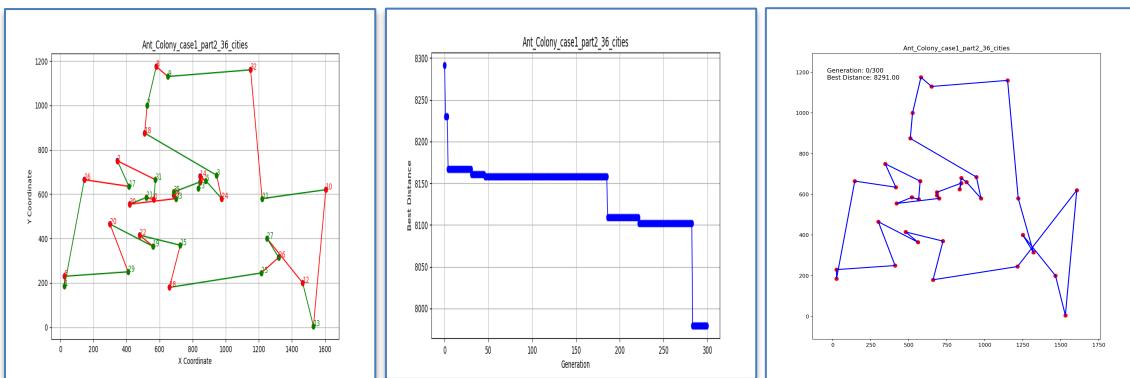
PSO, case1, 36 cities



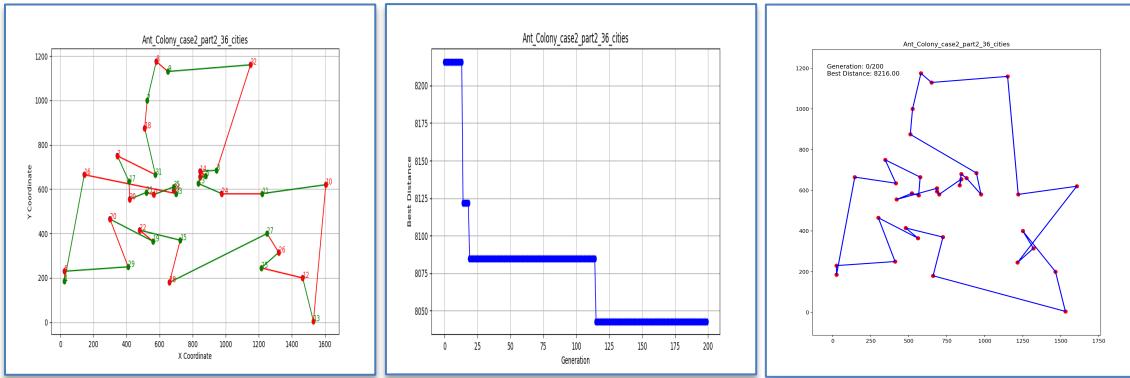
PSO, case2, 36 cities



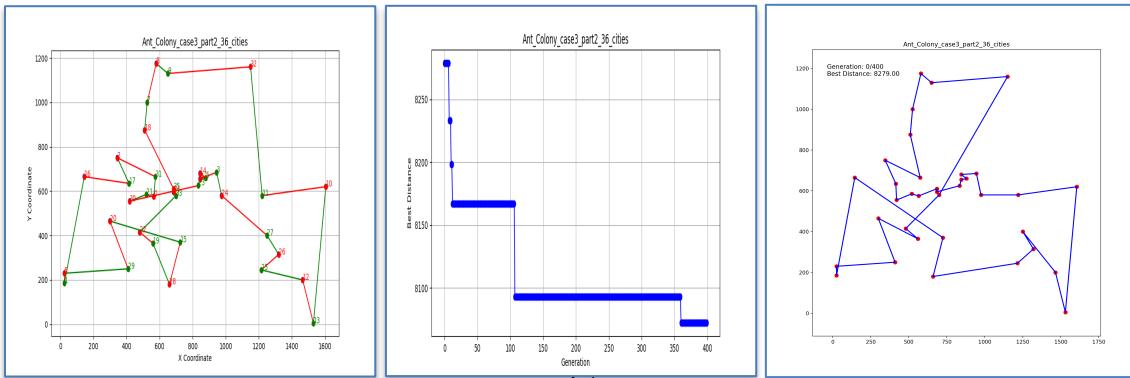
ACO, case1, 36 cities



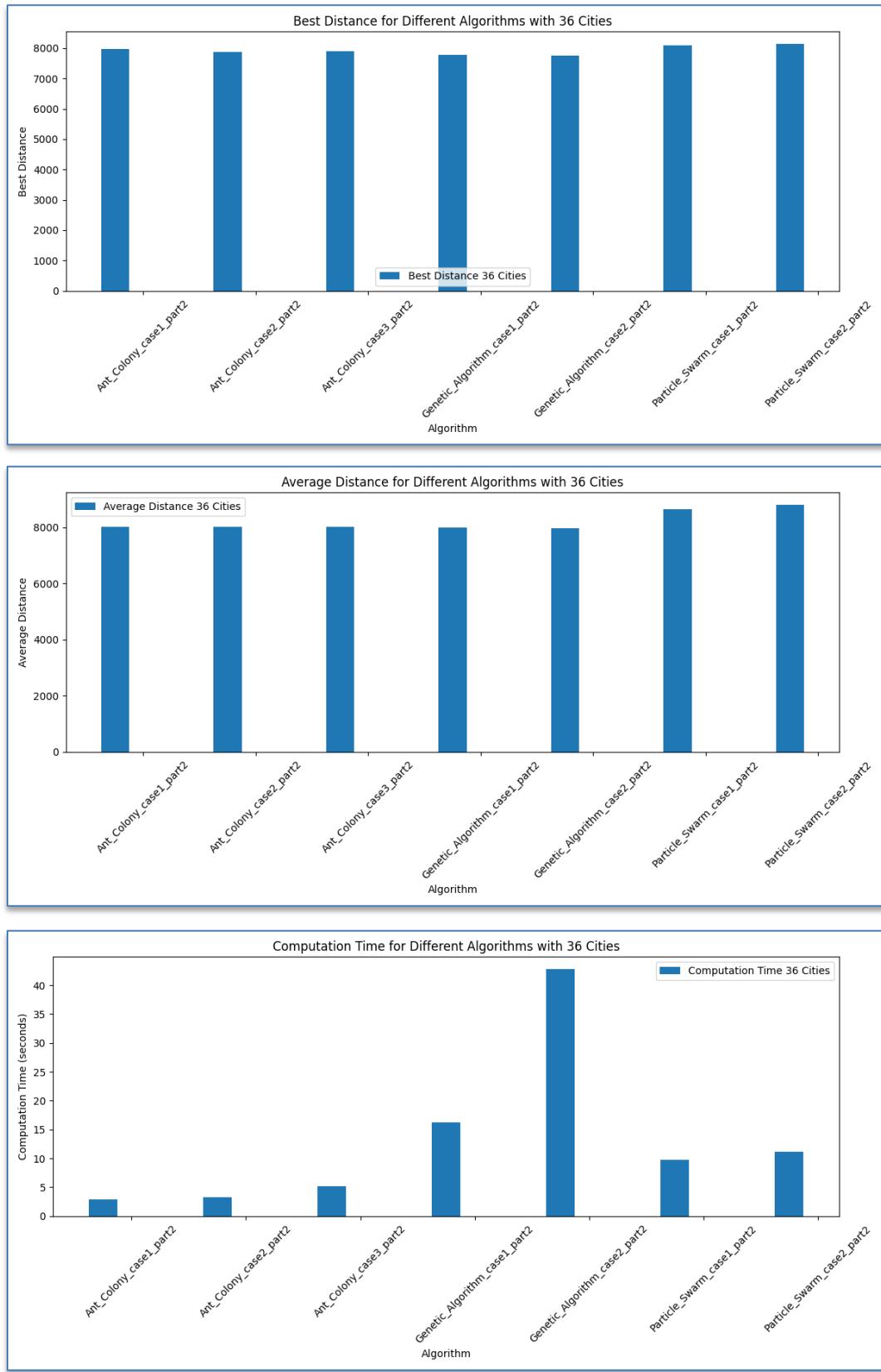
ACO, case2, 36 cities



ACO, case3, 36 cities



Analysis



在 Part II 的實驗結果中可以觀察到每種演算法都因為奇數城市之間不連接的關係導致模型收斂上的困難，就連 ACO 也較難收斂。另外由於 PSO 參數設定的關係，它依然是收斂最快的演算法，當然結果也相對較差。GA 依然是最好找到最佳解的演算法，儘管運算時間較多。

Table of The Part 1 statistic (12 Cities)

Algorithm	Case	Average Distance	Best Distance	Computation Time
GA	1	4056.680	4056.680	4.664
GA	2	4056.680	4056.680	8.884
PSO	1	4056.680	4056.680	3.934
PSO	2	4056.680	4056.680	3.810
ACO	1	4056.680	4056.680	0.223
ACO	2	4056.680	4056.680	0.225
ACO	3	4121.534	4056.680	0.225

Table of The Part 1 statistic (36 Cities)

Algorithm	Case	Average Distance	Best Distance	Computation Time
GA	1	6790.720	6564.721	11.875
GA	2	6936.336	6639.381	23.125
PSO	1	7792.086	6968.425	12.774
PSO	2	8241.828	7292.091	17.854
ACO	1	6698.689	6620.549	1.414
ACO	2	6651.953	6599.564	1.438
ACO	3	7061.704	6995.032	1.419

Table of The Part 1 statistic (52 Cities)

Algorithm	Case	Average Distance	Best Distance	Computation Time
GA	1	8187.813	7880.089	21.081
GA	2	8332.53	7926.841	33.463
PSO	1	10179.187	9063.404	48.396
PSO	2	10675.612	10017.933	16.071
ACO	1	8101.655	7867.887	2.981
ACO	2	8025.308	7683.379	3.023
ACO	3	8675.317	8304.713	3.010

Table of The Part 1 statistic (12 Cities)

Algorithm	Case	Average Distance	Best Distance	Computation Time
GA	1	7984.843	7781.173	16.217
GA	2	7960.151	7764.424	42.788
PSO	1	8632.840	8083.710	9.801
PSO	2	8798.415	8138.418	11.197
ACO	1	8019.556	7980.196	2.930
ACO	2	8021.281	7885.072	3.268
ACO	3	8017.4516	7903.935	5.206

第 5 章 結論

本研究針對基因演算法（Genetic Algorithm, GA）、粒子群最佳化（Particle Swarm Optimization, PSO）以及螞蟻演算法（Ant Colony Optimization, ACO）在旅行商問題（TSP）上的表現進行了深入比較與分析。透過對不同城市數量（12、36、52）的測試，綜合評估了各演算法在最佳距離、平均距離以及計算時間上的表現。

1. GA 演算法

GA 演算法展現了穩定的收斂能力，特別是在多數測試案例中能夠找到接近最佳解的結果。從歷史曲線與動畫演示中可以觀察到，GA 的收斂速度會隨著城市數量的增加而減慢，但其穩定性較高，且在多峰問題中能有效避免過早陷入局部極小值。然而，由於突變操作的隨機性，最佳解的演化過程中偶爾會出現大幅度的路線交換，導致收斂過程稍顯波動。

2. PSO 演算法

PSO 演算法在收斂速度上表現優異，尤其在城市數量較少的情況下能快速找到解。然而，隨著城市數量的增加，PSO 容易陷入局部極小值，導致最終解的品質不如 GA 或 ACO。動畫演示顯示，由於 PSO 的離散運算特性，其路線交換頻率較高，且演化過程中經常出現大幅度的路線變化。儘管如此，PSO 在計算效率上仍具優勢，適合用於對時間要求較高的場景。

3. ACO 演算法

ACO 演算法在計算時間上遠勝於其他兩種演算法，且在小規模問題中展現了極快的收斂速度。然而，ACO 的穩定性較差，特別是在螞蟻數量增加或信息素留存率提高的情況下，容易快速陷入局部極小值，導致探索能力下降。從統計圖表中可以觀察到，ACO 在多數測試案例中計算出的最小距離偏大，顯示其在複雜問題上的表現仍有提升空間。

4. Part II 實驗結果

在 Part II 的實驗中，由於奇數城市之間不連接的特性，所有演算法的模型收斂均面臨挑戰。即便是 ACO 也難以快速收斂，而 PSO 雖然收斂速度最快，但結果品質相對較差。GA 則憑藉其穩定性與探索能力，依然能找到較佳的解，儘管運算時間較長。

綜合結論

在本研究中，GA 演算法整體表現最為穩定，適合用於需要高精度解的場景；PSO 演算法則在計算效率上具優勢，但需改善其避免局部極小值的能力；ACO 演算法適合用於小規模問題或對計算時間要求較高的場景，但在複雜問題中仍需優化其探索能力。未來研究可考慮結合多種演算法的優勢，設計混合式演算法以提升解決 TSP 的效率與精度。

本研究結果為不同演算法在 TSP 上的應用提供了實驗依據，未來可針對更大規模的問題或結合混合式演算法進行進一步研究，以提升解決複雜優化問題的能力。

第 6 章 實驗結果

如附件資料