以基因演算法最佳化基因演算法參數
-以 TSP 問題為例

基因演算法與管理科學應用期末報告

第二組

0711239 李勝維

0713218 蔡沛瑀

H092638 廖淯舜

目錄

一、 前言

二、 研究背景與動機

三、 研究目的

四、 研究方法

4.1 程式架構簡介

4.2 Child Genetic Algorithm

4.3 Parent Genetic Algorithm

五、 實驗設定與成果

5.1 參數設定

5.2 參數比較

六、 結論

七、 參考資料

一、 前言

本組的期末專題報告採取以「自己訂定一個有興趣實作的主題」之方式進行，題目為在旅行推銷員問題(Traveling Salesman Problem)環境下，實作以基因演算法最佳化基因演算法參數。

本組在 TSPLIB 網站中挑選出 7 張地圖作為欲解決之問題環境，藉此探討本次實作之效率、成果。本組將應用課程中所學基因演算法之知識於本次實作，並參考其他文獻中的競爭、交配等機制，嘗試改善本組的演算法。

二、 研究背景與動機

基因演算法可謂現今非常普遍的一種啟發式演算法，應用於許多問題中的最佳解搜尋。而基因演算法的重要機制中包含了擇偶、交配、突變。也因此，使用基因演算法時需要調整程式內的參數，如迴圈數、染色體數目、交配率及突變率等，以達到接近或求得最佳解的目的。

綜覽中外文獻，傳統上的基因演算法，抑或課程中使用的最基本基因演算法中，程式內的參數都需要以人工的方式手動調整。通常人們會為這些參數設定一區間，至於確切的數值應設定在多少，方能在世代中得出最好的適應度，將會因問題環境不同而有所差異。我們在文獻中常能看見的做法是對各項參數進行敏感度分析，藉著數值的改變，逐個數字觀察執行結果的差異，最後獲得適合的參數設定組。

如此一來，面對到問題規模很大或限制條件很繁雜的最佳化，調整參數就會相當耗費時間與電腦效能。舉例而言，我們曾在課堂中實作流水車間調度問題(Job Shop Scheduling)，在此課題下，目標為最小化排成總製造時間，而學生需要透過調整各種基因演算法中的參數，讓最大完工時間(makespan)愈小愈好，且落在規定之區間內。本組在實作時即花費相當多時間才找到可接受的解。倘若在環境限制更加複雜的實務問題中求解，想必將消耗更多時間才能得到品質好的答案。

這個問題同時帶出了本組的研究動機，若能做到非依靠人工逐一輸入參數，而是以撰寫電腦程式的做法，自動地幫助我們挑選在現有題目環境下最適合的一組參數，此舉即可省去多餘的勞力成本和時間浪費。

其中一種可行的做法為透過執行迴圈，嘗試各參數區間內的每一個可能數字。但此作法類似敏感度分析，且當要求的數字精度提高，將無效率可言。在尋找相關資料的過程中，存在不少以基因演算法調整強化學習(Reinforcement Learning)中參數的文獻，但並未發現針對基因演算法本身參數的自動調整方法，這也讓我們覺得該方面的相當新奇，成為我們實作的動力。

綜上原因，本組在自動選擇演算法參數為研究動機下，我們將本次報告主題定為「以基因演算法決定基因演算法的參數」，而我們挑選旅行推銷員問題作

為需解決的環境，期望能在此做法下得到最好的參數組。

三、 研究目的

綜第二部份所述原因，本組在自動選擇演算法參數為研究動機下，我們將本次報告主題定為「以基因演算法決定基因演算法的參數」，而我們從 TSPLIB 中挑選 7 張地圖大小不同的旅行推銷員問題作為本次實作環境，目的為以下兩點：

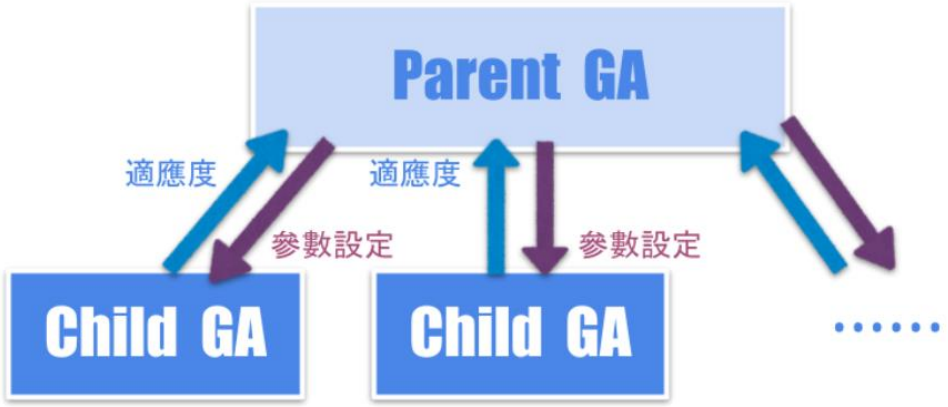
（１）製作由自動方法(同為基因演算法)決定基因演算法參數的演算法模型。

（２）探討自動化決定參數的基因演算法於本次實作中是否在效率上優於傳統基因演算法。

四、 研究方法

4.1 程式架構簡介

為了實現本次的研究目標，我們分成兩部分撰寫基因演算法，分別為自動調整參數的 Parent Genetic Algorithm 及實際對旅行推銷員問題求解的 Child Genetic Algorithm，Parent Genetic Algorithm 提供自動設定的參數給 Child Genetic Algorithm，而 Child Genetic Algorithm 在該參數組下求解並回傳適應度給 Parent Genetic Algorithm，如下圖所示。



（圖一）本次演算法簡略架構圖

以下將分別介紹兩部分之編碼方式與適應度計算方式，並輔以程式碼說明實作中較特別或課程中未提及的程式寫法。

4.2 Child Genetic Algorithm

I. 編碼方式

在 Child Genetic Algorithm 部分中，染色體的形式採用排序編碼

(permutation encoding)，染色體中的每一基因(genome)為整數數字字元，存放對應的旅行推銷員問題中所有城市點，而城市點在染色體中的排列順序即為造訪城市的順序。

編碼範例

X = [1 2 3 4 5]

假設有一世代中的染色體如上列所示，則此染色體代表該次 TSP 問題中的旅行推銷員將以「場站-顧客點 1-顧客點 2-顧客點 3-顧客點 4-顧客點 5-場站」之順序進行。

而 Child Genetic Algorithm 產生初始世代的方法與課堂中所授內容相同，以隨機方式產生，且禁止多次拜訪同一城市點，故單一染色體中部會出現重複字元。

II. 解碼與適應度計算

由於本組採用以排序編碼方式撰寫 Child Genetic Algorithm，因此無須解碼。而適應度的計算在此部分的目的為評估對於當前旅行推銷員問題的解之品質，我們將適應度函數定義為計算各染色體之旅行順序中，所有城市點間的旅行距離加總，且因此最佳化問題之目標式為旅行成本最小化，故於程式中回傳染色體中路經加總長度之負值。

III. 交配方式

本組最初參考 Abid Hussain 等人（2017）對基因演算法運用於旅行推銷員問題的研究，嘗試使用一種名為 Cycle Crossover Operator 的交配方式，惟因實作完成後，此交配方式之效能大幅劣於課程教材使用的均勻交配法（Uniform Crossover），故本組最後仍使用均勻交配法。其運作方式如下：

P1	1	2	3	4	5
P2	2	4	5	3	1
Mask	0	1	0	0	1
O1	4	2	3	1	5
O2	2	4	3	5	1

（圖二）均勻交配法示意圖

假設經二元競爭選擇法後，挑選出兩親代染色體 P1 與 P2 進行交配，在均勻交配做法下，程式將會隨機產生一組遮罩的編碼，其任何位置的值皆為二元數字 0 或 1。在此遮罩的作用下產生子代染色體 O1，O1 中會從 P1 裡直接繼承遮罩中值為 1 時，對應位置裡面的數字，此例中為 2 和 4。而尚未被放入 O1 中的顧客點 1,3,5 便會以此三點在 P2 染色體中的排序方式再度繼承至 O1，在表中以顏色幫助理解。程序完成後，新染色體 O1 中基因排序即為 5,2,1,4,3；染色體 O2 亦是透過相同程序所求得，差異僅是

P1 跟 P2 的執行內容互換，故不重新敘述。

IV. 突變方式

由於現行許多突變方式應用在旅行推銷員問題上容易產生不可行解，因此本組選定的突變方式為簡單地將單一染色體中兩個基因體互換，意即將兩座城市的造訪順序對調。具體程式碼如下：

```
def mutation(p):
    for _ in range(NUM_CHROME):
        if TF("Pm"):
            row = np.random.randint(NUM_CROSSOVER_2)
            [j, k] = np.random.choice(NUM_BIT, 2)
            p[row][j], p[row][k] = p[row][k], p[row][j]
```

（圖三）Child Genetic Algorithm 中突變方法之程式碼

其中函數 TF()的作用為根據突變率做為權重而隨機回傳 True 或是 False，因為課程教材的實作方式是直接使用期望值（染色體數目乘以突變率）來當作每次迴圈執行突變的次數。但是，若突變率太小導致執行期望值小於一的情況發生，在經過整數轉換後的突變次數為零，我們認為這是一個程式邏輯上的缺陷。

V. 主要程式執行部分

```
pop = initPop()
pop_fit = evaluatePop(pop)
memory = MEMORY(capacity=len(TSP_graph)**2)

for i in range(1, MAX_NUM_ITERATION+1):
    parent = selection(pop, pop_fit)
    offspring = crossover_uniform(parent)
    mutation(offspring)
    offspring_fit = evaluatePop(offspring)
    pop, pop_fit = replace(pop, pop_fit, offspring, offspring_fit)
    mean = -1 * np.average(pop_fit)
    if mean == memory.get():
        return i
    memory.add(mean)

return 2*MAX_NUM_ITERATION
```

（圖四）Child Genetic Algorithm 中主要執行部分

由於效率是本組實作的一大考量，為減少運算量，我們為 Child Genetic Algorithm 設定了最大迴圈數上限（*MAX_NUM_ITERATION*），如果

執行時無法在迴圈上限之內收斂，則會放棄這次的求解，並回傳兩倍的迴圈數上限當作懲罰（penalty），藉此促使程式執行能更快收斂。而判斷收斂的方式是透過我們自己撰寫的 *memory class* 來完成，每次的 *memory.get()* 會回傳最近 *capacity* 次結果的平均，而如果本次解的平均結果和 *memory* 內儲存的值相同，就當作其已收斂，並結束這次的求解。而在經過幾次實驗後，我們選定每次的 *capacity* 值等於 TSP 地圖中城市數的平方（n^2）。

最後，得到交配率、突變率、染色體數目和要求解的 TSP 地圖等輸入參數的 Child Genetic Algorithm 將會輸出迭代直到收斂所需要的迴圈數給 Parent Genetic Algorithm。

4.3 Parent Genetic Algorithm

I. 編碼方式

在 Parent Genetic Algorithm 部分的染色體編碼中，我們決定選擇以值編碼（value encoding）的方式處理。因為在經過本組實驗後，相較於二元編碼（binary encoding），我們認為值編碼的優點是它能夠頗有效的區分每個基因體而不互相影響，此特徵有利於在交配時保留親代好的基因特徵。而染色體中三個基因體分別代表交配率、突變率、染色體數目。

由於電腦計算能力與時間等資源限制，我們在實務上必須減少 Parent Genetic Algorithm 對於參數的搜尋空間，而本次採用的方式是對每個參數都預先給定一個合理的範圍，讓演算法在此區間內自動搜尋解，此舉可以去除每種參數多餘的可能性。設定的範圍如下：

參數範圍限制	區間
交配率	[0.6 , 1]
突變率	[0 , 0.01]
染色體數目	[30 , 50]

II. 解碼及適應度計算

本組在編碼方式上使用了以值編碼，因此不需要解碼。而適應度的計算方式是透過同樣的參數組設定對每個 TSP 地圖（本次專題中選擇了 7 張地圖）都進行一次 Child Genetic Algorithm 的計算，而我們將這組參數的適應度設定為在這 7 張地圖求解時所花費的平均迴圈數，目標為平均迴圈數的最小化，其原因是本組的研究目的希望人工挑選參數的作法能在執行效率上越快越好。具體程式碼如下：

```
def fitFunc(self, x):
    '''
    Definition of fitness function:
    average of iterations for each map
    '''
    Pc, Pm, chrome = x
    sum_ = 0
    for graph in self.TSP_maps:
        sum_ += float(Child_GA(Pc=Pc, Pm=Pm,
                               NUM_CHROME=chrome, TSP_graph=graph))
    fit_value = sum_ / len(self.TSP_maps)
    return fit_value
```

（圖五）Parent Genetic Algorithm 中適應度的定義

III. 擇偶方式

Parent Genetic Algorithm 中的親代擇偶方式，我們使用了排名式輪盤選擇法（rank based wheel selection），是一種為適應度分配排名的方法。本組希望能夠透過輪盤選擇法（wheel selection）優先挑選較「優秀」的親代做交配，但礙於相似的參數通常在運算後會得到相似的適應度，若直接以適應度來當作選擇的權重，則很容易重複挑選到類似的參數設定。

為了突破此一困境，我們使用了排名式的輪盤選擇法。和基本的輪排選擇法不同，本組的作法會先將整個族群內的染色體依照適應度排名，而排名越靠前的染色體將被賦予越大的權重。透過這樣的方式，即便是相似的參數，也必定會有不同的排名，亦會凸顯出兩者被分配到的權重差異，如此一來即可以避免以往太過容易挑到相似參數的問題。具體的程式碼如下：

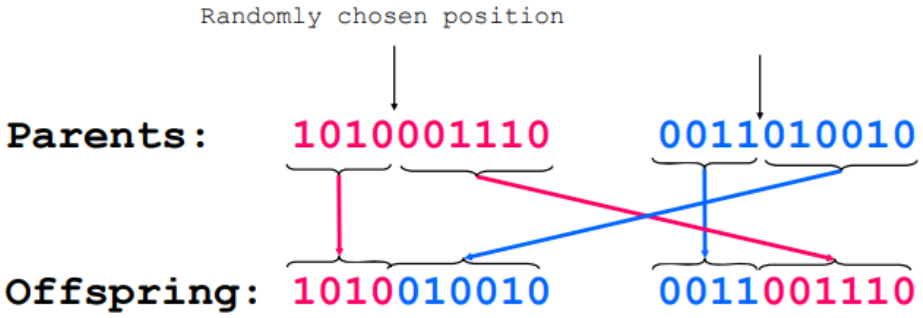
```
def selection(self, p, p_fit):
    '''
    Rank selection
    '''
    a = []
    sorted_p = [x for _, x in sorted(zip(p_fit, p), reverse=True)]
    weights = list(range(1, len(sorted_p)+1))
    for _ in range(self.NUM_PARENT):
        parent = random.choices(sorted_p, weights=weights)[0]
        a.append(parent)
    return a
```

（圖六）Parent Genetic Algorithm 中擇偶方法之程式碼

在總共有 N 個染色體進行交配時，適應度最佳的染色體排名第一，被賦予的權重為 N，排名第二染色體則是 N-1，以此類推。

IV. 交配方式

由於本組希望在 Parent Genetic Algorithm 中挑選參數時能保持染色體中每個基因的獨立，即保留基因的特徵，故採用單點交配法（one-point crossover），讓程式挑選一隨機位置，在此位置前的部分來自親代染色體 P1，在此位置後的部分來自 P2，組成新的子代染色體 O1；O2 亦進行相同程序，只是從 P1 與 P2 繼承的基因相反。



（圖七）單點交配法示意圖，擷取自課程講義《GA05-GA-basic-2》

V. 突變方式

在本組設備與時間資源的限制下，Parent Genetic Algorithm 無法進行迴圈數太多、世代中染色體數目太大的計算。因此，我們期望染色體中的基因在經過突變機制後能夠保留些許原本的特徵，即突變後的變化不要過大。在此部分我們使用了 Uniform mutation 的機制，具體做法是每次發時突變時選擇一個基因（代表著一個參數種類），並將它重新做初始化，圖例如下：

（圖八）Uniform mutation 機制範例

此範例染色體中隨機選擇到以「突變率」為進行突變的項目，此時會將該基因內的數值初始化，方法是隨機分配一個突變率區間（0 至 0.01）內的數值給它，就完成一次突變。

五、 實驗設定與成果

為了確定本次研究所利用的 Parent GA 是否有調整一組近似最佳的參數及達到快速收斂的目的，因此我們透過手動調整三個參數以找出最好的搭配(後稱“**專家設定**”)，來與本研究的設定做比較。

5.1 參數設定

本研究透過調整三個參數:**交配率**、**突變率**及**染色體數目**，在選擇的 7 個 TSP instances[3]中，求得每一組參數的收斂迭代數並將之平均:$\frac{\sum_{i=1}^{n} iter_i}{n}$，$\boldsymbol{n}$:TSP instances 的個數(本研究$n$=7), $\boldsymbol{iter_i}$:每個 instance 收斂所需迭代數。

I. Parent GA

Parent GA 目的在於解決 Child GA 所解問題的最佳參數設定，但其也需設定參數如(表一)。

(表一) Parent GA 參數設定

參數	**數值**
交配率	**0.8**
突變率	**0.01**
染色體數目	**5**
迭代數	**20**

考慮到訓練時間有限，因此在染色體數目及迭代數上數值設計的較小。其實驗數據如(表二)、(表三)、(表四)

(表二) iteration 1 數據

交配率	**突變率**	**染色體數目**	**最佳適應值**
0.8019	0.002818	33	3791.286
0.8019	0.002818	33	4102.571
0.7620	0.007838	39	4138
0.8019	0.002818	33	4154.429
0.9378	0.007580	43	4218.429

(表三) iteration 10 數據

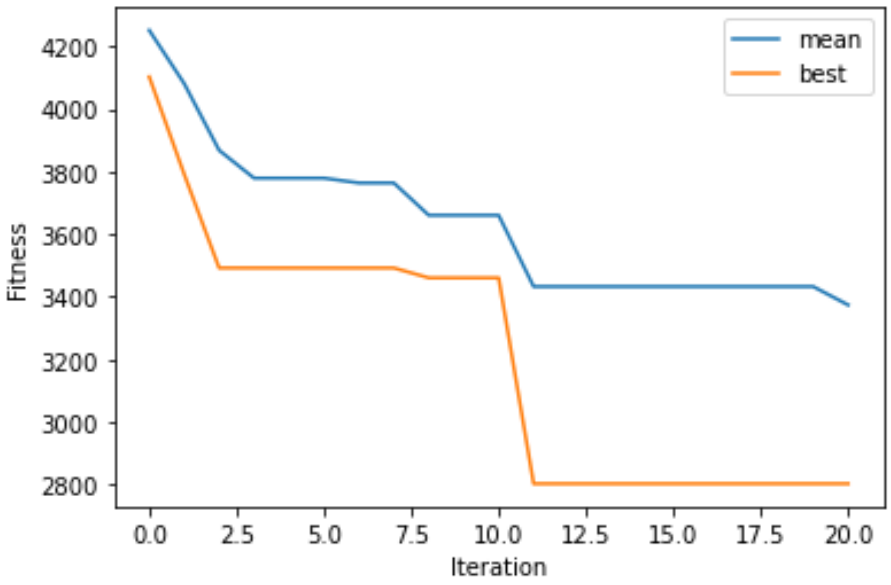
交配率	突變率	染色體數目	最佳適應值
0.7620	0.002818	33	3460.143
0.7620	0.002818	33	3491.714
0.7620	0.002818	33	3756.429
0.8019	0.002818	33	3791.286
0.8019	0.002818	33	3801

(表四) iteration 20 數據

交配率	突變率	染色體數目	最佳適應值
0.7620	0.002818	33	2802.286
0.7620	0.002818	33	3460.143
0.7620	0.002818	33	3462.286
0.7620	0.002818	33	3491.714
0.7620	0.002818	33	3650.714

從上述三張表可以看得出來此模型是有擷取到過去親代優良的基因，以交配率為例，iteration 1 共有三種不同的交配率，而隨著訓練到中期 iteration 10 就只剩下兩種，最後結束訓練時就剩一種交配率。

從(圖九)可以看得出來在大約 iteration 11 時就已經收斂了，但是兩條線數值卻還是相差相當的多，這邊歸納出的結果是染色體的數目太少，導致每條染色體的影響成分太多。



(圖九) Parent GA 收斂曲線

II. 專家設定

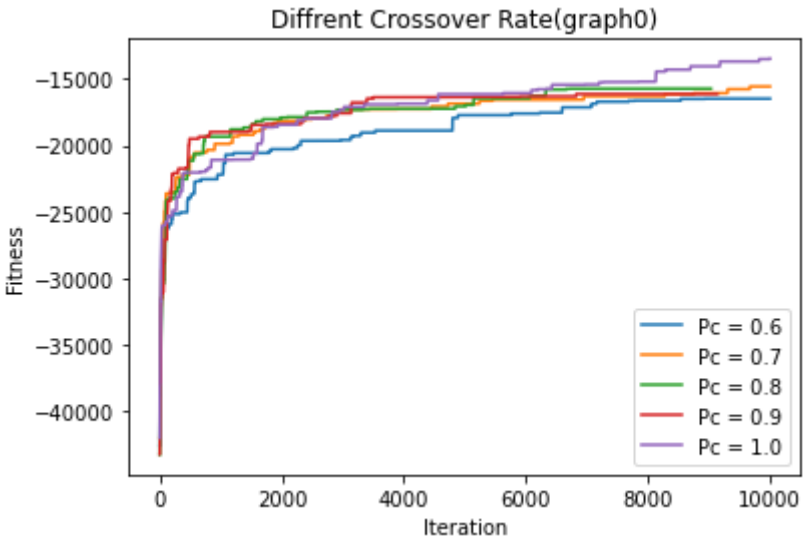
專家參數設定的部分是將三個參數分別變動，也就是變動一個參數，固定其餘兩者，並且變動的幅度是將前面所定義的區間分成五組，因此，總共會有 15 組的參數並分為三個方案實驗，個別找出最好的參數。

Case 1: **變動交配率**，固定突變率、染色體數目

參數設定如(表五)所示，將前面所限制的範圍區間平均分成五塊，結果如(圖十)，當交配率越高時期收斂的結果較佳。

(表五) Case 1 參數設定

參數	**Case 1 - 交配率差異**
交配率	**[0.6, 0.7, 0.8, 0.9, 1.0]**
突變率	0.01
染色體數目	30



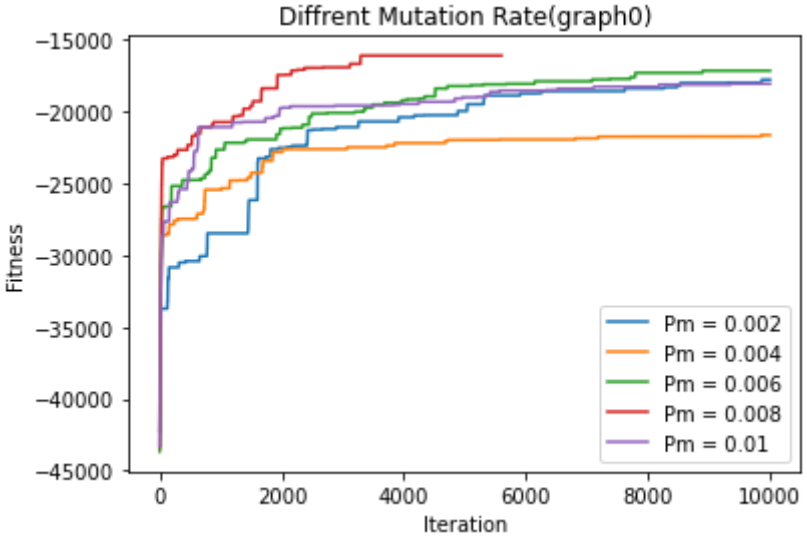
（圖十）交配率差異圖

Case 2: **變動突變率**，固定交配率、染色體數目

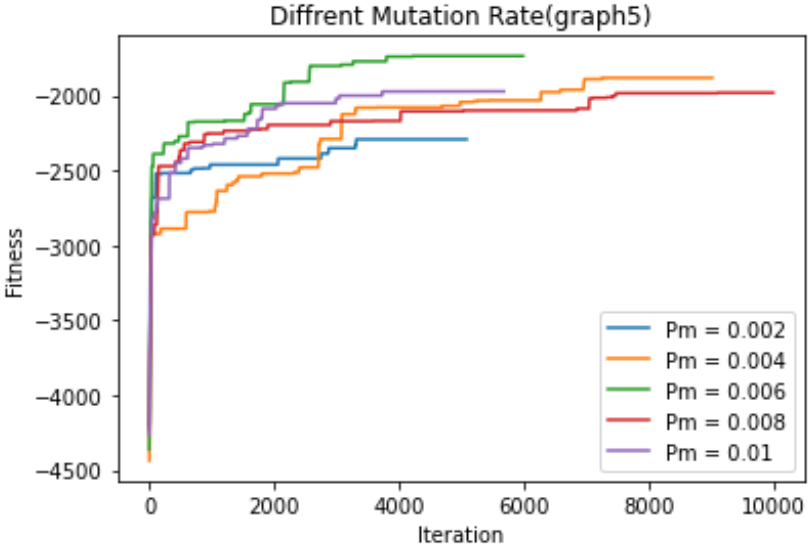
參數設定如(表六)所示，將前面所限制的範圍區間平均分成五塊，結果如(圖十一、圖十二)，在 Graph 0(圖十一)時突變率為 0.008 時不僅收斂結果較佳且收斂速度快；但是在 Graph 5(圖十二)則是 0.006 表現得最好。因此有這種不同 instance 會有不一樣的結果出現時，後續在取最佳參數時就會統計哪一個突變率出現的次數最為頻繁。

(表六) Case 2 參數設定

參數	Case 2 – 突變率差異
交配率	1.0
突變率	**[0.002, 0.004, 0.006, 0.008, 0.01]**
染色體數目	30



（圖十一）Graph 0 突變率差異圖



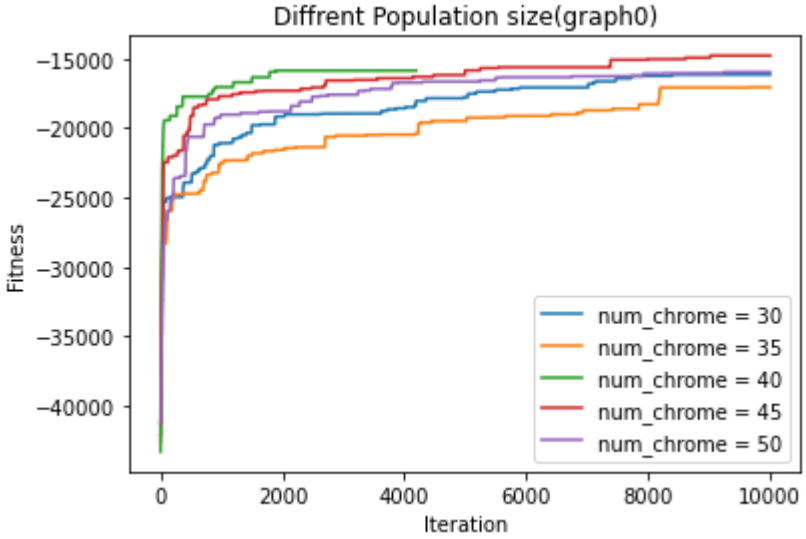
（圖十二）Graph 5 突變率差異圖

Case 3: **變動染色體數目**，固定突變率、交配率

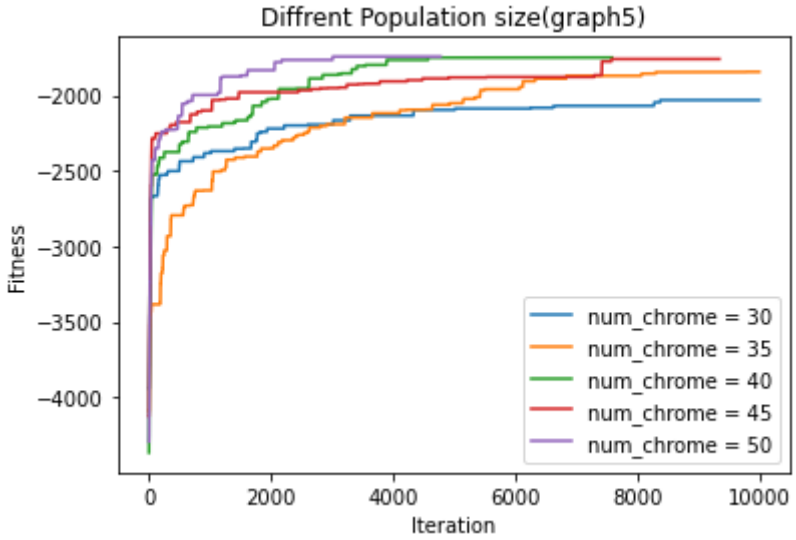
參數設定如(表七)所示，將前面所限制的範圍區間平均分成五塊，結果如(圖十三、圖十四)，與突變率情形相似，在不同的 instance 下，各個參數的表現好壞不一。

(表七) Case 3 參數設定

參數	Case 3 – 染色體數目差異
交配率	1.0
突變率	0.01
染色體數目	**[30, 35, 40, 45, 50]**



（圖十三）Graph 0 染色體數目差異圖



（圖十四）Graph 5 染色體數目差異圖

最後決定參數時，我們將所有圖中表現最好的數值記錄下來，並以最直觀的方式選擇出現次數較多的設為候選值，因此最後得出三組解，如(表八)。

(表八)三組最佳參數組合結果

交配率	突變率	染色體數目	最佳適應值
0.7	0.008	40	4373
0.7	0.006	40	4959
0.8	0.008	35	4561

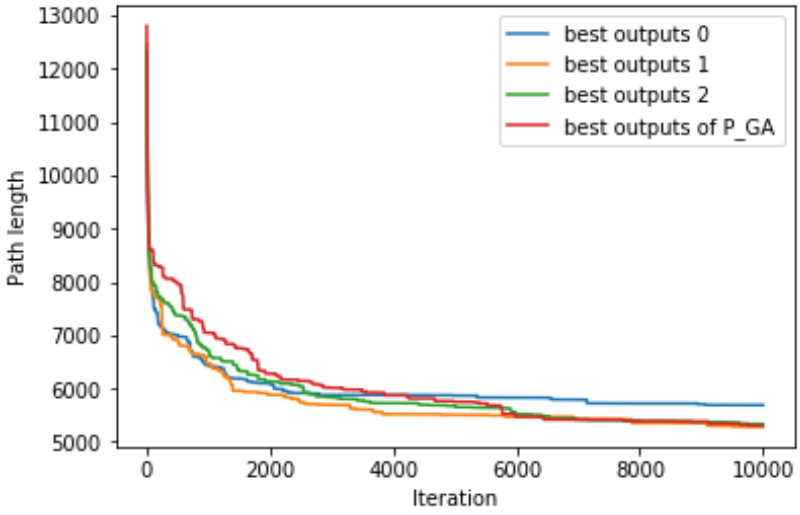
5.2 結果比較

最後將 Parent GA 在 iteration 20 的結果與專家設定的結果做比較，可以看得出來 Parent GA 設定的參數都比專家設定的結果來的好(圖十五)且收斂速度也較快(表九)。

在 Parent GA 尋找最佳參數時，大約花費了 2 小時 16 分；專家設定則是花了 18 分鐘跑完，但加上後續分析的時間，大約也是耗費了 2 小時左右，因此在時間上並沒相差太多。不過，由於我們區間分割的設定只有 5 個，若是將區間分割得更細小，則耗費的時間也會隨之增加，而此時 Parent GA 的優勢就顯現出來了。

(表九)Parent GA 與專家設定參數比較

Parent GA 設定		專家設定	
染色體編號	收斂所需迭代數	組合編號	收斂所需迭代數
1	2802	1	4373
2	3460	2	4959
3	3462	3	4561



(圖十五) Parent GA 與專家設定參數之結果表現

六、 結論

在本研究中所使用的 Parent GA 跟傳統的專家手動設定相比，其結果表現及收斂速度都較好。不過有可能是本研究中的專家設定沒有對各個參數的關聯性去做研究，這部分可後續接下去研究。

另外，此次研究只有考慮到三個參數，未來可以再將其他參數加進來以優化整個演算法，或者將 Parent GA 的參數優化應用至其他演算法或是機器學習的參數優化。

最後，基因演算法雖然在參數的設計上會影響收斂的快慢與好壞，但是染色體的 Encoding 及 Decoding 和 Fit function 設計也對問題的結果有相當大的影響，因此也能多加著墨在此部分。

七、 參考資料

1. Abid Hussain, Yousaf Shad Muhammad, M. Nauman Sajid, Ijaz Hussain, Alaa Mohamd Shoukry and Showkat Gani. (2017, October 25). Genetic Algorithm for Traveling Salesman Problem with Modified Cycle Crossover Operator. Computational Intelligence and Neuroscience, Article ID 7430125.
2. Avni Rexhepi, Adnan Maxhuni and Agni Dika. (2013, January). Analysis of the impact of parameters values on the Genetic Algorithm for TSP
3. TSPLIB: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/
4. 國立陽明交通大學 109 學年度 1491 號課程 「基因演算法與管理科學應用」上課教材”GA05 基因演算法 Part 2”
5. 國立陽明交通大學 109 學年度 1491 號課程 「基因演算法與管理科學應用」上課教材”GA07 排列解的編碼與 Traveling Salesman Problem (TSP)”