東海大學資訊工程學系研究所
碩士論文

指導教授：林祝興博士

Dr. Chu-Hsing Lin

基因演算法在 Matlab 單一程式多工處理之設計與分析

Design and Analysis of Genetic Algorithm on Single Program/Multiple Data for Matlab

研究生：姚信任

中　華　民　國　一　○　四　年　七　月

東海大學碩士學位論文考試審定書

東海大學資訊工程學系 研究所

研究生 姚 信 任 所提之論文

基因演算法在MATLAB單一程式/多工處理之設計與分析

經本委員會審查，符合碩士學位論文標準。

學位考試委員會
召 集 人 _____ 簽章

委 員 _____

_____

_____

指 導 教 授 _____ 簽章

中華民國 104 年 6 月 24 日

中文摘要

本文目的在探討基因演算法在矩陣實驗室的單一程式/多工處理之實作設計與效能分析，我們發現在使用矩陣實驗室實現平行運算時，可能會造成一些問題，並提出解決方案。我們以旅行者問題為基礎設計程式，採用平行運算改良程式，將基因演算法程式迴圈平行化，初始族群分割到多個工作區域同時運算，以提高程式運算速度。此外，此方法也使得程式不易收斂到局部最佳解，所產生平行路徑可以對初始族群做重複修正。當初始族群總數和演化代數總數相同時，我們發現平行基因演算法具有更高效能並找到更優化的解答。

關鍵詞：旅行者問題，基因演算法，染色體編碼，平行計算，單一程式/多工處理

ABSTRACT

In this thesis, we aim to design and analyze the genetic algorithm implemented on single program/multiple data for Matlab. Some experiments based on the Travelling Salesman Problem were conducted and parallel program were developed for it. For-loop programs are paralleled, initial population was split to multiple work area for speed-up computation. Additional advantages were such as local optimization could be avoided, and initial population can be adjusted repeatedly by using each parallel path generated. When the number of initial population and the generation number of the final convergence are the same, we found that the parallel genetic algorithm has higher performance and better solution.

Keywords: Travelling salesman problem, genetic algorithms, chromosome coding, parallel computing, single program/multiple data

誌 謝

時光匆匆，漫長的研究生活隨著論文的完成也即將告一個段落。回想起這段日子的磨練和學習，雖然過程十分辛苦卻也獲益良多。現在，完成最後的謝誌就要到分離的時候，濃濃的不捨和深深的感謝全部湧上心頭。

首先要感謝的是我的指導教授林祝興老師，由於他的諄諄教誨，以及不厭其煩的給予鼓勵與支持，我才能在一次次的難關中支持下去，獲得知識與專心致志的學習；感謝詹進科老師、江憲坤老師、劉榮春老師、陳雍宗老師在口試時提供許多寶貴的建議，不僅讓我對學術研究上有更多的了解，也使得論文的品質更臻完善；感謝所上的楊朝棟、朱正忠、黃育仁與石志雄老師在課內外的悉心教導，讓我得到豐富的學習經歷。

此外，要感謝實驗室中的學習夥伴們，謝謝傑立與正傑在我研究過程中提出的諸多建議和學習上的指導，栢葦，祥霖，思縈在我求學過程中的互相幫助、支持與打氣，看著你們不停努力奮鬥的樣子，讓我在困擾與疲倦的時候，支持我持續的思考並且耐心研究。最後，感謝我的父親與母親，既使在我學習過程中的低潮期，仍然不時鼓舞著我，提醒我不要輕言放棄。這些求學時的點點滴滴都會成為解決問題的鑰匙，對我的人生有所幫助。

僅將此份感謝獻給所有的您們，謝謝!

姚信任 謹致

2015/7/28 於東海大學資工所

目錄

圖目錄

表目錄

第一章 簡介

在本章節我們會簡單陳述旅行推銷員問題的由來和基因演算法的基本架構，並藉由旅行推銷員問題來探討改良式平行基因演算法的設置和成效。

1.1. 研究背景

旅行推銷員問題(Travelling Salesman Problem)[14]最早是由 Hassler Whitney 提出的，是指一名推銷員要拜訪多個地點時，想找到一條只拜訪每個地點一次後再回到起點的最短路徑，是一個 NP 完全問題(NP-Complete problem)[18]，如果有 n 個地點的話，其答案的可能組合數為$(n-1)!/2$。雖然看起來很簡單，但在節點數目增多後求解卻極為複雜。目前常用來解決此問題的方法有基因演算法(Genetic Algorithm)[2][10] [17] [19]、螞蟻系統(Ant System) [4] [13]、粒子群[7]或各種改良方法來求解。這些演算法利用各種不同的演化方式，試圖在演化過程中找到最佳解。

基因演算法 (或稱遺傳演算法)是模仿自然界生物進化機制發展起來的隨機全域搜索和優化方法，它參考了達爾文的進化論和孟德爾的遺傳學說。其優點是能在搜索過程中自動獲取和累積有關搜索空間的知識，並求得最優解。

1.2 研究目的

在求解旅行推銷員問題時，通常透過兩種方法來執行，一種是利用傳統啟發式方法透過對路徑的交換來逐漸逼近最佳解，然而當節點規模變大時，很容易陷入區域最佳解(local optimal solution)，所以現在大多數研究旅行推銷員問題的方法改用巨集啟發式的方法，如前面所談到的基因演算法、螞蟻系統、粒子群、模擬退火法...等方法，也因為這些改良的方法比起傳統的交換法來得效果卓越，逐漸成為主流的研究方法。

本論文探討的是由於基因演算法容易在節點數過大時，可能過早陷入局部最佳解；因此為了改善基因演算法執行效果的問題，本論文在傳統基因演算法(GA)的架構下，搭配 SPMD 來分割族群和分割演化代數的方式做平行計算，希望可以對解題有更好的成效。

1.3 研究架構和流程

本文共分為五章，架構安排如圖 1 所示:

第一章:簡介，說明研究背與文章架構。

第二章:相關文獻探討，介紹 TSP 問題和相關研究。

第三章:研究方法與實驗環境，介紹實驗環境與平行基因演算法程式。

第四章:研究結果與分析，將提出的成果進行比較。

第五章:結論與未來方向，針對實驗的結果進行總結。

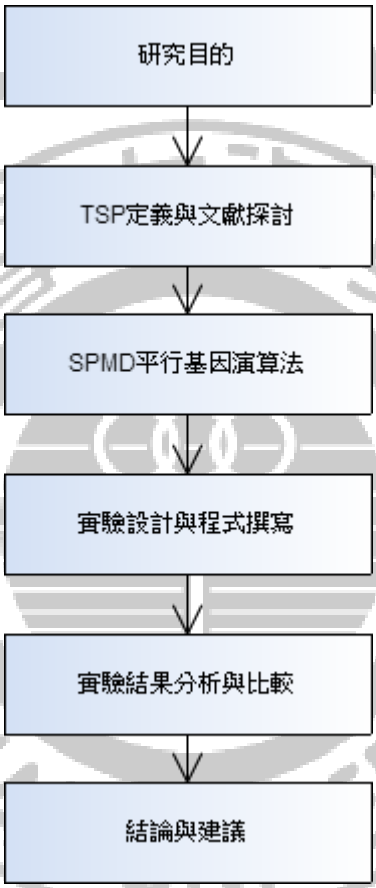

圖 1　研究架構圖

第二章 相關文獻探討

2.1. 旅行推銷員問題

“旅行推銷員問題” (Travelling Salesman Problem)常被稱為“旅行商問題”，是一個廣泛被研究的組合最佳化問題(Combinational Optimization Problem) [20]，指一名推銷員要拜訪多個地點時，找到一條只拜訪每個地點一次後再回到起點的最短路徑。雖然看起來很簡單，但在節點數目增多後求解卻極為複雜。以 40 個地點為例，如果要列舉所有路徑後再確定最佳行程，那麼總路徑數量之大，幾乎難以計算出來。

“旅行推銷員問題”可以應用領域包括：1.如何規劃最合理高效的道路交通，以減少擁塞[11] [16]；2.如何規劃物流，以減少實體營運成本；3.在網路環境中如何更好地設置節點，使得資訊流暢等；4.司機輪班的方式[5]；5.關於行車導航的改良研究[6] [8] [12]。

2.1.1 旅行推銷員問題的函式規劃和範例

若一個旅行商要到各個城市販賣商品，但每一個城市都只能拜訪一次，而且最後一定要回到最初拜訪的城市，旅行推銷員希望可以找到一條最短的路線，由於沒有指定初始造訪的城市，所以如果有 n 個城市需要拜訪，則方法數總共為 (n-1)!/2 種，當城鎮數量足夠多的時候，旅行推銷員問題的解答很難在短時間中計算出來。

旅行推銷員問題的函式如(2-1)所示，在 n 個城市中路徑排序中可以找到最短路徑 $\alpha_{\min}$，使其總路徑和最小。

$$f(\alpha_{\min}) = \min_{\alpha_i \in S} f(\alpha_i), i = 1 \dots n \qquad (2-1)$$

其中$\alpha_{\min}$代表最短路徑，α_i代表城市，S 代表所有城市的集合，f 代表旅行商經過城鎮的順序。

圖 2 是個旅行商問題的示意圖，$\alpha_i = (x_i, y_i)$表示座標平面上的位置，對應的坐標點如表 1，從圖中我們可以找出一條路徑把所有點連接起來且路徑長度最短，此時我們就稱連線順序方式(2-2)為此旅行者問題的最短路徑，其路徑總長度就是最短路徑長度。

$$f(\alpha_{\min}) = (\alpha_1, \alpha_9, \alpha_8, \alpha_{10}, \alpha_2, \alpha_7, \alpha_3, \alpha_4, \alpha_5, \alpha_6) \qquad (2-2)$$

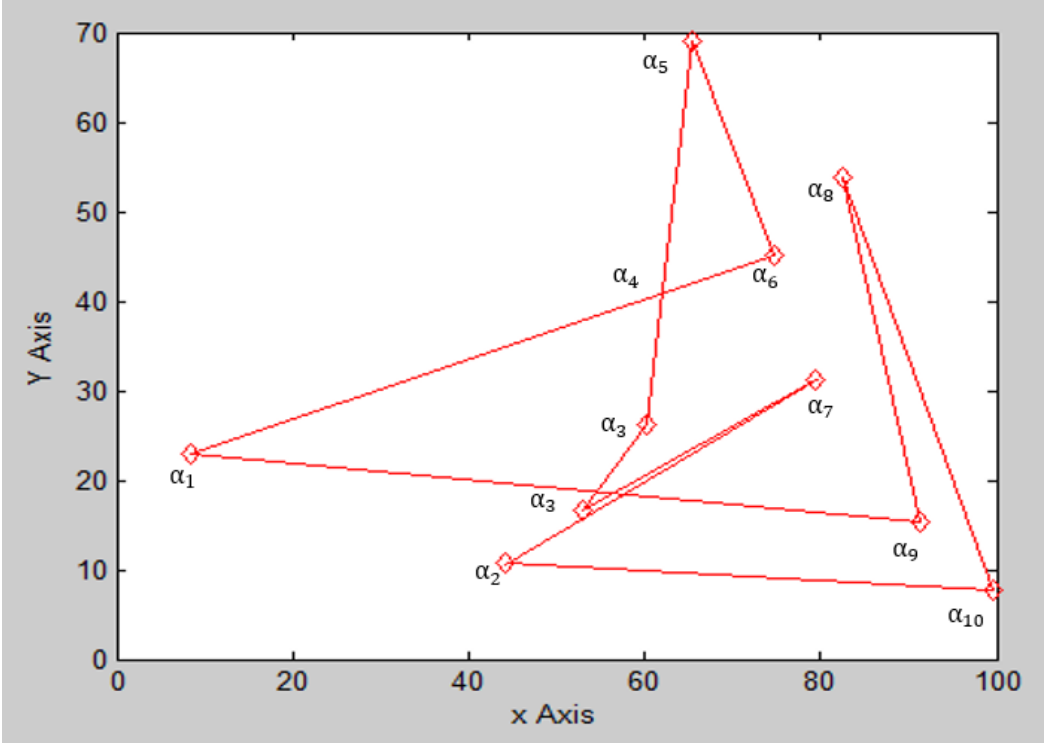

圖 2 旅行者問題示意圖

表 1.旅行者問題中隨機點的(x,y)位置

α_i	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9	α_{10}
x_i	8	42	55	61	64	76	80	83	91	100
y_i	23	11	15	25	70	42	31	54	17	8

2.2. 基因演算法

基因演算法(genetic algorithm)使用適者生存的原則，是模仿自然界生物進化機制發展起來的隨機全域搜索和優化方法，它參考了達爾文的進化論和孟德爾的遺傳學說並結合電腦科技所產生出來的。基因演算法於 1962 年由 John Holland 所提出，在 1968 年完成，其演算法的流程圖如圖 3 所示，是把已知的解決方案以族群表示並利用選擇、交配、突變、菁英政策等方法進行變化，然後從中挑出最佳方法，其優點是能在搜索過程中自動獲取和積累有關搜索空間的知識，並求得其中的全域最佳解(global optimal solution)，而缺點是當題目複雜時有可能只找到區域最佳解(local optimal solution)，如圖 4 所示，其中 B 點是全域最佳解，而其他的 3 點都只是區域最佳解。

圖 3 為基因演算法的流程圖，首先設定染色體個數、演化代數、交配率、突變率...等參數，然後產生一組初始族群，再進行評估適應值，計算每條染色體的適應值，如果適應值滿足條件就結束，條件沒有滿足就從初始染色體中挑出較優秀的父代並交配出子代，在一定機率下產生突變，並保留較優秀的染色體，彙集成新族群，並重新做評估適應值，直到終止條件滿足。

為了後續平行基因算法的說明，我們稱基因演算法中產生初始族群這部分程式為 GA1，其餘部分程式為 GA2，而 GA2 是預定做單一程式多工處理（SPMD）的區間。

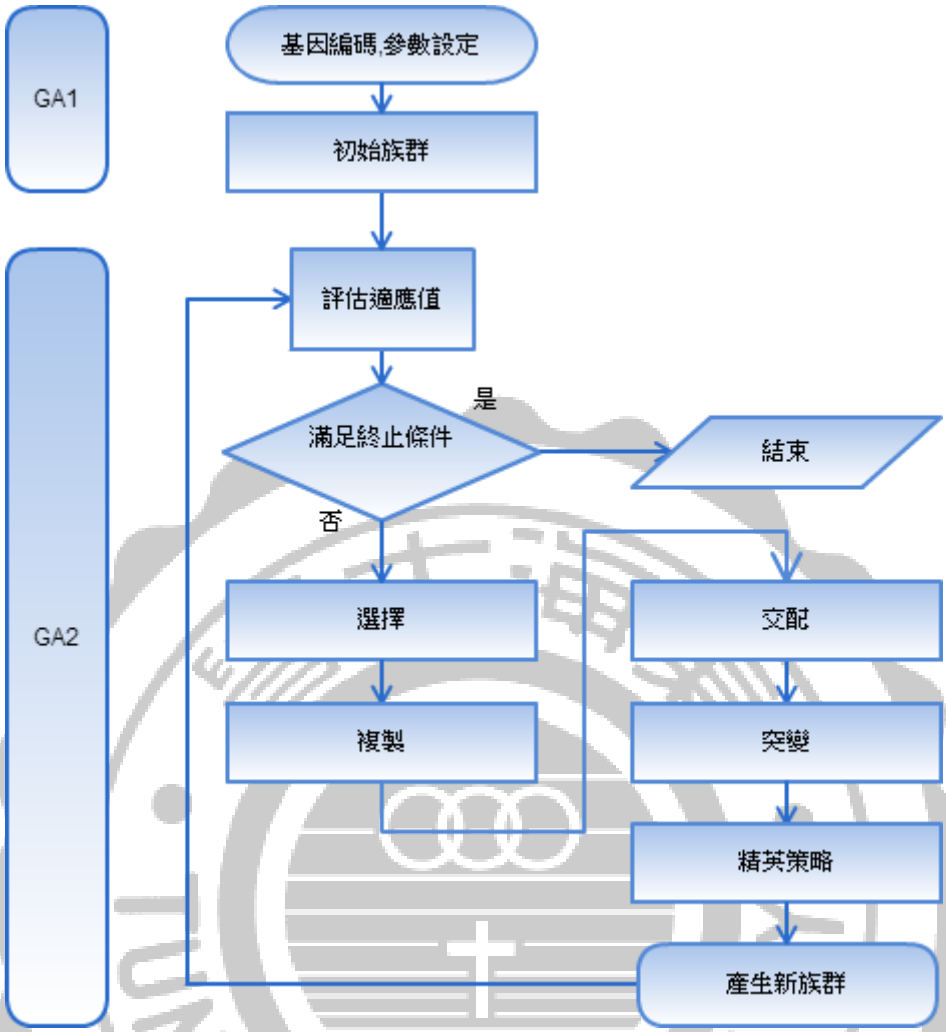

圖 3 基因演算法流程圖

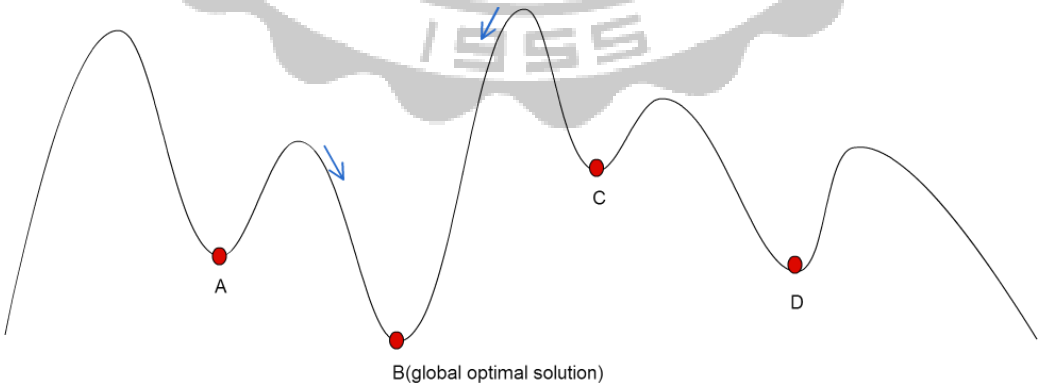

圖 4 全域最佳解

(1) 基因編碼(gene coding)

基因演算法在運算之前需要對問題進行編碼，讓問題改成用染色體(chromosome)表示，圖 5 是以二進制編碼的染色體，染色體中有 7 個基因(gene)，每個基因用 0 或者 1 來表示。或者可以使用實數制方式來表示染色體，本文使用基因表示節點，所以基因都設為有順序且不重複的正整數，如圖 6 所示，使用好的編碼方式可以更容易解題與分析結果，本文中使用實數編碼與 Grefenstette 編碼來求解旅行商問題。

圖 5 基因演算法二進制編碼

圖 6 基因演算法實數編碼

(2)產生初始族群(generate initial population)

一般基因演算法都是使用隨機產生的方式產生初始族群，有時也會因為問題的限制而對初始族群產生一些限制，例如題目要求基因只有偶數時，就會限制基因不產生奇數。

(3)評估適應值(evaluate fitness)

把問題轉為基因編碼的形式之後，要設計一個適應值函數(fitness function)，來分辨染色體的優劣，藉由評估適應值，可以幫助基因演算法辨別哪些是優秀該保留的染色體，哪些是需要去除的，一般而言，經過評估適應值產生的數值越大其解答(染色體)越好。

(4)Grefenstette 編碼

為了解決在交配和突變做基因交換時產生的染色體錯誤問題，本文使用Grefenstette[15]編碼方式(encoding)來對染色體重新編碼，其用法為每排中所選節點在未選節點中的位置，因為每排的數字代表節點的位置，如果前面有出現比自己還小的節點，則按順序減去那些節點的個數來形成新的字串，Grefenstette 編碼會在適應值評估後執行並且在精英策略結束之後才會進行解碼。

(5)選擇(selection)

主要觀念為「留下好的染色體，排除不好的染色體」，如果適應函數較差的話，表示這是一條較長的路線，很難找到最佳解，所以就使用較優秀的染色體取代較差的染色體，主要有兩種方法，第一種是由 Wetzel 於 1983 年提出的輪盤選擇法，如圖 7 所示。輪盤選擇法的選擇方式是越優秀的染色體越容易保留，較差的染色體就去除，藉由去蕪存菁的方式，讓擁有較好基因的染色體當父代，如圖 7 所示，由於染色體 A 較優良，有 40%機會被選上，染色體 D 就只有 15%機率被選上。

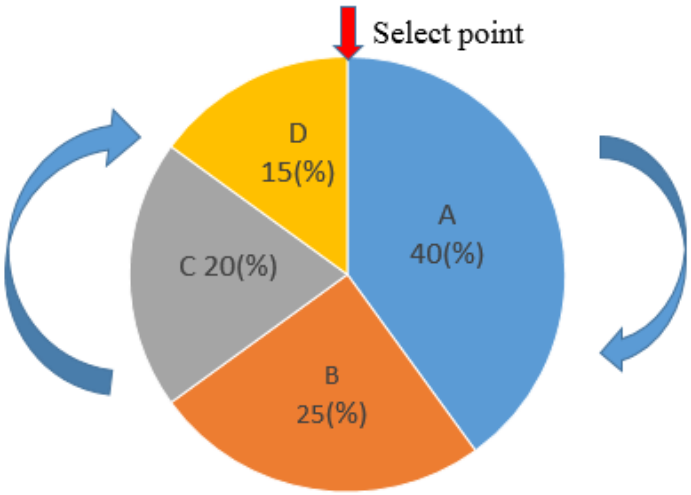

圖 7 輪盤選擇法

第二種是由 Goldbug 在 1989 年提出的競賽選擇法，從群體中隨機選出若干條染色體，然後再從其中挑出最好的染色體做為父代染色體，為後面的演化做準備，如圖 8 所示，隨機選擇 5 條染色體 A,B,C,D,E，從中挑選出最好的一條 E，同樣的從另外 5 條染色體 F,G,*H*,I,J，從中挑選出最好的一條 H。

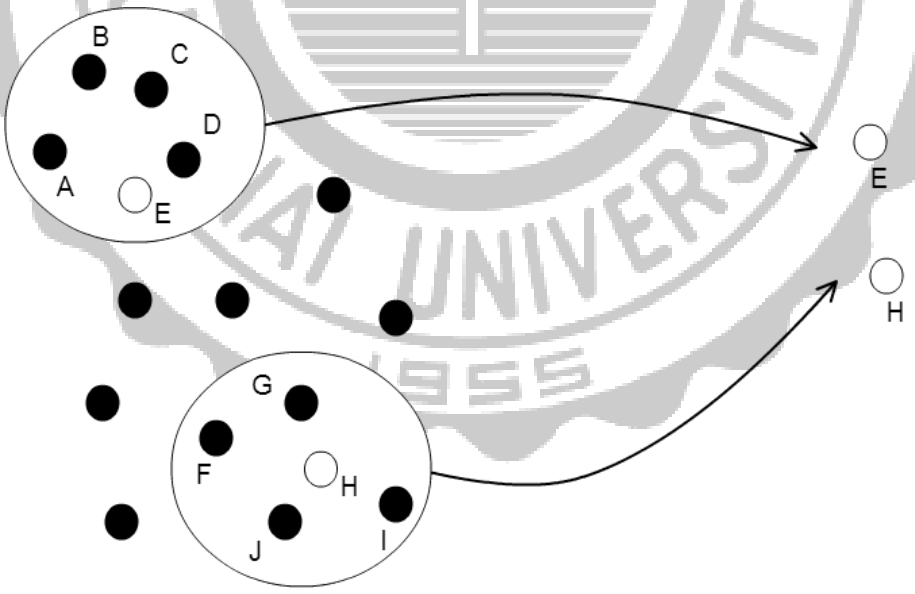

圖 8 競賽選擇法

(6)交配(crossover)

交配的目的是使得子代染色體繼承父代染色體的優點，希望子代染色體能有比父代染色體更好的表現(適應值越大)，所以該如何選擇交配的方法和交配的次數是交配的關鍵，圖 9 中的單點交配是較常使用的交配方法，隨機選擇某個基因片段，例如挑選第一條染色體與第二條染色體從第 3 個基因之後的染色體片段，然後兩個染色體基因片段後的基因都對調。

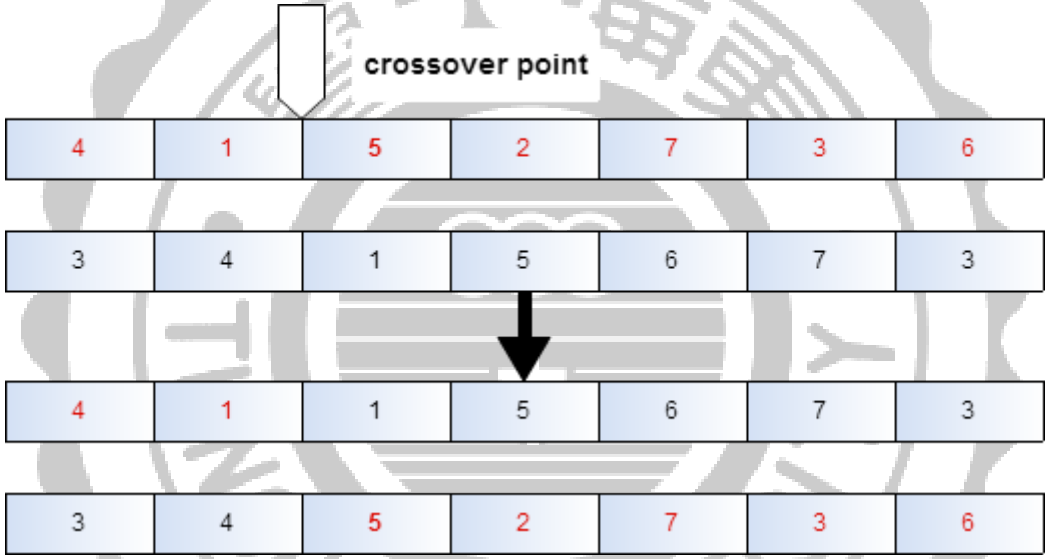

圖 9 單點交配

(7)突變(mutation)

如果只有做交配來改變染色體，染色體的變化就會非常少，所以要利用突變跳脫區域最佳解，通常突變的方法有 3 種，圖 10 是隨機突變(random mutation)隨機挑選突變點來改變染色體；圖 11 是互換突變(reciprocal exchange mutation)以互換基因的方式來改變染色體，從染色體中隨機挑出 2 與 6 這兩個基因交換來產生新的染色體；圖 12 是反轉突變(inversion mutation)隨機挑選基因片段並且改

用反轉放回的方式來改變染色體，圖中要換的基因片段為 2, 7, 3, 6 其中最後的基因移到第一個位子，而倒數第二的基因移到第二個位子，依此類推，產生新的基因片段 6, 3, 7, 2。

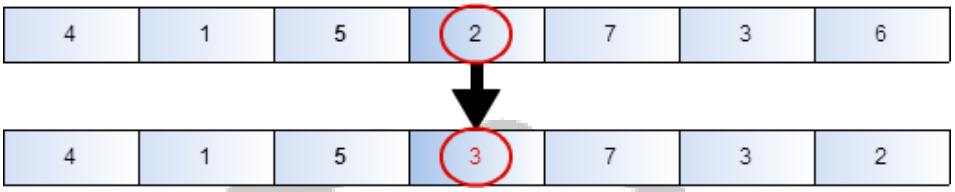

圖 10 隨機突變

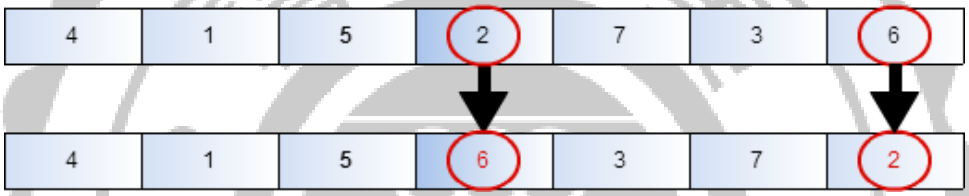

圖 11 互換突變

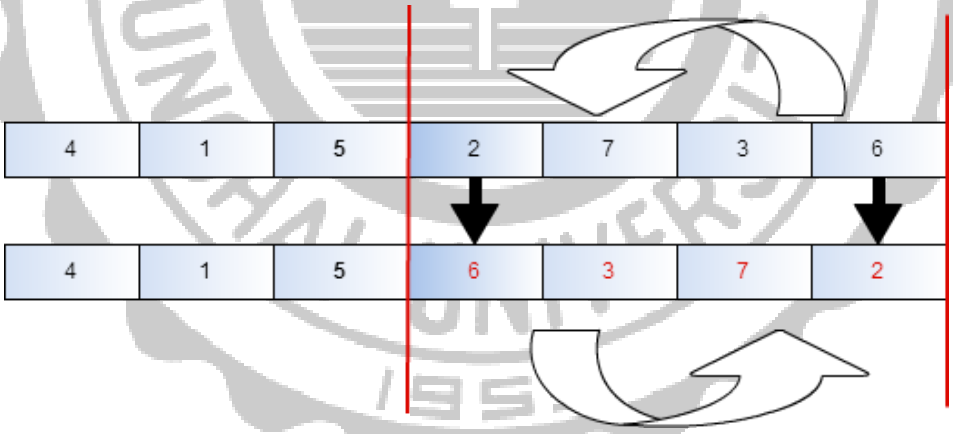

圖 12 反轉突變

(8)菁英策略(elite policy)

本文採用的菁英策略(elite policy)是先複製父代染色體中的最佳染色體，每當突變結束後，就用最佳染色體取代之把子代族群中最差的染色體取代之，詳細敘述會在後續章節中提到。

2.3 單一程式多工處理

單一程式多工處理（Single Program/Multiple Data, SPMD）指的是複製單一程式到多個工作區域中平行處理，如果想要在 Matlab 的中操作平行處理，要先安裝 Matlab 的平行處理工具箱(Parallel Computing Toolbox)才可以開始運作，本次實驗使用單一程式多工處理(SPMD)來執行平行基因演算法，利用平行計算來加快得到結果的速度。

單一程式多工處理說明:

單一程式(single program)指的是使用者可以在 Matlab 中事先創造多個工作區域(lab)，然後設定要平行運算的單一程式，系統會把它複製並且分配給每個工作區域，然後做平行運算。

多工處理(Multiple data)指的是雖然我們使用相同的單一程式在運算，但實際上每個工作區域(lab) 運算的過程和結果都是各自獨立的，而當這些平行運算的程序運算完畢，我們可以收集各個不同的結果。

由上述章節所述，本次實驗選擇 GA2 這塊程式改為單一程式多工資料來平行運算，如圖 13 所示，所以在做平行基因演算法時，是把一般基因演算法中除了初始族群生成以外的部分全部都放到 SPMD 中去做平行處理。

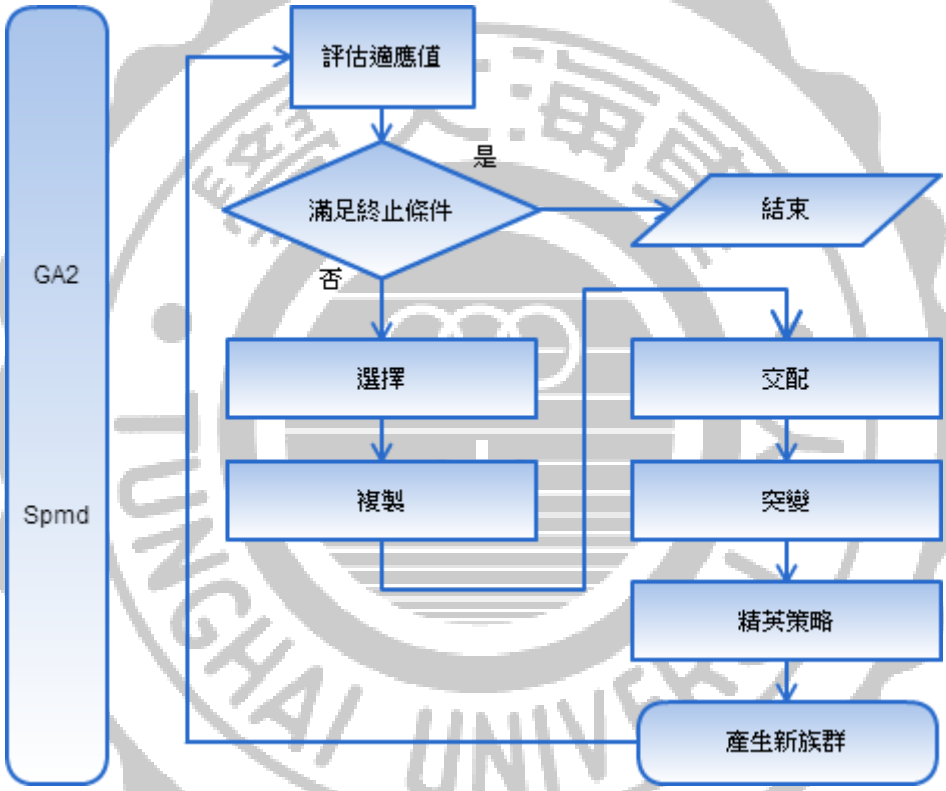

圖 13 單一程式多工資料

2.4. 平行基因演算法

平行基因演算法[9]的設置方法有許多種，本文是利用 SPMD 來達成平行計算，首先是先產生一組共用的初始族群，再生成多個工作區(lab)，同時在每個工作區(lab)運算基因演算法，如果在工作區中沒有得到最佳解，會把所有工作區的區域最佳解匯集起來，混合成新族群並重新投入運算直到條件滿足為止，整體流程如圖 14 所示。

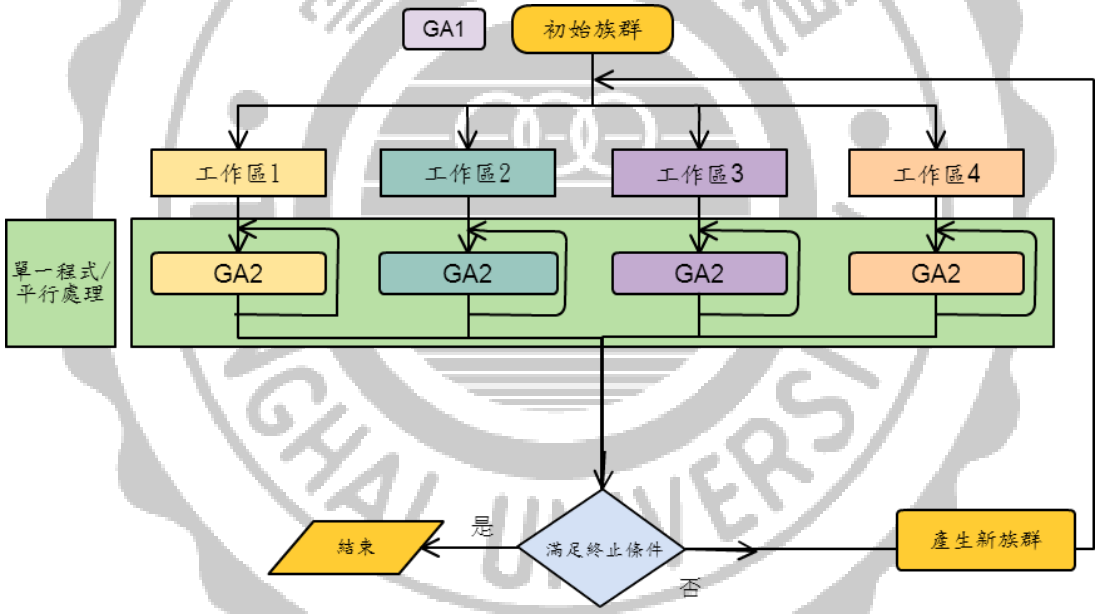

圖 14 平行基因演算法概略流程圖

第三章研究方法與實驗環境

本章節將會介紹本研究提出的方法:結合基因演算法和 SPMD 平行處理來探討對稱路徑長度的旅行商問題。本論文將初始族群分割到獨立環境中，各別進行基因演算法，然後再挑選較好的染色體混合編成新族群重新分配到獨立環境中運算，詳細內容會在下面各章節說明。

3.1. 實驗環境介紹

本實驗使用 Matlab R2013a 操作，電腦系統為 Windows7 64 位元作業系統，記憶體為 4GB，核心數為 4，使用處理器為 Intel(R) 2QuadCPUQ8200 @2.33Ghz，以 PC1 稱之。

本實驗在 4.4 增加使用 Matlab R2013a 操作，電腦系統為 Windows7 64 位元作業系統，記憶體為 8GB，核心數為 2，使用處理器為 Intel i3-3110M CPU@2.40Ghz 以 PC2 稱之。

3.2. 平行基因演算法介紹

平行基因演算法一開始是採用族群分割到不同工作區(lab)的方式，利用工作區的獨立性來保持族群的多樣性來解決過早陷入局部最佳解的問題，但是單純的切割族群只是避免陷入局部最佳解而已，並無法穩定得到較優秀的解，所以增加了切割演化代數來增加混合族群的次數，根據以下公式分割族群和演化代數:

初始族群=平行初始族群×工作區個數 (3-1)

演化代數=平行演化代數×混合次數 (3-2)

其中工作區個數表示同時進行基因演算法的個數，混合次數則是混合不同工作區染色體的次數。

本次實驗的平行基因演算法是把基因演算法中產生初始族群以外的部分複製並分配到多核心工作區域（lab）進行單一程式多工處理 (SPMD)來平行處理。從圖 15 中，在 GA1 和 GA2 為原本的基因演算法，在其中多加了工作區域的分割，以及把不同工作區產生的染色體混合成新族群；工作區域的分割數量，由電腦本身的 CPU 個數決定上限，由於本次實驗主要使用 4 核心電腦，所以把工作區域數目設定為 4。在 4 個工作區域之中，雖然單一程式是執行相同的程式，不過由於多工處理，所以每個工作區域的結果都會不同。每當達到平行演化代數的計算次數或者找到全域最佳解時，滿足第一個終止條件，SPMD 會宣告結束執行；

如果沒有找到全域最佳解，我們收集優秀的染色體來產生新族群，並重新計算直到第二個終止條件找到全域最佳解或者混合次數滿足為止。

由圖 16 提出平行基因演算法和一般基因演算法的比較，假設初始族群的總數為 80，演化代數總數為 4000，一般基因演算法就如同右圖，投入初始族群總數 80 進行 4000 次的演化，試著找出最佳解。而平行基因演算法則是設定混合次數為 2，平行演算代數為 2000，初始族群中的染色體數量設定為 20，因為有 4 個工作區，所以初始族群總數同樣為 80，首先每個工作區各自投入初始族群數 20 並進行 2000 次的演化，得出各個工作區的區域最佳解，然後把全部區域最佳解混合產生新族群，重新進行 2000 次的演化，我們認為用平行基因演算法，可以比一般基因演算法更容易找出較佳的解。

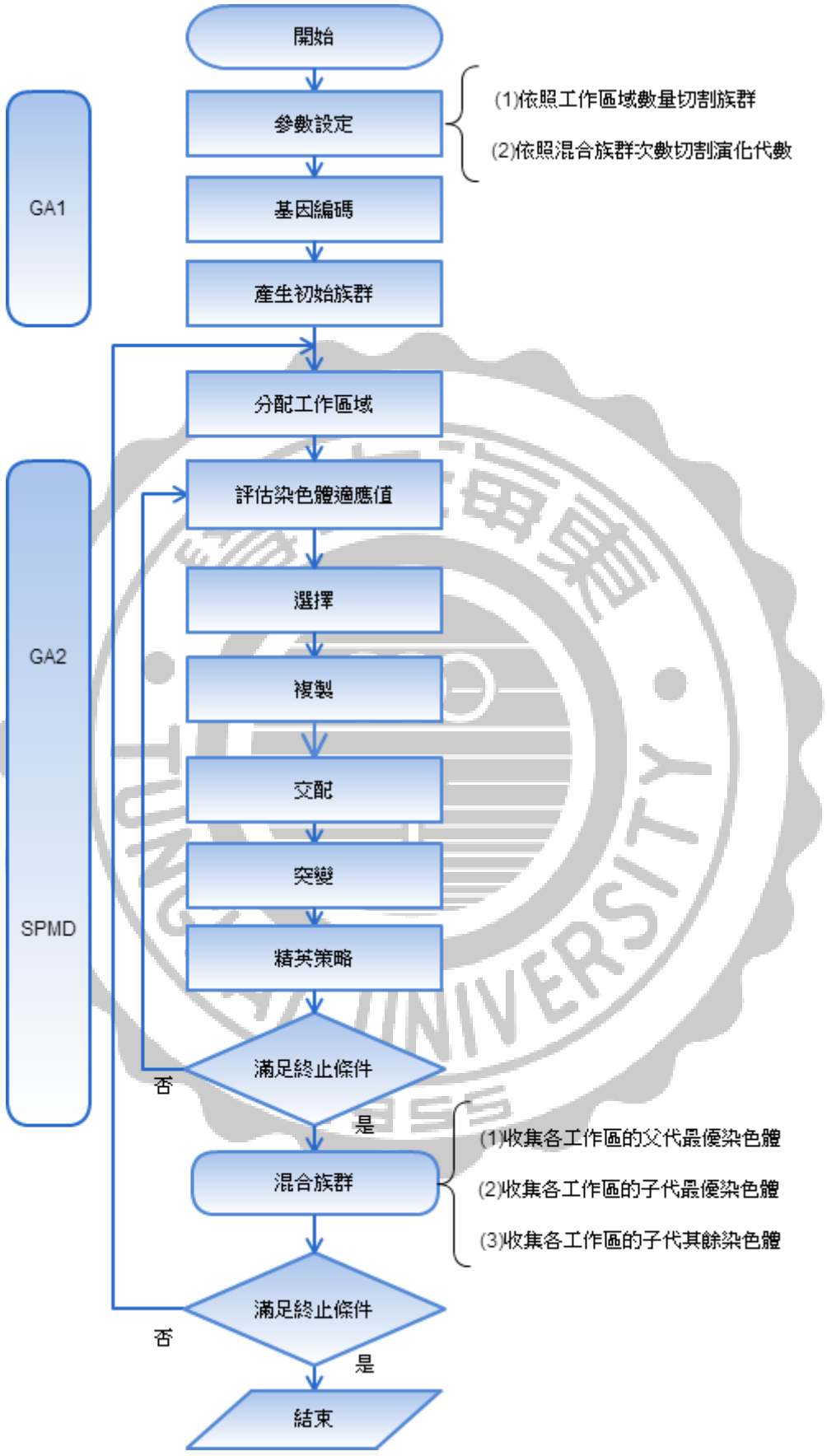

圖 15 平行基因演算法流程圖

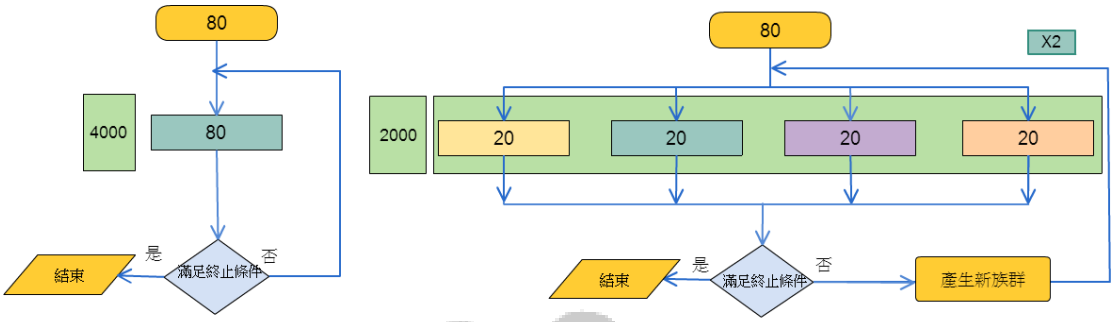

圖 16 基因演算法與平行基因演算法比較圖

3.2.1 基因編碼

為了解決旅行商問題，基因編碼方式如圖 17 所示，一條染色體代表一條完整的旅行商問題路徑，染色體上的基因代表城市的編號，圖 17 的路徑表示從編號 7 的城市開始，行走順序為 7→5→4→9→3→6→1→2→8 最後再回到城市 7，形成一組旅行商問題的解。

圖 17 旅行商問題基因編碼

3.2.2 初始族群

在本研究中，一般基因演算法並沒有對初始族群做特別優化，採取常用的亂數產生，但是平行基因演算法因為增加了混合次數的設置，所以會在過程中產生新族群。

3.2.3 適應函數

適應函數的公式如同定義(3-3)，p_i表示第 i 條染色體的基因順序，F(p_i)表示第 i 條染色體的適應值，就是城市間路徑長度總合的反比，其中$d(k_i, k_{i+1})$代表第 i 個城市到第 i+1 個城市的路徑長度，而$d(k_n, k_1)$則是第 n 個城市走回起點的長度，因為希望適應值越大其路徑越好，所以使用反比表示。

$$\mathrm{F}(p_i) = 1/[\sum_{i=1}^{n-1} d(k_i, k_{i+1}) + d(k_n, k_1)] \qquad (3-3)$$

3.2.4 編碼

Grefenstette 編碼方式有三個特色，其一是第一個基因的值始終不變，因為它是編碼還原時的對照點;其二是所有轉換後的數字,都是大於或等於 1 的整數;其三就是編碼是由數字扣掉比自己小的排序個數，如表 2 所示。

表 2 編碼方式說明

初始路徑 A	4	1	2	5	3
編碼路徑 A	4	1	1	2	3
初始路徑 B	3	2	5	1	4
編碼路徑 B	3	2	3	1	1

首先初始路徑 A（4 1 2 5 3）：第一個數字 4 不動；第二個數字 1 因為前面的數字中只有一個 4，所以 1 不變；第三個數字是 2，由於在它之前只有 1 這個數字比它小，所以它自己扣掉 1 之後，變成 1。同樣的做法，第四個數字 5，因為前面

比 5 小的數字有 4, 1, 2 這 3 個數字，所以 5 扣除 3 後得到 2，所以 5 變成了 2；最後比 3 小的數字有 1 跟 2， 3 扣除 2 之後可以得到 1，我們就可以得到編碼後的的新路徑 A(4 1 1 2 1)同樣的做法就可以得到新路徑 B(3 2 3 1 1)。

3.2.5 選擇

本研究使用輪盤選擇法來決定要做交配和突變的染色體，以公式(3-4)來決定每個染色體被選擇的機率，其中S_i代表第 i 條染色體被選上的機率，其中分子為個別染色體的適應值，分母為全部染色體的適應值加總，利用輪盤法可以挑選較佳的染色體做交配與突變。

$$S_i = \frac{\mathrm{F}(p_i)}{\sum_{i=1}^{n} \mathrm{F}(p_i)} \qquad (3-4)$$

3.2.6 交配

本論文在交換的部分主要使用單點交配(crossover)來做為改變染色體的方式，每兩組染色體，隨機選擇交配點[3]，再由隨機亂數與交配機率(Pc)來決定是否交配；如果交配則兩組染色體包含交配點與後方的基因片段做對調。然而直接對調會造成重複到訪城市的問題而產生錯誤，如圖 18 所示，所以會在交配前使用 Grefenstette 編碼，如圖 19 所示，因為交配是在選擇之後的染色體組合而成，所以是有方向的產生新的優化染色體。

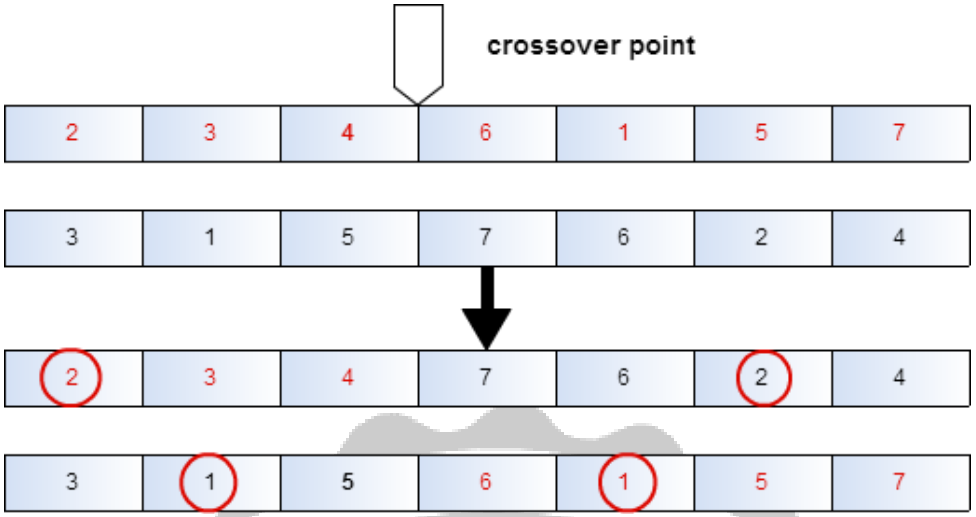

圖 18　Grefenstette 單點交配錯誤

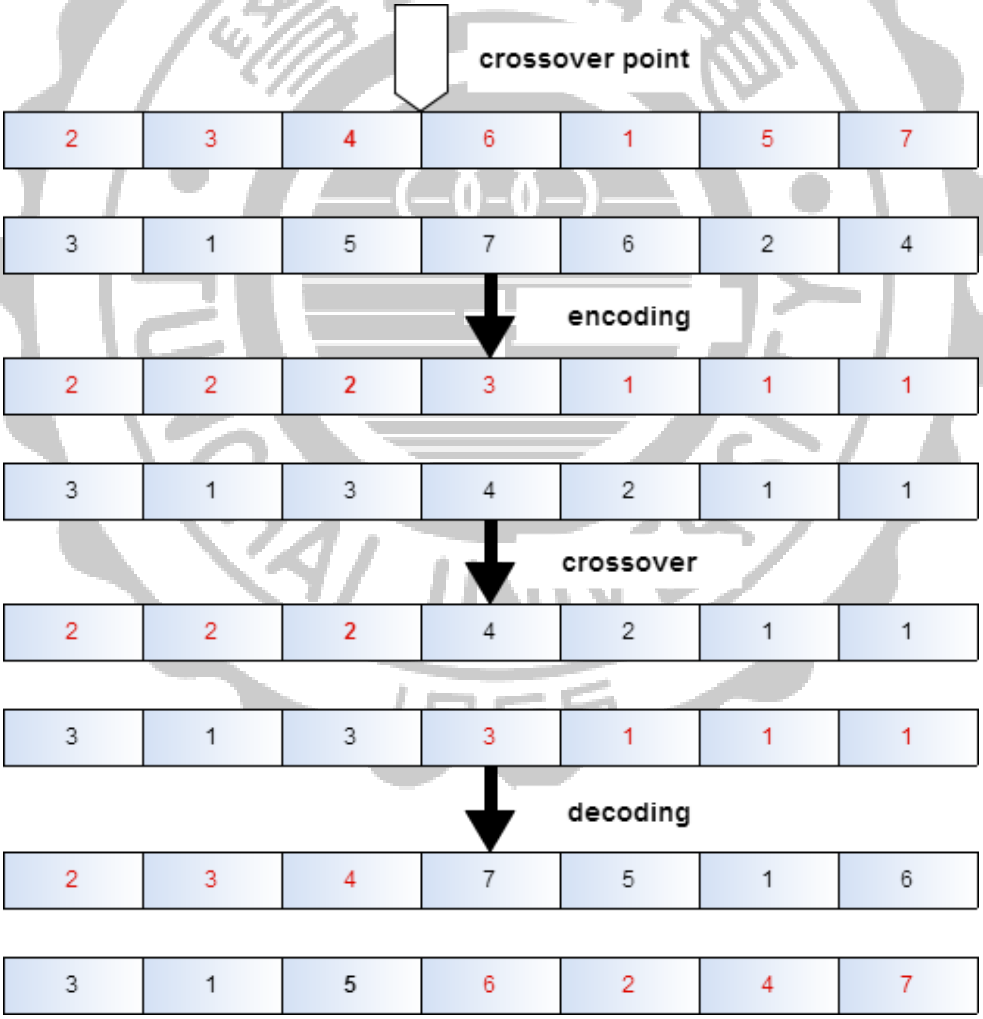

圖 19　Grefenstette 單點交配

3.2.7 突變

本論文中使用隨機突變(random mutation)來改變基因，在圖 20 中展示當突變點選擇第 2 個基因時之情形，突變的轉換方式與圖 19 的交配方式同樣需要先經過 Grefenstette 編碼再轉換，但不同的是，交配一定需要兩條染色體才可以做改變，但是突變只需要一條染色體就可以產生變化。

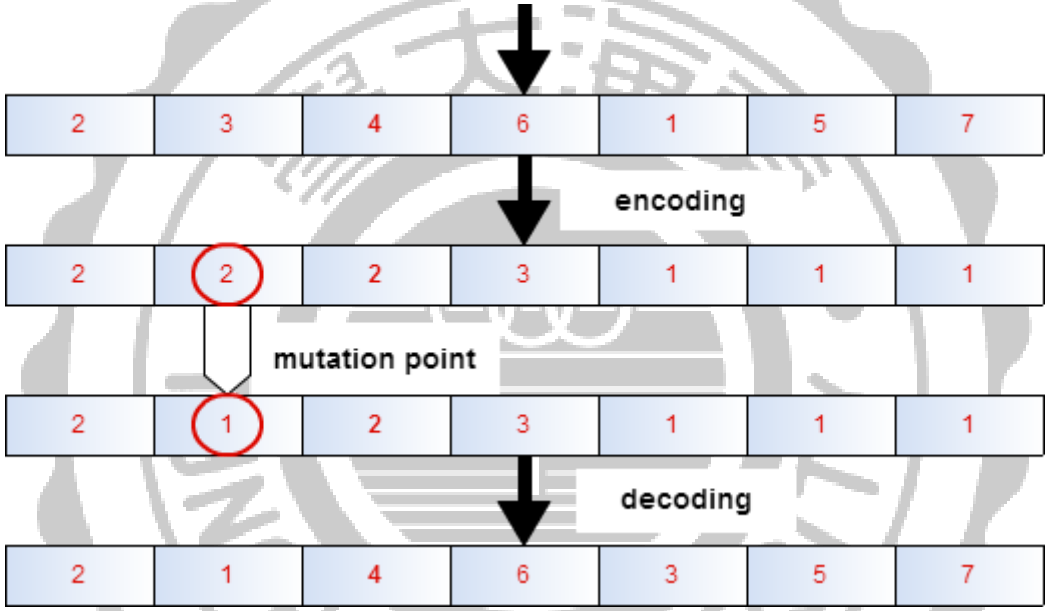

圖 20　Grefenstette 隨機突變

3.2.8 菁英策略

採用菁英策略(elite policy)是為了讓上一代與這一代的最佳染色體，都可保留到下一代繼續演化，使得下一代可以找到更佳的染色體，本研究設定保留上一代最佳的一組染色體與子代中較佳的染色體組合成下一代演化用的染色體群體。

3.2.9 混合族群

每當各工作區域的基因演算法在 SPMD 的平行運算結束，利用各個工作區計算完後產生的優化路徑，取出其中的最佳染色體和其他染色體混合成新的初始族群(mix initial population)，這樣可以使得平行基因演算法不會陷入局部最佳解，混合族群的次數由「混合次數」來決定，可由以下公式來表示:

混合次數=基因演算法的演化代數/平行演算法的演化代數　(3-5)

混合次數代表產生族群的次數，如果混合次數是 4，表示除了產生初始族群的 1 次外，混合了染色體而產生新族群的次數為 3 次。

3.2.10 節點產生器

節點產生器（node　generator）是利用圓的每一點到圓心的距離均相等，所有點的最短路徑總和，即為每一點與自己最接近點的連線長度總和，本次設定圓半徑長度為 1，所以最短路徑長度會隨著節點數的增加逼近 6.28，這樣產生節點，會因為所有節點的最短順序是依序的，方便觀測當節點數大量時的數據結果。

圖 6 是一個利用節點產生器設定 12 個節點數時所產生的圖，其中小圓圈代表示的是每個節點的位置，並且利用極坐標表示法，我們可以設定所有點的位置和每個點之間的路徑長度，由於每個點旁邊的兩點即為最接近自己的點，所以當節點數量為 12 時的最短路徑如同(3-6)，且可以快速計算出最短路徑長度，跟節點數只有 4 個時相比(3-7)，可以發現當節點數越多時，路徑形狀會近似圓形。

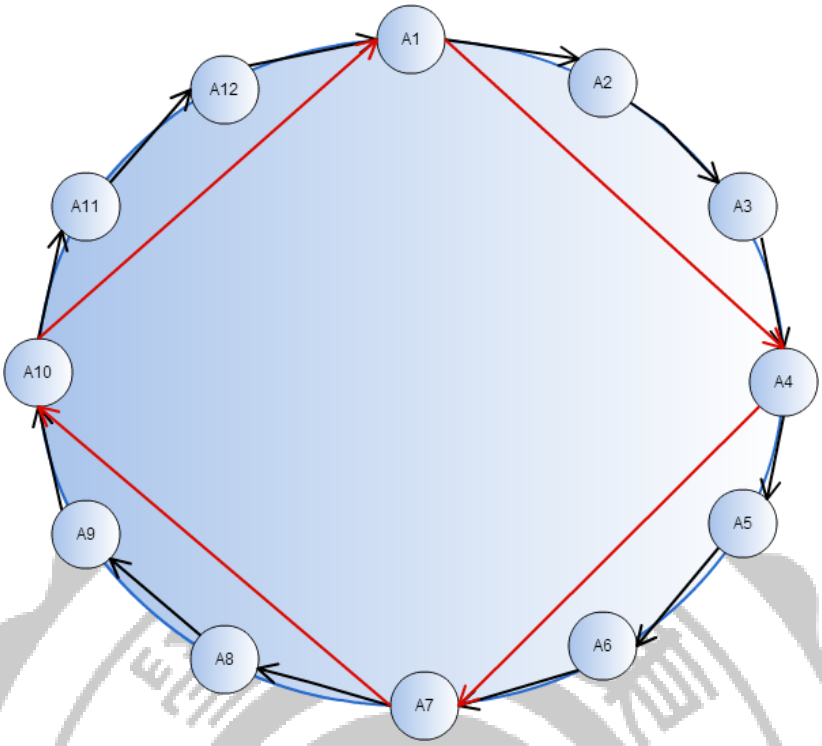

圖 21 節點產生器

$$f(\mathrm{A_{min}}) = (\mathrm{A_1}, \mathrm{A_2}, \mathrm{A_3}, \mathrm{A_4}, \mathrm{A_5}, \mathrm{A_6}, \mathrm{A_7}, \mathrm{A_8}, \mathrm{A_9}, \mathrm{A_{10}}, \mathrm{A_{11}}, \mathrm{A_{12}}) \qquad (3-6)$$

$$f(\mathrm{A_{min}}) = (\mathrm{A_1}, \mathrm{A_4}, \mathrm{A_7}, \mathrm{A_{10}}) \qquad (3-7)$$

3.2.11 時間比值

「平行基因演算法」運算時間除以「一般基因演算法」運算時間，其結果就是時間比值(Time Rate)，當時間比值小於 1 時，表示平行基因演算法計算速度優於一般基因演算法，並稱為「時間比值」，其反比為「加速比」。

設一般基因演算法執行時間為 T_0，而平行基因演算法耗時為 T_1 我們可以用以下公式表示兩者間的關係。

$$\mathrm{TimeRate} = \frac{\mathrm{T1}}{\mathrm{T0}} \qquad (3-8)$$

3.2.12 誤差率

設全域最佳解(global optimal solution)為 r，由基因演算法找出的最短路徑長度減掉全域最佳解的差，再除於全域最佳解，就可以得到誤差率(error rate)。

$$\delta = \frac{L_{min} - r}{r} \qquad (3-9)$$

3.3 相關變因

3.3.1 參數設定

本實驗的參數設定如表 3 所示。

表 3 參數設定

基因編碼	實數編碼+Grefenstette 編碼
適應函數	路線距離總和的反比
選擇方法	輪盤選擇法
交配方法	單點交配法(交配率 0. 8)
突變方法	隨機突變(突變率 0.08)
精英策略	4 組中最好的一個+子代混合
基因演算法染色體個數	64
平行基因演算法染色體個數	16
工作區域數量	4

3.3.2 記憶體影響

圖 22 是基於圖 14 的流程圖的架構下再做修改，主要是在 SPMD 結束到資料重新載入到工作區之前，把除了新族群染色體、路徑矩陣、演化代數、突變機率、交配機率、已知區域最佳解以外的變數清除。要清除的不單只是實驗中使用的變數，還包含在計算過程中產生的隱藏變數，這些變數會造成記憶體不足而使得計算速度下降而影響計算時間的觀測，這部分會在接下來的實驗 4.2 與實驗 4.3 中證實。

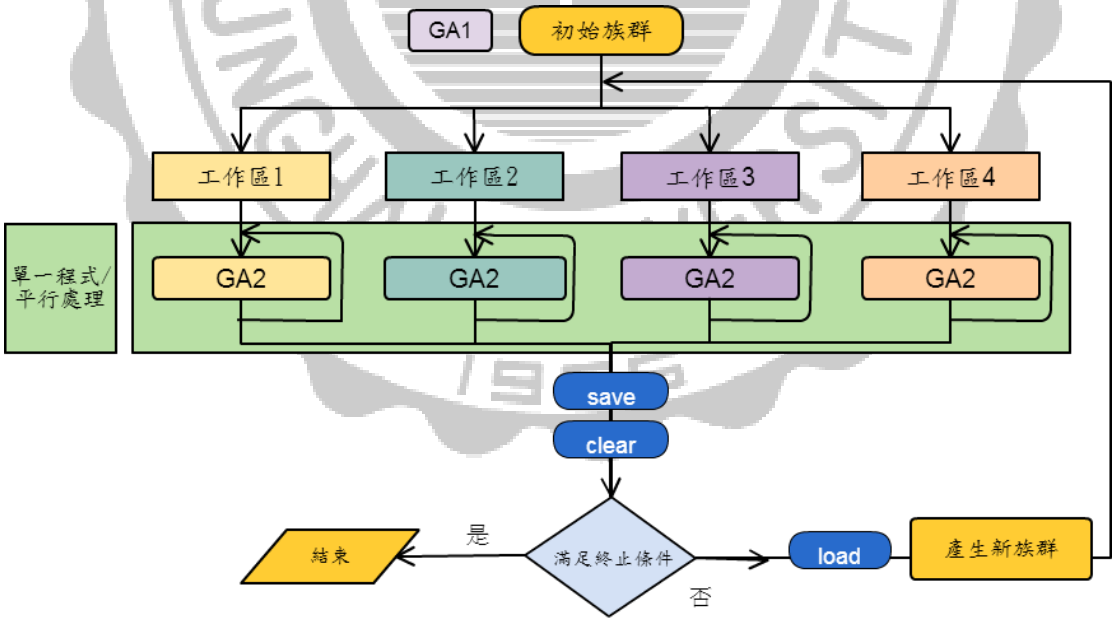

圖 22　修改後的平行基因演算法示意圖

3.3.3 終止條件

一般基因演算法的終止條件有:1. 演化代數 2.找到由節點產生器設定的全域最佳解(global optimal solution)，當以上任一條件滿足，則停止演算法。

第四章 研究結果與分析

影響基因演算法變因的四大要件分別為初始族群的染色體數量(初始族群)、演化的代數(演化代數) 、交配率、突變率，本次實驗探討改良平行基因演算法和一般演算法在不同節點數的比較，並提供改善計算時間的方法。

1. 初始族群的分割。
2. 一般基因演算法與平行基因演算法比較。
3. 改良平行基因演算法的設置與比較。
4. 不同節點數的加速情形。
5. 不同核心數的比較。

4.1 初始族群的分割

初始族群與演化代數的乘積為定值，透過固定的節點數，調整初始族群和演化代數，以檢測它們的變化以及時間的關係。

在 32 個節點的條件下，設定初始族群×演化代數=102400，突變率為 0.08，交配率為 0.8，由節點產生器可知最佳路徑及最佳路徑長度為 6.2731，我們進行基因演算法其結果如表 4 所示，表 4 中的演化代數指的是沒有找到全域最佳解的終止代數，收斂代數則是演化代數結束前找到區域最佳解的執行代數。

從表 4 中可以發現，隨著初始族群不斷變小，誤差率越來越小，但是計算時間也會不斷提高，不過當初始族群的染色體數目低於 16 時，誤差值開始變大。因此我們可以得到兩個結論：

1. 基因演算法的初始族群不是越小越好，所以平行基因演算法的初始族群的染色體數目設置為 16，因為有 4 個工作區在執行，所以一般基因演算法的初始族群的染色體數目就固定為 64。

2. 基因演算法的演化代數設定太長，可能會發生如表 8 中初始族群在 8 的時候，收斂代數停在 4220 而發生過早收斂，使得後面計算只是單純浪費時間，所以在平行基因演算法中設置了混合次數來解決此問題。

表 4　初始族群與演化代數為定值結果表

初始族群	演化代數	收斂代數	時間(s)	最佳解	誤差率
256	400	396	59.4	12.7	102.8%
64	1600	1549	61.8	12.3	96.9%
16	6400	6337	72.6	7.4	18.5%
8	12800	4220	87.62	10.5	68.3%

4.2 一般基因演算法與平行基因演算法比較

為了檢測一般基因演算法和平行基因演算法在相同節點數、初始族群總數與演化代數總數相同，但不同混合次數時的所需時間、最佳解、誤差值的比較。

設置當節點數為 64 時，其它參數皆與表 3 相同時做比較，從表 5 一般基因演算法跟表 6 平行基因演算法的比較中可以發現

1.一般基因演算法的誤差率都遠高於平行基因演算法的誤差率。

2.從表 6 中我們發現運算時間的不合理之處，隨著混合次數的增加應該會穩定的增加運算時間，然而運算時間實際上在混合次數 2 時，卻是出現比混合次數 4 時還要少的情形。

3.從表 6 中把演化代數不斷切割雖然可以使得平行基因演算法得到更佳的解，不過隨著每次為了要產生新的初始族群會增加更多的時間，而且不一定切割越細就一定可以得到更佳的解。

表 5　64 節點一般基因演算法

演化代數	混合代數	時間(s)	最佳解	誤差率(%)
5000	1	420.7	28.1	347.8

表 6　64 節點平行基因演算法

演化代數	混合次數	時間(s)	最佳解	誤差率(%)
5000	1	366.3	27.0	331.4
2500	2	502.7	25.0	299.2
1250	4	483.9	24.7	294.2
625	8	569.0	26.0	314.8

4.3改良平行基因演算法的設置與比較

為了解決在 4.2 中，因為提高混合次數而造成的時間不規律累進問題，所以嘗試在每次 SPMD 平行運算過後都會清 workspace(變數區) 來進行實驗，同 3.3.2 所示。

設定當 64 個節點，初始族群總數為 64 與演化代數總數為 5000，最佳路徑為 6.2807 時，在清空變數區域（workspace）的條件下，不同混合次數的比較，其結果顯示在表 7 與表 8。

表 7　64 節點平行基因演算法

演化代數	混合次數	時間(s)	最佳解	誤差率(%)
5000	1	366.3	27.0	331.4
2500	2	502.7	25.0	299.2
1250	4	483.9	24.7	294.2
625	8	569.0	26.0	314.8

表 8　64 節點修正平行基因演算法

演化代數	混合次數	時間(s)	最佳解	誤差率(%)
5000	1	366.3	27.0	331.4
2500	2	379.6	25.0	299.2
1250	4	410.6	24.7	294.2
625	8	438.1	26.0	314.8

從表 7 與表 8 的實驗結果，我們可以發現，經由清空變數區域（workspace）可以使得運算時間明顯規律許多，如表 8 所示；而且可以使得運算時間大幅減少，如果想要最短的運算時間應該取混合次數 1 為最佳，若想要最佳的誤差率應該取混合次數 4 為佳，並可以從表 8 觀察到每增加混合次數一次所需時間大約 25 秒，這種規律可以對後續觀測程式的運算時間是否正常有所幫助。

4.4 不同節點數的加速情形

本次實驗採用一般基因演算法與平行的基因演算法做比較，在初始族群總數與演化代數總數相同時，探討在不同節點數增加混合次數的時間比值。

演化代數設置為 2000，其它參數設定同表 3 所示。

採用 A(x)，B(x)，C(x)，D(x)方式表示不同演算法和混合次數，x 表示節點個數；舉個例子，C(8)表示在 8 節點數時，混合次數 2 的平行基因演算法，並且根據公式(3-1)與(3-2)設定初始族群的染色體數目與演化代數。

A（x）: 一般基因演算法，初始族群數為 64，演化代數為 2000

B（x）:平行基因演算法，初始族群數為 16，平行演化代數為 2000，混合次數 1

C（x）:平行基因演算法，初始族群數為 16，平行演化代數為 1000，混合次數 2

D（x）:平行基因演算法，初始族群數為 16，平行演化代數為 500，混合次數 4

從表 8 與表 9 中可以發現，分割族群使得路徑優化的機率其實不穩定，而從表 10 與表 11 中可以發現，必須配合混合族群的方式才能穩定找到更佳的解，也就是混合次數必須為兩次以上，不過因為提高混合次數會使得運算時間增加，所以取 2 次為佳。

表 9　A（x）在不同節點之實驗結果

節點 / 資料	A 8	A 16	A32	A64	A128	A256
最佳值	6.1229	6.2429	12.24	32.12	87.08	225.45
誤差率	0	0	95.15	411.39	1286.1	3488.4
時間	24.76	42.83	79.37	170.50	378.00	775.91
時間比值	1	1	1	1	1	1

表 10　B（x）在不同節點之實驗結果

節點 / 資料	B 8	B16	B32	B64	B128	B256
最佳值	6.1229	6.2429	11.41	33.55	86.89	227.49
誤差率	0	0	81.90	434.17	1283.1	3520.8
時間	17.17	25.94	45.75	89.30	262.28	640.49
時間比值	1.4	1.7	1.7	1.9	1.4	1.2

表 11　C（x）在不同節點之實驗結果

節點 / 資料	C 8	C 16	C32	C64	C128	C256
最佳值	6.1229	6.2429	11.23	30.31	83.07	222.44
誤差率	0	0	79.09	382.73	1222.3	3440.4
時間	19.01	28.12	51.73	92.76	328.60	761.57
時間比值	1.3	1.5	1.5	1.8	1.2	1.0

表 12　D（x）在不同節點之實驗結果

節點 / 資料	D 8	D16	D32	D64	D128	D256
最佳值	6.1229	6.2429	11.39	31.73	87.74	224.76
誤差率	0	0	81.62	405.27	1296.7	3477.3
時間	20.27	29.78	60.56	110.52	332.45	772.72
時間比值	1.2	1.4	1.3	1.5	1.1	1.0

透過對六種不同的節點數（8, 16, 32, 64, 128, 256），測試提高混合次數是否可以得到更佳的解與時間比值，由表 8 到表 11 的資料配合時間比值的定義可以繪出圖 23，其意義在於跟一般基因演算法的計算時間相比，平行基因演算法在不同節點數時的時間比值圖；從圖中可以發現，當節點數量小於 64 時，平行基因演算法最快只需要一般基因演算法一半的計算時間，然而隨著節點數的增加，需求時間會逐漸增加。

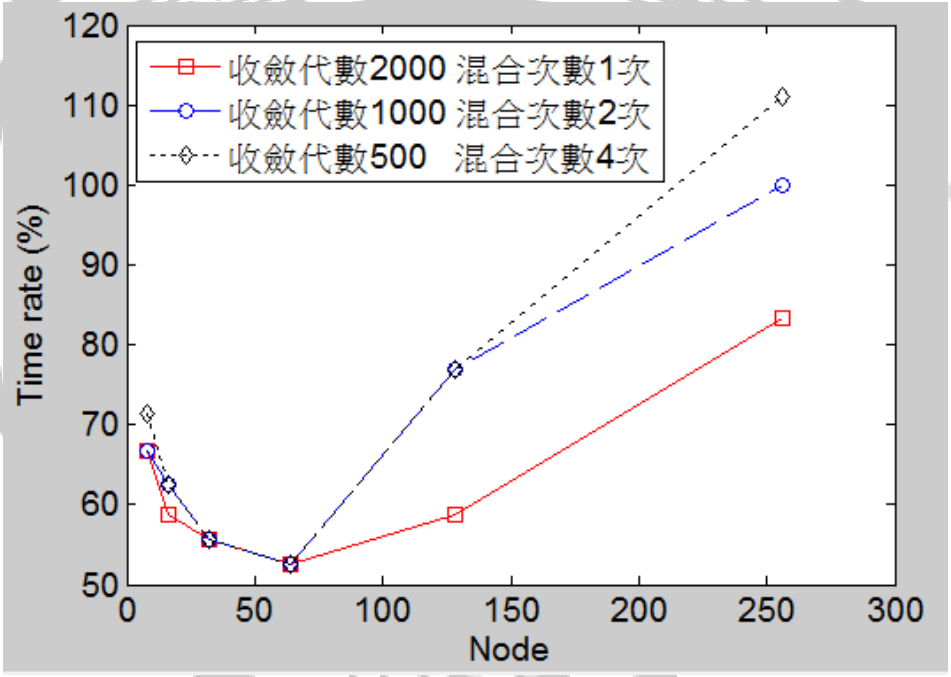

圖 23 不同節點數的時間比值圖

4.5 不同核心數的比較

設混合次數為 2 時，與圖 23 相同條件下使用第二台電腦(PC2)做不同核心數的檢測，其結果如圖 24 所示。因為第二台電腦的核心數比較少，所以在節點數 64 時就開始時間比值開使回升，所以我們可以推測並非演算法有問題才使得時間比值沒有在高節點數時逐漸下滑，可能是硬體受到限制的緣故。此外使用 SPMD 來做平行基因演算法，如果希望降低運算時間，則核心數最好有 4 個以上才會有較明顯的效果。

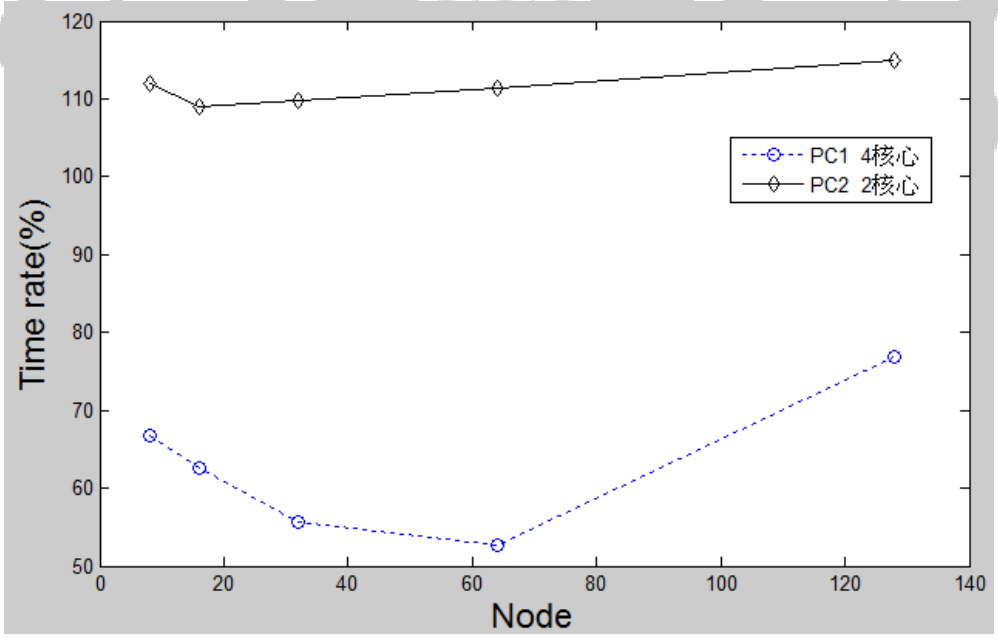

圖 24 不同核心數的時間比值圖

4.6 實驗回顧

從 4.1 的結果中可以發現，初始族群 8 可能因為過早收斂而使得結果比初始族群 16 來得差，所以我們使用平行基因演算法設置混合次數的方法來分割演化代數，解決一般基因演算法因為過早收斂而浪費時間的問題，並且決定後續進行實驗時，平行基因演算法的初始族群數。

從 4.2 的結果中可以發現，把演化代數不斷切割雖然可能因為混合族群次數增加而使得平行基因演算法得到更佳的解，不過隨著每次為了要產生新的初始族群和重新把資料分配到工作區都會使得運算時間增加；而且與 4.1 中的初始族群分割一樣，並非把演化代數切割越細就可以得到更佳的解，同時發現使用平行基因演算法產生計算時間的不穩定。

在 4.3 實驗之前，就曾多次發現相同程式會隨著執行次數而不斷增加運算時間，原本認為是工作區域(lab)多次重啟的影響，但之後發現其實是變數區域(workspace)沒有清空的緣故；因為 Matlab 在執行時間過長或過量運算，會在記憶體遺留中間資料或者控制碼，而使得計算緩慢。由 4.3 的結果，如果想要最佳運算時間，除了電腦重開，還需要在平行運算過程中加入 clear 指令來清除變數。

從 4.4 的實驗結果中可以發現，使用 SPMD 平行計算基因演算法，如果有混合族群的優化在不同節點數相較容易得到更佳的解。當旅行者問題的節點數小於 64，在初始族群數為 64，演化代數為 2000 的情況下，只需要一般基因演算法一半的計算時間。不過隨著節點數的增高與混合的次數增加都會使得計算速度逐漸下滑，目前的實驗結果顯示跟一般基因演算法相比，混合一次有最佳加速比，混合兩次以上可以得到較好的解答。

由 4.5 不同核心數電腦的比較證實，使用 SPMD 來做平行基因演算法，如果想要得到明顯的計算速度加快，電腦的核心數應該在 4 以上，並猜測最佳的時間比值會發生在 64 節點可能是硬體方面的限制，而非程式的問題。

第五章 結論與未來方向

5.1. 結論

在研究初期，經常發生運算時間延長的問題，這對於程式在做時間的比較上產生相當大的問題；經實驗證實，因為分割初始族群來做 SPMD 的平行運算會使得系統殘存過多的隱藏變數而造成運算時間增加，所以在增加了混合族群後只保留重要資料的變數清理其它參數就可以使得計算時間穩定且下降。

本文採用將初始族群分割到不同工作區域(lab)的方式，配合混合族群較佳的染色體來避免陷入局部最佳解的問題來優化成果，並配合平行處理來加速運算時間。由實驗結果，平行基因演算法的在混合次數兩次以上時的確有較好機率提升結果，在節點數小於 64 的情況下都有優秀的加速效果，並在節點數 64 時只需要一般基因演算法一半的計算時間；此外混合次數增加雖然會使得結果較佳，但同樣會使得計算時間增加，所以設定 2 次為佳。另外由 4.5 不同核心數電腦的比較，推測最佳的時間比值會發生在 64 個節點可能是硬體方面的限制。

5.2. 未來方向

本論文使用單一程式多工處理(SPMD)來平行處理基因演算法於旅行者問題，目標為利用多核心的平行處理更快的得到近似最佳解，並且更快速計算；利用初始族群的分割來使得程式不容易陷入到局部最佳解，以及混合族群的重複修正，可以使得平行基因演算法比相同架構的基因演算法容易取得較優秀的解答。但是目前的計算速度和誤差率仍然有改進空間，我們未來會嘗試更新程式版本或者硬體來解決記憶體限制的問題，或者加入如蟻群演算法、退火法...等其它巨集啟發式演算法來優化基因演算法。

參考文獻

[1] 劉昱德，黃士滔，比較三種萬用啟發式演算法於 TSP 問題之探討，工程科技與教育學刊第八卷第三期第 443～452 頁。

[2] 許為元，複合式自我學習之基因演算法應用於旅行推銷員問題，國立台灣大學資訊工程研究所碩士論文，2000。

[3] 劉明竑，植基於雲端技術最佳行車時間之路徑搜尋方法，私立東海大學資訊工程研究所碩士論文，2013。

[4] 秋以強，結合基因演算法與螞蟻演算法求解旅行推銷員問題，國立東華大學資訊工程研究所碩士論文，2012。

[5] 謝欣宏，台鐵司機員排班與輪班問題之研究-以基因演算法求解，國立成功大學交通管理學研究所博士論文，2002。

[6] 李嘉仁，應用基因演算法於智慧型行車資訊系統尋找最短時間路徑，私立東海大學資訊工程研究所碩士論文， 2008。

[7] 廖丞宇，植基於 Hadoop 雲端運算架構之平行基因演算法與粒子群演算法的應用，崑山科技大學資訊管理研究所碩士論文，2013。

[8] 林勇，使用基因演算法求解旅行銷售員問題於自走車路徑規劃之應用實現，國立中央大學電機工程研究所碩士論文，2013。

[9] 侯建花，楊長青，一種求解TSP問題的並行遺傳演算法，計算機仿真第二十二卷第二期第 82～85 頁。

[10] Chu-Hsing Lin, Jung-Chun Liu, Jui-Ling Yu, Chia-Jen Li, “Genetic algorithm for shortest driving time in intelligent transportation systems,” The 2nd International Conference on Multimedia and Ubiquitous Engineering (MUE2008), Hanwha Resort Haeundae, Busan, Korea, April 24 - 26, 2008, pp.402-406. (EI)

[11] Chu-Hsing Lin, Jung-Chun Liu, Hao-Ting Zou, “Investigations of factors affecting the genetic algorithm for shortest driving time,” International Conference on SOft Computing and Pattern Recognition (SoCPaR 2009), December 4-7, 2009, Malacca, Malaysia. (EI)

[12] Dorigo, M., Optimization learning and natural algorithms, Phd thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy, (1992).

[13] Dorigo, M., Maniezzo, V. and Coloeni, “The ant system: optimization by a colony of cooperating agents,” IEEE Transactions on Systems, Man, and Cybernetics-Part B, Vol.26, pp.29-41.

[14] Flood, M. M., (1956), “The travelling salesman problem,” Operations Research, Vol.4, pp.61-75.

[15] Grefenstette, J.J. (1986). "Optimization of control parameters for genetic algorithms". IEEE Transactions Systems, Man, and Cybernetics 16 (1): pp.122–128 .doi:10.1109/TSMC.1986.289288。

[16] Hao-TianZuo “Using genetic algorithm to solve the shortest driving time problem and improvement of population initialization”, SOCPAR '09 Proceedings

of the 2009 International Conference of Soft Computing and Pattern Recognition, pp.106-111.

[17] Holland, J. H.,(1975), Adaptation in natural and artificial systems, Ann Arbor, MI:Univ, Michigan Press.

[18] Karp, R., (1972), "Reducibility among combinatorial problems," Complexity of Computer Computations, Plenum Press, pp.85-104.

[19] Lawler, E. L., Lenstra, J. K., RinnooyKan, A. H. G. and Shmoys, D. B.,(1985) The traveling salesman problem: a guided tour of combinatorial optimization, Wiley and Sons, New York.

[20] Tina, P., Ma, J. and Zhang, D. M., (1999), "Application of the simulated annealing algorithm to the combinatorial optimization problem with permutation property: an investigation of generation mechanism," European Journal of Operational Research, Vol.118, pp.81-94.