由較佳邊集合引導之基因區域搜尋法及其應用於解旅行推銷員問題

Promising Edge Set Guided Genetic Local Search Algorithm for the Traveling Salesman Problem

曾怜玉
國立中興大學資訊網路與多媒體研究所
國立中興大學資訊科學與工程系
lytseng@cs.nchu.edu.tw

余家華
國立中興大學資訊科學與工程系
s9014035@gmail.com

摘要

旅行推銷員問題(Traveling Salesman Problem ; TSP) 為典型的組合最佳化(Combinatorial Optimization)問題之一。自西元 1930 年以來，它便吸引許多不同領域的學者投入研究，然而 TSP 問題已被證明是 NP-hard 問題，因此如何發展出在有限時間內能找出全域最佳解(global optimum)的方法，便是研究學者的最大挑戰。近年來的研究中，發現可利用 TSP 問題修正過之邊成本結合最小生成樹的規則來取得一些較佳邊，稱為較佳邊集合，在過去研究中利用了較佳邊集合來加速求解過程中收斂速度，但較為可惜的是，過去研究並無針對較佳邊集合的特性作進一步的利用。

因此本論文結合較佳邊集合與基因區域搜尋法以期能有效應用於求解 TSP 問題的研究上，且透過較佳邊集合能夠有效連結全域搜尋與區域搜尋的合作機制，使其兩者達到更完善的搜尋效果與求解品質。本論文採用 TSPLIB 題庫之 35 題來測試本研究提出的基因區域搜尋法。在測試結果發現，本研究的基因區域搜尋法相較於目前表現較佳的 LKH 演算法與 ANGEL 演算法，有相當不錯的求解品質與效能表現。

關鍵詞：旅行推銷者問題，基因演算法，由較佳邊集合引導之基因區域搜尋法，LKH 演算法，ANGEL 演算法。

Abstract

The traveling salesman problem is an important combinatorial optimization problem. Since 1930, many researchers had been devoting their efforts in solving this problem. Being an NP-hard problem, the traveling salesman problem is unlikely to be solved in polynomial time. Hence a main research issue is to design efficient algorithms to find near optimal solutions for the TSP. In recent studies, the collection of promising edges by deriving the edges of the minimum spanning trees had been proposed. But the promising edges were not fully utilized in previous studies.

In the paper, we used the promising edge set to guide the search of a genetic local search algorithm. First, the promising edge set was generated by iteratively generating the minimum spanning trees. Then, the promising edge set was used to guide the genetic local search algorithm in searching the promising areas. Furthermore, the promising edge set will evolve along the search. The proposed algorithm was tested on 35 instances taken from TSPLIB. The experimental results revealed that the solution quality of the proposed algorithm is similar to that of the ANGEL and is better than that of the LKH.

Keywords: Traveling salesman problem, genetic algorithm, promising edge set guided genetic local search algorithm, LKH algorithm, ANGEL algorithm.

一、緒論

旅行推銷員問題(Traveling Salesman Problem ; TSP)[22] 為典型的組合最佳化 (Combinatorial Optimization)問題之一。除此之外，TSP也是許多問題的基礎，如排程問題(Scheduling Problem)[6]、車輛巡迴問題(Vehicle Routing Problems)[5]等問題都是TSP問題的延伸。所以自西元 1930 年以來，它便吸引許多不同領域的學者投入研究，然而TSP問題也已被證明是屬於NP-hard問題[16]，因此如何發展出在有限時間內能找出全域最佳解(global optimum)的方法，便是研究學者的最大挑戰。

TSP問題的定義非常單純，從某一城市出發，在所有城市皆走過一次，最後回到起始城市的限制下，找到一個最短路徑解。在較早期的研究，學者們利用確切解法(Exact Algorithm)，如分枝定限法(Branch-and-Bound ; B&B)[31]、動態規劃法(Dynamic Programming ; DP)[20]等方法來求得全域最佳解，可是在TSP問題規模不斷提升的時候，TSP問題的解空間(solution space)也以n階層(n!)的方式在增加，使得求解之時間成指數型的成長。

為解決確切解法的效能瓶頸，研究學者們開始利用近似解法(Approximation Algorithm)裡的超啟發式演算法(Meta-Heuristic)，如基因演算法(Genetic Algorithm ; GA)[19][29]、蟻群最佳化法(Ant Colony Optimization ; ACO)[10][15]、模擬退火法(Simulated Annealing ; SA)[2][26]、禁忌搜尋法(Tabu Search ; TS)[12]等方法與啟發式演算法(Heuristic)，如Lin-Kernighan(LK)[24]和Heslgaun's LK(LKH)[17]等方法及其結合[3][13][25][30]得到相當好的解品質與速度效能。

在近年的研究裡，我們於Heslgaun's LK(LKH)[17]與ANGEL[30]的研究中發現，ANGEL引用LKH的研究來提出一項針對TSP問題特性極為有利的選邊規則，簡單來說，就是利用最小生成樹(minimum spanning tree ; MST)的規則來選擇出較佳的邊。在ANGEL的研究結果中顯示，一般的TSP問題大多可利用執行數次的選邊規則來挑選出大多數甚至全部最佳解的邊來構成一較佳邊集合，但在研究中較為可惜的是，ANGEL並無針對較佳邊集合的初始分佈特徵(有較多最佳解的邊會落在較早取得的較佳邊集合裡，且愈多次的取邊動作會帶來愈多不必要的非最佳解邊)來多作利用，只將其應用在蟻群用來產生解上與交配機制上。

於是本研究提出方法來有效利用上述選邊規則所產生之較佳邊集合(promising edge set ; PES)的初始分佈特徵，以期節省不少求解過程中無效的搜尋時間。我們利用基因區域搜尋法作為本研究的搜尋模型，並嘗試結合較佳邊集合來引導基因區域搜尋法求解 TSP 問題，並建構出一個不錯的搜尋架構。

基於前述目的，本研究發展重點描述如下：

1. 設計出有效利用較佳邊集合來建立較佳初始解的方法：在過去文獻中，除了ANGEL[30]外，並無特別利用較佳邊集合來建立較佳初始解的建構法。因此本研究將設計一個能較為有效利用較佳邊集合的初始解建構法以提升初始解的品質。
2. 對較佳邊集合的品質與數量的控制：較佳邊集合的選邊規則中，雖然能透過較多次的取邊動作來提高最佳解邊的涵蓋率，可是過多的非最佳解邊依舊會帶來相當大的負擔。所以本研究將採用較少次數的取邊動作，雖然會降低最佳解邊的涵蓋率，不過也同時降低非最佳解邊的數量及不必要的搜尋負擔。而相對一開始就沒被選入較佳邊集合的最佳解邊也須利用其他機制來把它找出，並將其置於較佳邊集合裡以提高找到最佳解的機會。
3. 建構基因區域搜尋法與較佳邊集合的結合機制：本研究希望能透過基因區域搜尋法與較佳邊集合的共同合作來強化有效搜尋的能力。本研究擬發展出由較佳解集合引導之基因區域搜尋法。

而本論文架構描述如下，第一節概述TSP問題的研究背景，第二節將回顧TSP問題的相關文獻，以及與本研究相關演算法的介紹，而第三、四節將介紹較佳邊集合的生成機制及詳細說明本研究所

使用的基因區域搜尋法的交配、突變、選擇法則和區域搜尋法等機制。第五節主要針對TSPLIB[28]之35題進行實驗並附上實驗參數與數據，且分析討論實驗結果。第六節為結論。

二、文獻回顧

本節將針對旅行推銷員問題於近年來之發展做一簡單闡述，接下來會把本研究中所使用之基因演算法與 LKH 演算法做一基本介紹。

2.1 旅行推銷員問題

旅行推銷員問題(Traveling Salesman Problem ; TSP)[22]，是指一名推銷員在一些城市推銷貨品，在城市之間距離已知的情況下，並假設每個城市只能被拜訪一次且最後回到起始城市的條件下，找出循環所有城市最短的路徑解(tour)。進一步，本論文利用下面的敘述來定義TSP問題：有 n 城市 $\{c_1, c_2, ..., c_n\}$ 且任二個城市 $\{c_i, c_j\}$ 之間的距離為 $d(c_i, c_j)$，問題目標在找出一個城市排列的順序來可以使下式(1)能得到最小數值：

$$\sum_{i=1}^{n-1} d(c_{\pi(i)}, c_{\pi(i+1)}) + d(c_{\pi(n)}, c_{\pi(1)}) \qquad (1)$$

，這也是 TSP 問題最短路徑解的長度。

此外，對於所有的城市只要 $d(c_i, c_j) = d(c_j, c_i)$ 對所有 $i \neq j$，則稱為 symmetric TSP (STSP)，反之，對某一 $i \neq j$ 且 $d(c_i, c_j) \neq d(c_j, c_i)$，便稱作 asymmetric TSP (ATSP)。

從 1930 年開始到現在，它吸引許多不同領域的學者投入研究。同時TSP問題也被証明是屬於NP-hard[16]問題，所以如何能發展出在有限時間內找出近似最佳解或最佳解的快速近似解法，便是研究學者的最大挑戰。

就目前的研究而言，對於求解 TSP 問題的方式可概分為二類：確切解法(Exact Algorithm)與近似解法(Approximation Algorithm)，可見圖 1。

在早期的研究學者們利用確切解法(Exact Algorithm)；如分枝定限法(Branch-and-Bound ; B&B)[31]、動態規劃法(Dynamic Programming ; DP)[20]等方法來求得全域最佳解，可是在TSP問題規模不斷提升的時候，TSP的解空間也以n階層(n!)的方式在增加，使得求解的時間成指數型的成長，因此確切解法只能解較小型的TSP。

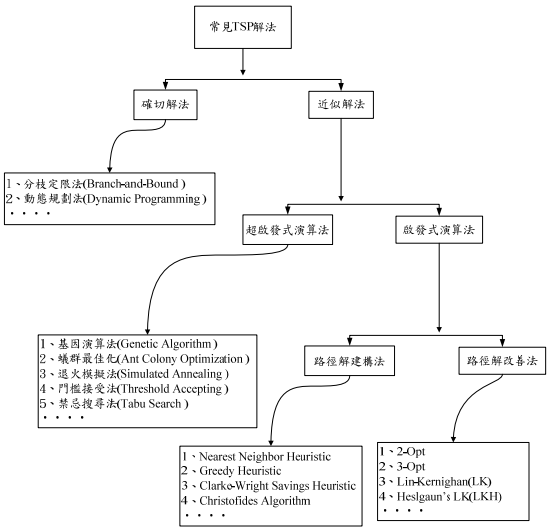


圖 1　常見 TSP 解法之分類圖

而較後期的研究裡，研究學者們開始利用近似解法來求解TSP，在過去的研究文獻中可以發現數種近似解法在於解決TSP問題的最佳化上皆有不錯的表現，如 Domain-Specific Local Search Algorithms[23][24][27]、基因演算法(Genetic Algorithm ; GA)[19][29]、蟻群最佳化法(Ant Colony Optimization ; ACO) [10][15]、退火模擬法(Simulated Annealing ; SA)[2][26]、門檻接受法(Threshold Accepting ; TA)[14]、禁忌搜尋法(Tabu Search ; TS) [12]、Elastic Nets[11]和類神經網路(Neural Networks ; NN)[1]等。

近似解法還可再依搜尋特性分為二種方法：超啟發式演算法(Meta-Heuristic)與啟發式演算法(Heuristic)，在上面所提到的，如基因演算法、蟻群最佳化法、退火模擬法、門檻接受法、禁忌搜尋法等方法皆屬於超啟發式演算法，由於具跳脫區域最佳解(Local Minima)的能力，一般而言，都可以找到品質比啟發式演算法所能找到的解還好的解。

而啟發式演算法多是利用較單純規則來作為

求解的依據，追求能在短時間內找出近似最佳解或全域最佳解的方法。啟發式演算法依方法不同可分成二類：路徑解建構法(Tour Construction Heuristic)和路徑解改善法(Tour Improvement Heuristic)。路徑解建構法是逐步建立一個較佳的初始路徑解，作法通常是先在城市集合中隨機選一個城市為起點，然後利用一些簡單的規則選出最有優勢的下個城市，之後重覆選城市動作，直到產生符合TSP問題的路徑解為止，如Nearest Neighbor Heuristic[21]、Greedy Heuristic[21]、Clarke-Wright Savings Heuristic[8]和Christofides Algorithm[7]等方法。另外，路徑解改善法是利用一組已產生好的路徑解來優化，利用交換改變路徑解中的城市之排序位置，來讓路徑解達到區域最佳解為止，最終期望能達到全域最佳解。通常使用的路徑解改善法為 2-Opt[9]、3-Opt[4][23]、Lin-Kernighan(LK)[24]和Heslgaun's LK(LKH)[17]。從過去的研究中顯示，雖然啟發式演算法的速度都很快，但通常容易使得路徑解落入區域最佳解而無法得到全域最佳解。

因此，近年來TSP問題的研究多數著重於結合超啟發式演算法與啟發式演算法，所幸在超啟發式演算法與啟發式演算法的有效互補作用之下，使得不少研究結果皆有極佳的求解品質與效能[3][13][25][30]。

2.2 基因演算法

基因演算法(Genetic Algorithm；GA)的概念是由John Holland [18]於 1975 年首度發表，其概念是透過科學方法去表現大自然「物競天擇、適者生存、不適者淘汰」的演化過程。

在基因演算法中最主要的元素便是染色體(chromosome)，一條染色體就代表一個個體，染色體則由許多遺傳基因(genes)組成，每個基因對應著此物種某些特性。許多個體的集合便形成一個母體族群(population)，而基因演算法便是模擬該母體族群在生物界的自然演化過程。演化過程是經由染色體交配(crossover)、染色體突變(mutation)與選擇(selection)的機制來達成。

此外為了能夠有效評估染色體的優劣程度。尚需設計一合理的評估方法，通稱之為適應函式(fitness function)，每一條染色體經由適應函式評估後將得到一個適應值(fitness value)。適應值較佳的染色體通常代表擁有較佳的基因特徵，因而在母體族群中適應值較佳的染色體應可擁有較大機會保留至下一世代。一般而言，適應函式與問題的目標函式有相當大的關係。

基因演算法的主要三個遺傳法則如下：

1. 交配機制：係指兩條父代染色體(parents)透過配對交換機制產生一至多條的子代染色體(children)，並且子代染色體將會遺傳父代的染色體的部份特徵，這也是整個交配機制中的重要特點。
2. 突變機制：則是在族群的演化過程中一些染色體發生了不可預期的變化現象，使得子代染色體含有父代染色體所不具有的特徵。
3. 選擇機制：從族群中選出適應值較好的染色體保留到下一世代，使得較「優良」的遺傳因子能流傳與繼承。

所以基因演算法的一個世代就是經過交配、突變、選擇，這三個動作，藉由世代的交替將使得族群不斷的演化，期待產生的子代能愈來愈好。

迄今基因演算法已應用到許多求解NP-hard及最佳化的問題中，如排程問題(Scheduling Problem)[6]、旅行銷售員問題(Traveling Salesman Problem；TSP) [19][29]、車輛巡迴問題(Vehicle Routing Problems)[5]等問題上皆有相當不錯結果。

2.3 LKH 演算法

Helsgaun's LK(LKH)[17]是利用Lin-Kernighan (LK)[24]的基本原理加以改良的啟發式演算法。簡單來說，LKH主要改良的特色有二點：強化節點順序重排機制與提高節點順序重排之有效性的設計，以下將依序說明這二大特色的重點。

在強化節點順序重排機制上，LKH 保留原本LK 的循序式 2-opt、3-opt 與 4-opt 等如圖 2 所示的換邊規則外，又加入循序式 5-opt 與非循序式 4-opt，

如圖 3 所示，來改善可能因換邊次數太少或循序換邊的盲點而無法進一步改良路徑解的問題。在提高節點順序重排之有效性上，LKH 也針對 LK 的挑選候選邊集合(candidate set)方法作了修改。原始 LK 的節點順序重排設計裡，只對路徑解嘗試交換在候選邊集合裡的邊以減少不必要的交換，而 LK 挑選候選邊集合的方式卻只利用原始的邊之成本來決定。在 Helsgaun 分析 LK 挑選候選邊集合的規則之後，覺得 LK 候選邊集合對最佳解邊的涵蓋率依舊不高，於是在 LKH 導入一個較佳的選邊依據概念，稱為 $\alpha-nearness$ 。

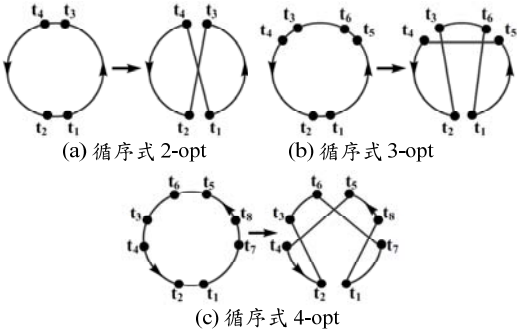


圖 2　循序式 2-opt、3-opt 與 4-opt 示意圖

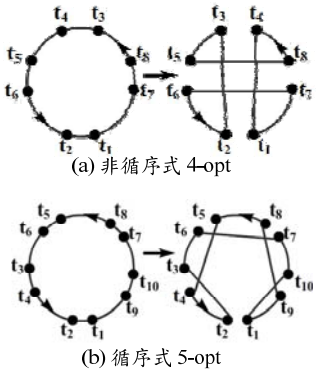


圖 3　非循序式 4-opt 與循序式 5-opt 示意圖

$\alpha-nearness$ 主要是藉由 minimum 1-tree 與最佳解之間差異所產生的相對貢獻值而定，minimum 1-tree 之定義可參考圖 4 所示，Helsgaun 藉由分析得知 minimum 1-tree 通常包含了最佳解邊的 70%到 80%。因此利用 $\alpha-nearness$ 定能取得比 LK 候選邊集合更正確的候選邊集合。$\alpha-nearness$ 的定義如下式(2)：

$$\alpha(r,s) = L(M_1^{+}(r,s)) - L(M_1) \qquad (2)$$

，$L(M_1)$ 是指 minimum 1-tree 的長度，$L(M_1^{+}(r,s))$ 是指強制必須包含邊 (r,s) 的 minimum 1-tree 長度。所以與 minimum 1-tree 完全相符之邊的 $\alpha-nearness$ 會等於 0，因此 LKH 只須利用 $\alpha-nearness$ 的大小便可更精確地決定該邊的優劣程度，當然也增加了節點順序重排的有效性。

於文獻[17][30]中可知，LKH雖無法保證解題的成功率，但透過較佳的候選邊集合與強力的換邊規則的幫助下，LKH的解題成效仍是其他啟發式演算法難以勝過的。

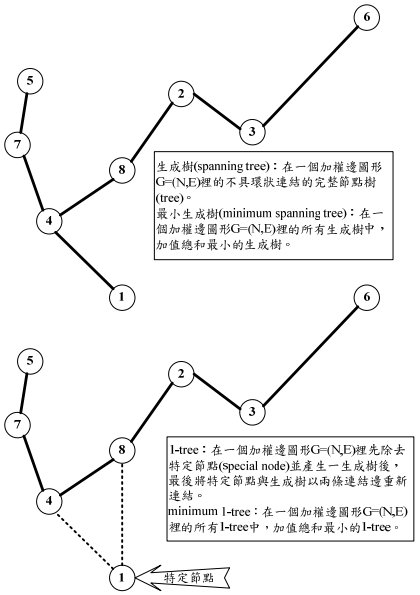


圖 4　最小生成樹與 minimum 1-tree 之基本定義關係圖

三、較佳邊集合

本節將描述較佳邊集合(promising edge set ; PES)的生成機制及介紹加權邊成本的定義與應用，並利用較簡潔的圖表將較佳邊集合的優勢作一簡單歸納說明。

3.1　生成機制

由過去研究[17] [30]可得知，最小生成樹(minimum spanning tree ; MST)通常可以包含很大比率的最佳邊解。因此本研究也利用最小生成樹作

為較佳邊集合的取邊規則。而較佳邊集合的生成流程如下：

Step 1. 首先將較佳邊集合設為空集合。

Step 2. 重覆執行 Step 3 與 Step 4 共 p 次。

Step 3. 排除已在較佳邊集合裡的邊，再次執行最小生成樹的取邊規則。

Step 4. 將最小生成樹取出來的邊放入較佳邊集合裡。

依上述流程可得知，在執行 p 次的取邊規則後，較佳邊集合便會包含 $p(n-1)$ 個較具優勢的邊。下面表 1 是針對本研究實驗題組TSPLIB[28]之 35 題中隨機選出 20 題在執行 7 次取邊規則後，記錄每增加一次取邊動作後還有多少最佳解的邊被遺漏。

經由表 1 的觀察，我們清楚了解在經過多次的取邊動作後，已有不少題目可收集到將近乎百分百的最佳解邊，如執行 4 次取邊動作後，在 20 題中的 2 題已經達到最佳解邊的零遺失率(missing rate)，在執行 7 次取邊動作後，已有 8 題達到最佳解邊的零遺失率。

除此之外，於表 1 中我們也觀察到較佳邊集合的初始分佈特徵-有較多最佳解的邊會落在較早取得的最小生成樹裡，且愈多次的取邊動作會帶來較多不必要的非最佳解邊。然而在過去的研究[30]中卻沒有針對上述特徵作有效的應用，實為可惜。然而我們使用原始邊成本來產生較佳邊集合的作法，雖然已得到不少最佳解的邊，但我們其實可以再透過一個簡單規則來改變原始邊成本以加速取得最佳解的邊，因此接下來我們將介紹：原始邊成本的改良-加權邊成本，這方法是由Helsgaun[17]提出來的。

3.2 加權邊成本

一般而言，我們知道每個路徑解(tour)其實都是一個 1-tree[17]，因此在[17]中，Helsgaun利用minimum 1-tree來取得TSP問題最佳路徑解的下限，然而在minimum 1-tree的取邊規則之下，其實容易使得每個節點的連結次數(degrees)不穩定，但如能將minimum 1-tree在保留住最佳順位邊的原則下調整到接近路徑解的話，那麼修正後minimum 1-tree就能更接近於最佳路徑解，而minimum 1-tree的最佳順位邊，也就是minimum 1-tree中考慮各個節點，與此節點連結的邊當中最短的即為最佳順位邊。基

表 1 分析較佳邊集合在不同取邊次數下的最佳解邊遺失率

	p	1		2		3		4		5		6		7	
		Missing		Missing		Missing		Missing		Missing		Missing		Missing	
Problem	n	Count	Rate	Count	Rate	Count	Rate	Count	Rate	Count	Rate	Count	Rate	Count	Rate
u574	574	137	0.2387	60	0.1045	21	0.0366	9	0.0157	6	0.0105	5	0.0087	3	0.0052
rat575	575	150	0.2609	33	0.0574	4	0.0070	0	0	0	0	0	0	0	0
p654	654	207	0.3165	28	0.0428	9	0.0138	7	0.0107	5	0.0076	5	0.0076	5	0.0076
d657	657	164	0.2496	54	0.0822	22	0.0335	7	0.0107	1	0.0015	0	0	0	0
u724	724	177	0.2445	57	0.0787	16	0.0221	4	0.0055	2	0.0028	0	0	0	0
rat783	783	185	0.2363	43	0.0549	6	0.0077	4	0.0051	1	0.0013	0	0	0	0
dsj1000	1000	256	0.2560	78	0.0780	31	0.0310	17	0.0170	12	0.0120	9	0.0090	3	0.0030
pcb1173	1173	260	0.2217	54	0.0460	20	0.0171	10	0.0085	7	0.0060	3	0.0026	1	0.0009
d1291	1291	154	0.1193	79	0.0612	51	0.0395	33	0.0256	26	0.0201	19	0.0147	17	0.0132
rl1304	1304	151	0.1158	93	0.0713	59	0.0452	43	0.0330	27	0.0207	19	0.0146	12	0.0092
rl1323	1323	143	0.1081	87	0.0658	58	0.0438	42	0.0317	29	0.0219	21	0.0159	15	0.0113
nrw1379	1379	341	0.2473	76	0.0551	16	0.0116	4	0.0029	1	0.0007	0	0	0	0
fl1400	1400	578	0.4129	160	0.1143	39	0.0279	19	0.0136	13	0.0093	9	0.0064	8	0.0057
u1432	1432	436	0.3045	48	0.0335	13	0.0091	5	0.0035	1	0.0007	1	0.0007	0	0.0000
rl1889	1889	239	0.1265	109	0.0577	65	0.0344	45	0.0238	32	0.0169	23	0.0122	17	0.0090
u2319	2319	1050	0.4528	78	0.0336	5	0.0022	0	0	0	0	0	0	0	0
pr2392	2392	551	0.2304	143	0.0598	65	0.0272	29	0.0121	15	0.0063	3	0.0013	3	0.0013
pcb3038	3038	692	0.2278	140	0.0461	33	0.0109	16	0.0053	7	0.0023	4	0.0013	0	0
fl3795	3795	894	0.2356	168	0.0443	86	0.0227	35	0.0092	25	0.0066	21	0.0055	19	0.0050
fl5934	5934	582	0.0981	287	0.0484	172	0.0290	103	0.0174	63	0.0106	44	0.0074	31	0.0052

於上述觀點，Helsgaun藉由導入懲罰值 π 來改變邊成本，使得修正後的minimum 1-tree得到更為接近最佳解的狀態。接下來將描述如何利用懲罰值 π 來推導出一個能修正最佳路徑解下限的目標式。

首先可假設懲罰值 π 皆為同樣數值並將邊成本改由式(3)來呈現：

$$d_{ij} = c_{ij} + \pi_i + \pi_j \qquad (3)$$

，c_{ij} 是代表原始的邊成本，d_{ij} 是加權後的邊成本，因此加權後的完整路徑解會剛好增加 $2\sum\pi_i$ 的大小。在此最佳路徑解下限也改成式(4)來表達：

$$w(\pi) = L(T_\pi) - 2\Sigma\pi_i \qquad (4)$$

，T_π 是指利用加權邊成本所產生的 minimum 1-tree，$L(T_\pi)$ 則是 T_π 的長度。推導到此，已找到能有效修正最佳路徑解下限的目標式，即為式(4)。因為利用式(4)可得知在 π_i 非皆同值的情況下，$w(\pi)$ 的大小將會與 $2\sum\pi_i$ 有絕對的關係，只有愈接近完整路徑解的 T_π 愈能使 $2\sum\pi_i$ 的反作用效果降低。由此推論，在 T_π 保持原始最佳順位邊不變的情況下，愈接近完整路徑解的 T_π 將容易得到較大的 $w(\pi)$，相反的，較不具完整路徑解形狀的 T_π 就會容易得到較小的 $w(\pi)$。所以目標便是在不變動原始 minimum 1-tree 中最佳順位邊的前提下，找出一組 $\pi = \{\pi_1, \pi_2, ..., \pi_n\}$ 來使得 $w(\pi)$ 最大化。

Helsgaun 利用次梯度最佳化 (subgradient optimization)的方式來獲得最大化的 $w(\pi)$。次梯度最佳化的演算法如圖 5 所示。在次梯度最佳化過程中，以節點的連結次數為適應條件，如果能使節點的連結次數皆為 2，就算達到完全最佳化的目的，也代表已找到路徑最佳解與完全精確的加權邊成本。

在此我們重新利用加權邊成本來取得新的較佳邊集合，對表 1 的 20 題進行取邊並記錄於表 2 中。觀察表 2 的數據，我們可清楚得知加權邊成本修正了原始邊成本的不足，使得更多最佳路徑解的邊能提早被發現，如執行 4 次取邊動作後，在 20 題中已經有 9 題達到最佳解邊的零遺失率(missing rate)。因此本論文之較佳邊集合是利用加權邊成本與較佳邊生成機制來共同產生，以提供更正確的初始較佳邊資訊。

表 2　分析加權邊成本之較佳邊集合在不同取邊次數下的最佳解邊遺失率

	p	1		2		3		4		5		6		7	
		Missing		Missing		Missing		Missing		Missing		Missing		Missing	
Problem	n	Count	Rate	Count	Rate	Count	Rate	Count	Rate	Count	Rate	Count	Rate	Count	Rate
u574	574	63	0.1098	5	0.0087	1	0.0017	0	0	0	0	0	0	0	0
rat575	575	89	0.1548	2	0.0035	0	0	0	0	0	0	0	0	0	0
p654	654	178	0.2722	8	0.0122	3	0.0046	2	0.0031	2	0.0031	2	0.0031	2	0.0031
d657	657	84	0.1279	10	0.0152	2	0.0030	0	0	0	0	0	0	0	0
u724	724	112	0.1547	11	0.0152	1	0.0014	0	0	0	0	0	0	0	0
rat783	783	76	0.0971	8	0.0102	0	0	0	0	0	0	0	0	0	0
dsj1000	1000	145	0.1450	18	0.0180	0	0	0	0	0	0	0	0	0	0
pcb1173	1173	142	0.1211	13	0.0111	3	0.0026	1	0.0009	0	0	0	0	0	0
d1291	1291	91	0.0705	27	0.0209	15	0.0116	7	0.0054	5	0.0039	1	0.0008	1	0.0008
rl1304	1304	90	0.0690	25	0.0192	11	0.0084	8	0.0061	6	0.0046	3	0.0023	2	0.0015
rl1323	1323	89	0.0673	24	0.0181	14	0.0106	11	0.0083	5	0.0038	3	0.0023	3	0.0023
nrw1379	1379	168	0.1218	14	0.0102	0	0	0	0	0	0	0	0	0	0
fl1400	1400	571	0.4079	127	0.0907	25	0.0179	4	0.0029	2	0.0014	2	0.0014	2	0.0014
u1432	1432	281	0.1962	17	0.0119	2	0.0014	0	0	0	0	0	0	0	0
rl1889	1889	153	0.0810	29	0.0154	12	0.0064	6	0.0032	5	0.0026	4	0.0021	4	0.0021
u2319	2319	583	0.2514	45	0.0194	6	0.0026	0	0.0000	0	0.0000	0	0.0000	0	0.0000
pr2392	2392	280	0.1171	38	0.0159	1	0.0004	0	0	0	0	0	0	0	0
pcb3038	3038	403	0.1327	29	0.0095	6	0.0020	1	0.0003	0	0	0	0	0	0
fl3795	3795	814	0.2145	175	0.0461	36	0.0095	19	0.0050	15	0.0040	11	0.0029	11	0.0029
rl5934	5934	357	0.0602	102	0.0172	50	0.0084	26	0.0044	19	0.0032	13	0.0022	11	0.0019

/* π^k, v^k and d^k are penalty vector, subgradient vector and vector of degress of nodes.*/
1. Let k = 0, $\pi^0 = 0$ and W = $-\infty$.
2. Find a minimum 1-tree, T_π^k.
3. Compute $w(\pi^k) = L(T_\pi^k) - 2\sum\pi_i$.
4. Let W = max(W, $w(\pi^k)$).
5. Let $v^k = d^k - 2$, where d^k contains the degress of nodes in T_π^k.
6. If $v^k = 0$ (T_π^k is an optimum tour), or a stop criterion is satisfied, then stop.
7. Choose a step size, t^k, t^k is a positive scalar and t^0 is set to 1.
8. Let $\pi^{k+1} = \pi^{k+1} + t^k(0.7v^k + 0.3v^{k-1})$ where $v^{-1} = v^0$.
9. Let k = k+1 and go to Step 2.

圖 5 次梯度最佳化演算法虛擬碼

(出處: K. Helsgaun, An effective implementation of the Lin–Kernighan traveling salesman heuristic [17])

四、由較佳邊集合引導之基因區域搜尋法

本論文提出由較佳邊集合引導之基因區域搜尋法(Promising Edge Set Guided Genetic Local Search Algorithm ; PESGLS)並用以解旅行推銷員問題(Traveling Salesman Problem ; TSP)。

為了能加快求解速度與提高解品質，本論文所應用的基因區域搜尋法除了運用基因演算法作為全域搜尋架構之外，更採用 LKH 演算法為區域搜尋架構以期更能發揮較佳邊集合的優勢。

本節重點在於介紹由較佳邊集合引導之基因區域搜尋法所運用之交配機制、突變機制、選擇機制及其他相關設定。

4.1 整體架構

首先說明整個由較佳邊集合引導之基因區域搜尋法的架構，一般的基因區域搜尋法是利用基因演算法的演化過程中加入了區域搜尋的機制，換言之，就是在交配機制後馬上對子代作區域搜尋以加快求解進步的速度。可是在不作任何限制的前提之下，傳統的基因區域搜尋法會任由基因演算法到處搜尋，且為了能在較短的時間內判斷是否為較佳解的動機下，時常會減少區域搜尋階段應有的步驟而使得求解品質不佳。

本論文的基因區域搜尋法結合了較佳邊集合的優勢，以降低不必要的全域搜尋成本並強化區域搜尋的能力，此外透過適度且多樣的發散機制來補強初始較佳邊集合未能包括最佳路徑解之所有邊的不足，以求達到更好的求解效能與品質。整個演算法架構如下圖 6 所示。

Procedure：PESGLS
1 /* Initialization phase */
2 Generation of the *promising edge set* (PES)
3 Initialization of *m* tours by three methods
4 Apply *local search* is applied to improve each new tour
5 /* Improvement phase (GLS) */
6 **While** (Global optimum not yet been found and predefined number of iterations not yet been reached) **do**
7 **For** i = 1 **to** (*m/2*) **do**
8 Select two parent tours randomly from unselected tours
9 Apply *OX crossover* or *PEOX crossover* operator to parent tours
10 Apply *local search* to improve each child tour
11 **If** child tour is not better than parent tours **then**
12 Apply *Mutation to* child tour with predefined probability
13 **End If**
14 **If** The tour is better than the best tour found thus far **then**
15 Apply the *PES* updating rule
16 **End If**
17 **End For**
18 Select *m* tours by *Selection* rule
19 /* finished *one* iteration*/
20 **End While**

圖 6 由較佳邊集合引導之基因區域搜尋法之虛擬碼

4.2 初始解建構法

本論文中的初始解建構法主要以應用較佳邊集合之初始分佈特徵與具備適度的廣度發散能力為設計重點。依上述要點，我們提出三種初始解建構法，依廣度發散能力的程度多寡，由低至高依序為：4.2.1 較佳邊優先之貪婪建構法。4.2.2 較佳邊優先之隨機建構法。4.2.3 參考較佳邊之隨機建構法。

4.2.1 較佳邊優先之貪婪建構法

在說明較佳邊優先之貪婪建構法前，我們先定義較佳邊子集合(promising edge subset)一詞，對每一節點都有一較佳邊子集合，每一節點的較佳邊子集合可完全包含較佳邊集合裡與此節點相連的所有邊。此外，較佳邊子集合為有序排列，其排列順序與較佳邊集合的生成順序是完全相同的。

由於接下來的論文中將會出現較多的較佳邊子集合一詞，所以在此先行統一說明。其兩者關係可參考圖 7。

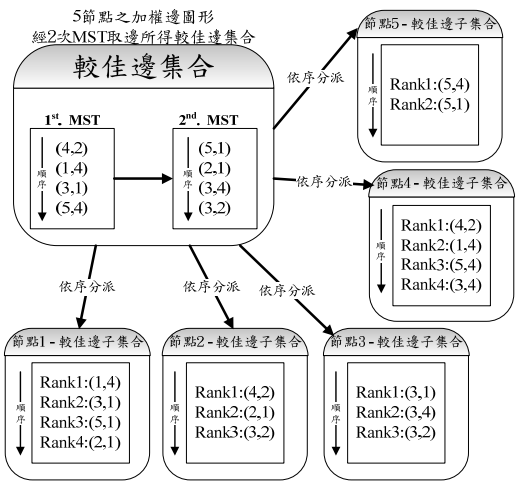


圖 7　較佳邊集合與較佳邊子集合關係圖

較佳邊優先之貪婪建構法主要是以利用較佳邊集合的初始分佈特徵為重點。建構動作的一開始，將會隨機選擇一個節點作為建構過程中的指標節點，指標節點是指用來決定下一個節點的基準節點，指標節點會先以較大機率 k_1 優先嘗試該節點的較佳邊子集合裡排序第一的候選節點，或較低機率 k_2 任意選擇候選節點，$k_1+k_2=1$，當該候選節點無法選取時，才會再依序嘗試排序較後的候選節點，被選出來的候選節點將再成為指標節點，如此重覆下去，直到完成初始路徑解為止。但如果指標節點的候選節點中皆無法挑選來擴展路徑時，便會執行貪婪插入機制來定出新的指標節點。

而貪婪插入機制的運作方式是先任選一個未選節點並找到一個離它最近的已選節點，接著透過該已選節點去找到一個接近於上述任選節點並與它距離最近的未選節點作為新的指標節點，最後將新的指標節點插入正建構路徑解中的最近位置，完成後便回歸到執行原本的建構動作。

以下用一簡單例子來說明較佳邊優先之貪婪建構法的運作過程，見圖 8，建構動作一開始如圖 8(a)所示，隨機從未選取節點中選擇一個節點為指標節點，接下來將視圖 8(b)的兩種情況決定是否優先挑選參考指標節點的較佳邊子集合之第一候選節點或任意候選節點，在挑選的候選節點無法擴展

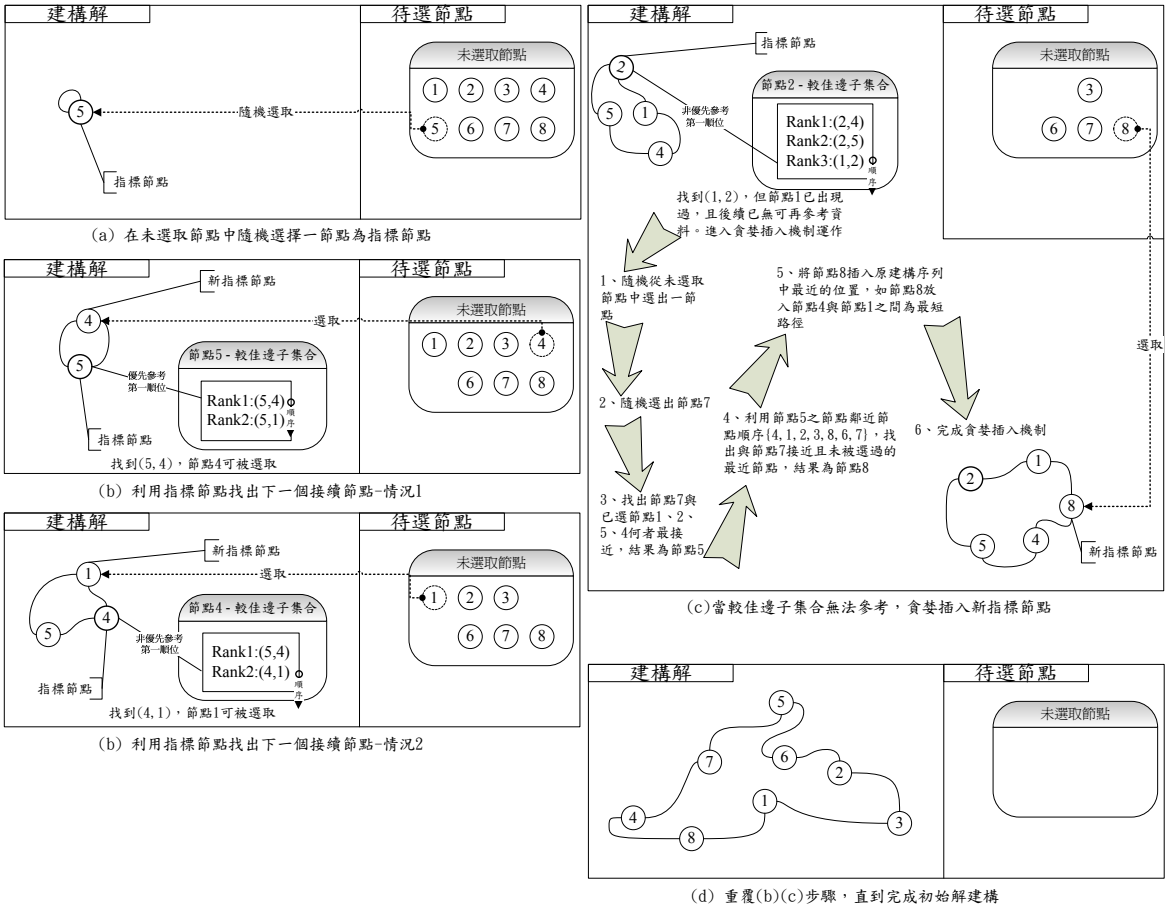


圖 8　較佳邊優先之貪婪建構法運作範例

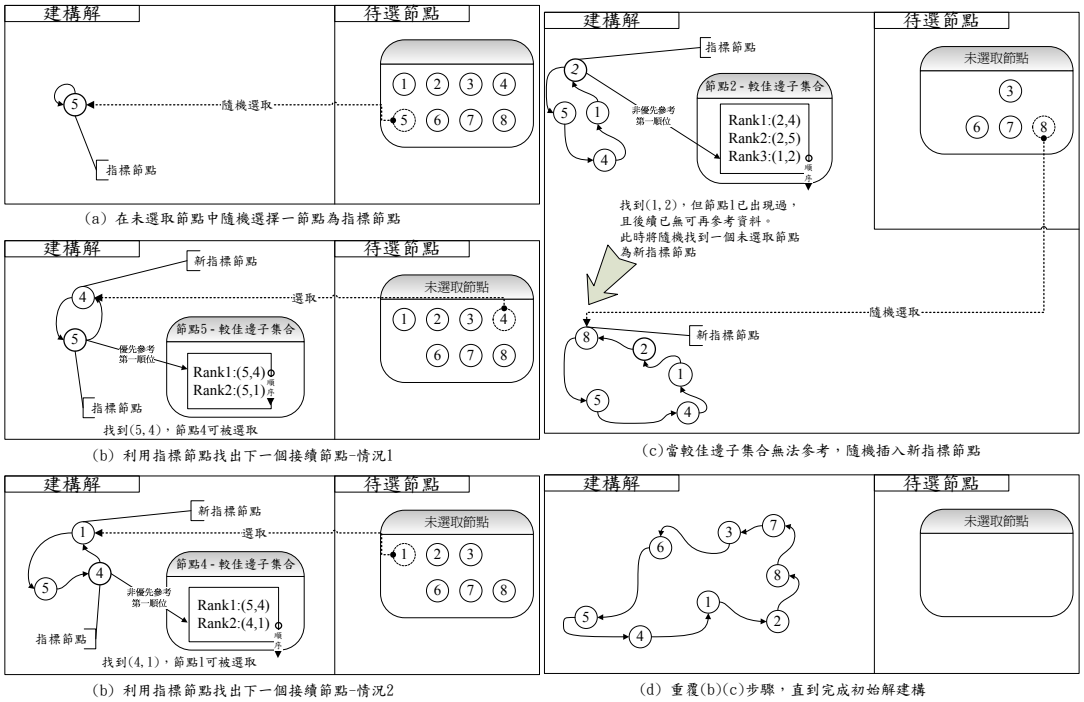


圖 9　較佳邊優先之隨機建構法運作範例

路徑時，會依序把該候選節點之後的節點都會嘗試過，直到無候選節點可選取為止。如發生無候選節點可選之時，會進入圖 8 (c)的狀態，藉由貪婪插入機制找到新的指標節點，重新又開始回到圖 8 (b)的動作，直到建構解完成，如圖 8 (d)。

4.2.2　較佳邊優先之隨機建構法

較佳邊優先之隨機建構法與較佳邊優先之貪婪建構法極為相似，便不再重覆描述相同的說明，而兩者的不同之處是如果指標節點的候選節點中皆無法挑選來擴展路徑時，便會隨機選擇一個未選過的節點來定出新的指標節點。

以下用一簡單例子來說明較佳邊優先之隨機建構法的運作過程，見圖 9，建構動作一開始如圖 9 (a)所示，將隨機從未選取節點中選擇一個節點為指標節點，接下來將視圖 9(b)的兩種情況決定是否優先挑選參考指標節點的較佳邊子集合之第一候選節點或任意候選節點，在挑選的候選節點無法擴展路徑時，會依序把該候選節點之後的節點都會嘗試過，直到無候選節點可選取為止。如發生無候選節點可選之時，會進入圖 9(c)的狀態，藉由隨機挑點來找到新的指標節點，重新又開始回到圖 9 (b)的動作，直到初始建構解完成，如圖 9 (d)。

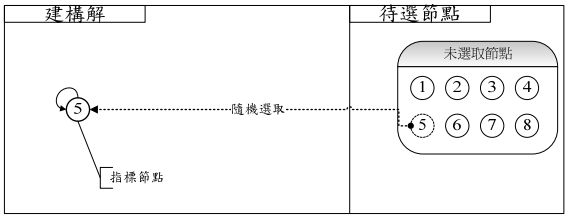
4.2.3　參考較佳邊之隨機建構法

參考較佳邊之隨機建構法也利用較佳邊集合來增高最佳解邊的取得機率，與前兩個建構法的不同在於此建構法並不利用較佳邊集合的初始分佈特徵來強化取點的偏向，相反的是利用隨機的特點來強化全域搜尋的能力以增加找到未知最佳解邊的機會。

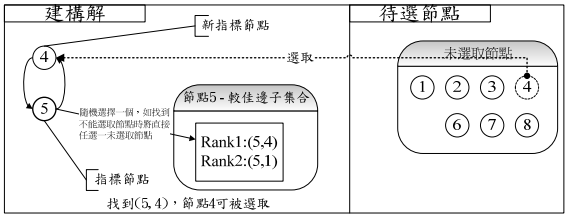
建構動作的一開始，也是隨機選擇一個節點作為建構過程中的指標節點，指標節點會先以較大機率 k_3 隨機選擇一個位於較佳邊子集合裡的候選節點，或較低機率 k_4 直接隨機選擇一個未選過的節點，$k_3 + k_4 = 1$，當該候選節點無法擴展路徑時，會再隨機選擇一個未選過的節點來定出新的指標節點，如此重覆下去，直到完成初始路徑解為止。

以下用一簡單例子來說明較佳邊參考之隨機建構法的運作過程，見圖 10，建構動作一開始如圖 10 (a)所示，將隨機從未選取節點中選擇一個節點為指標節點，接下來如圖 10 (b)情況所示，將隨機挑選指標節點的較佳邊子集合裡的一個候選節點或直接隨機挑選一未選節點為新指標節點，而在

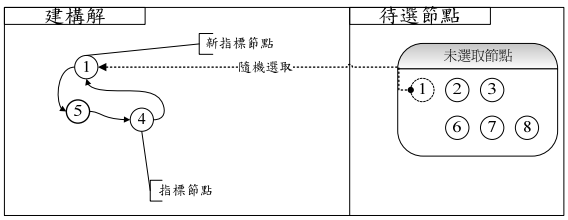
較佳邊子集合裡挑選的候選節點無法挑選來擴展路徑時，便會直接隨機挑選一未選節點，來作為下一指標節點，建構過程將重覆圖 10 (b)的動作，直到初始建構解完成，如圖 10 (c)。



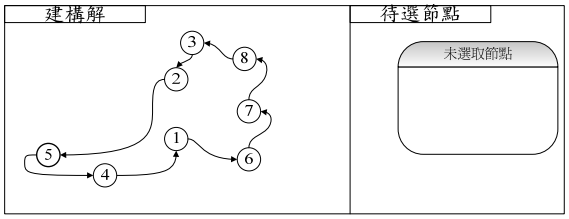
(a) 在未選取節點中隨機選擇一節點為指標節點



(b) 利用指標節點找出下一個接續節點-情況1



(b) 隨機找出下一個接續節點-情況2



(c) 重覆(b)步驟，直到完成初始解建構

圖 10　參考較佳邊之隨機建構法運作範例

本論文之初始族群的數量百分比 n_1、n_2、n_3，依序為較佳邊優先之貪婪建構法、較佳邊優先之隨機建構法與參考較佳邊之隨機建構法，$n_1+n_2+n_3=1$。

4.3　全域搜尋：基因演算法

本節將針對本論文所應用的基因演算法中相關機制作一詳細說明。以下將先介紹 TSP 問題路徑解(tour)在基因演算法中的表達法，再來依本研究的基因演算法中各相關機制被應用的順序來作相關介紹，依序為：4.3.1 染色體編碼。4.3.2 交配機制。4.3.3 突變機制。4.3.4 選擇機制。

4.3.1　染色體編碼(chromosome coding)

在基因演算法中，TSP 問題的路徑解我們可簡單利用路徑節點的連結順序來表達為一條完整染色體(chromosome)。

4.3.2　交配機制(crossover)

交配在自然界中族群的演化中扮演著結合與傳承的重要角色，在人工系統的基因演算法中也是如此。本論文在交配機制上採用了兩種方法，除了一般熟知的順序交配法(order crossover ; OX crossover)之外，另外為應用較佳邊集合來加速交配優質化的參考較佳邊之順序交配法(PEOX crossover)，我們將在下面的小節中依序介紹。

4.3.2.1　順序交配法(OX crossover)

順序交配法是較為傳統的交配法則，父代利用亂數產生的單切點(cut point)將自身的基因序列一分為二且保留切點前段給子代直接繼承，子代再利用交互參照父代的基因特徵重組成一個完整的基因序列，在交互參照的重組階段，如遇到不能被選擇的基因時，則會一直嘗試下一順序的基因是否可選擇。

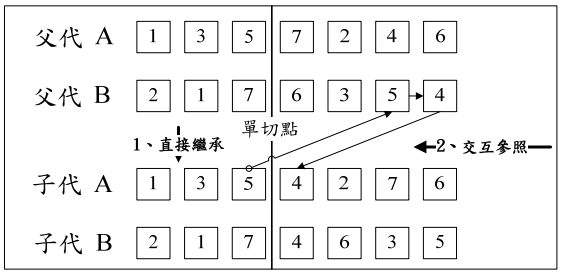


圖 11　順序交配法運作範例

以下我們利用一個簡單例子來說明順序交配法的運作過程，見圖 11，在例子中，我們產生一隨機切點位置，由切點位置將父代 A 與父代 B 的基因序列一分為二，子代直接繼承對應父代前半段，如子代 A 繼承父代 A 的基因序列(1,3,5)，子代 B 繼承父代 B 的基因序列(2,1,7)，而子代的後半段基因序列將利用另一個父代的基因序列為參考來挑選，如子代 A 前半段的最後一個基因為 5，此時將從父代 B 基因序列中為 5 的基因位置往下一基因尋

找，遇到未被選取基因即可加入子代 A 的基因序列中，如遇到已選過基因，像子代 A 在挑選基因 2 之後的基因時，會遇到已選過之基因 1，此時便會跳過基因 1 而往下一個基因尋找，也就是基因 7。再來經由上述過程不斷的重覆，直到所有子代都完成完整的基因序列組合為止，才算完成一次的交配運作。

4.3.2.2 參考較佳邊之順序交配法(PEOX crossover)

參考較佳邊之順序交配法是結合較佳邊集合以增進順序交配法之子代獲得較多優良基因特徵所設計的方法，它與順序交配法相似，但在交互參照的重組階段，如果遇到不能選擇的基因時，將先參考較佳邊集合並最多二次任意選擇一個可能較佳的基因作為重組基因的選擇，以增加較多優良基因特徵的獲得，但如果依舊挑到不能選擇的基因，就會直接嘗試下一序列的基因是否可選擇且不斷嘗試直到出現可選基因之後，才會再重新啟用參考較佳邊的選擇規則。

以下我們利用一個簡單例子來說明參考較佳邊之順序交配法的運作過程，見圖 12，在例子中，我們產生一隨機切點位置，由切點位置將父代 A 與父代 B 的基因序列一分為二，子代直接繼承對應父代前半段，如子代 A 繼承父代 A 的基因序列(1,3,5)，子代 B 繼承父代 B 的基因序列(2,1,7)，而子代的後半段基因序列將利用另一個父代的基因序列為參考來挑選，如子代 A 前半段的最後一個基因為 5，此時將從父代 B 基因序列中為 5 的基因位置往下一基因尋找，遇到未被選取基因即可加入子代 A 的基因序列中，如遇到已選過基因，像子代 A 在挑選基因 2 之後的基因時，會遇到已選過之基因 1，此時會在較佳邊集合中最多二次隨機挑選一個與節點 2 相關的候選節點，在例子中是挑選到邊(6,2)，所以找到的基因為基因 6，而基因 6 也未被選過，因此能將基因 6 放入子代 A 中，此時會從基因 6 再往後尋找未選過基因。可是如未能在候選節點中選到可用的節點，則將直接從剛才遇到已選基因的位置不斷往後尋找，直到取得一個可選取的基因為止，才會重新啟用參考較佳邊的選擇規則。再來經由上述過程不斷的重覆，直到所有子代都完成完整的基因序列組合為止，才算完成一次的交配運作。

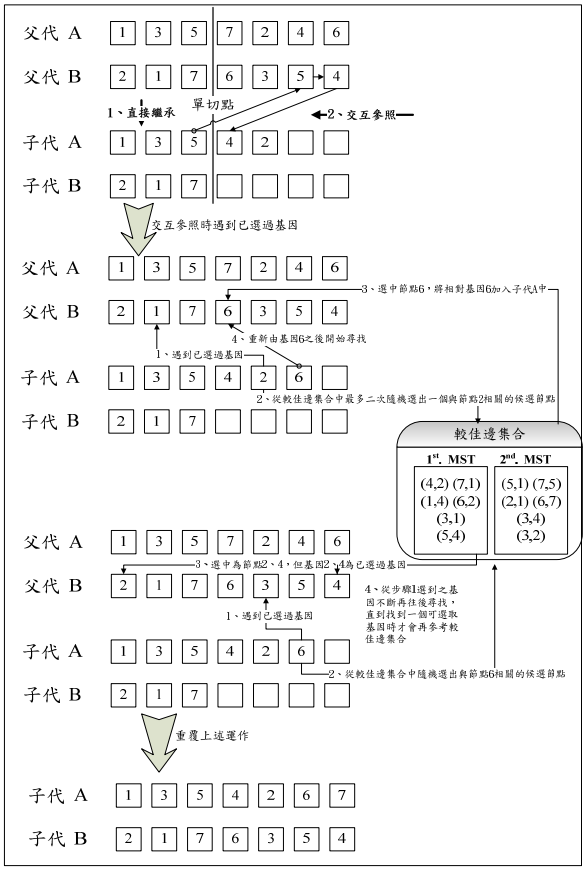


圖 12　參考較佳邊之順序交配法運作範例

最後本論文為了不讓交配機制的收斂效果太過強烈，將採用較高機率 r_1 使用順序交配法與較低機率 r_2 使用參考較佳邊之順序交配法，其中 $r_1 + r_2 = 1$。

4.3.3 突變機制(mutation)

突變在演化過程中是扮演著重生的機制，突變發生的時間與程度將決定演化過程中是否能跳脫自身束縛並成就新的開始。而本論文的突變機制直接引用ANGEL[30]的突變機制設計，即利用每個基因之間相互親疏關係來決定是否保留該段的基因特徵，換句話說就是依每個節點的鄰近關係 x_1 來作為決定路徑解邊是否被保留。然而在路徑解邊可依鄰近關係被保留時，仍有一定機率 x_2 被刪除。

以下我們利用一簡單例子來清楚說明整個突變機制的作法，見圖 13，先利用節點鄰近表的資訊，將挑選出來的染色體序列依突變規則將可能較差的基因特徵刪去，如例子中刪去的(6,1)、(7,2)、(4,6)，再隨機將各個基因片段重新組合起來，如例子中(1,3)與(2,4)重組成(1,3,2,4) 基因片段與(6)與(5,7)重組成(6,5,7)基因片段，直到將所有的片段完成組合，便完成一次突變機制的運作。

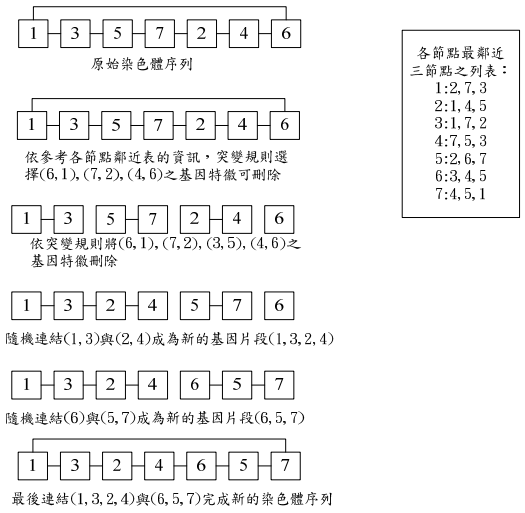


圖 13　突變機制運作範例

本研究的突變機制其啟動條件是在於新生的路徑解無法優於現行最佳的路徑解並利用區域搜尋也無法突破自身的劣勢時，將以一定機率 x_3 進行突變。

4.3.4　選擇機制(selection)

選擇機制是為了新一代演化所作的篩選動作，藉由保留較佳的染色體使得下一世代能有更強的競爭優勢。本論文在完成每一代的基因交配與突變動作後，所產生的子代數量與父代數量相同，為此我們必須從中挑選出較佳的染色體，也就是路徑解，使染色體數量回歸於演化開始的規模大小，以免存在過多的太差染色體而使得演化力量被分散。

本研究為了能強化每一代的演化動力，採用的選擇策略是保留三種不同特性的族群：優勢族群、一般族群與新生族群。顧名思義，優勢族群是指在完成一代的交配演化後，最佳且不重覆之百分比 g_1 數量的優良染色體，而一般族群是指除了優勢族群外，任意挑選出百分比 g_2 數量的染色體而形成，新生族群則是依百分比 g_3 的數量重新產生的新染色體，本研究希望透過保留這三種族群的作用下，能帶給下一世代足夠的演化動力與優勢。此外，新生族群的產生是應用較佳邊優先之隨機建構法與參考較佳邊之隨機建構法兩者比率各半來產生的。

4.4　區域搜尋：LKH 演算法

本論文為了讓較佳邊集合能發揮最大的效用，因此採用LKH[17]演算法作為本研究的區域搜尋法。本研究的區域搜尋法與原始LKH不同之處是，利用了最佳邊集合來取代原本LKH的候選邊集合(candidate set)，使本研究的基因區域搜尋法能達到整體搜尋的一致性，進而整合出更強的搜尋力量。

本研究之區域搜尋法的設定是採用循序 5-opt 與非循序 4-opt 作為節點順序重排機制的換邊規則，以有效利用 LKH 重覆修正的特性來獲得更好的路徑解改善效果。特別說明的是，循序 5-opt 是指換邊次數最大為 5 次，而非每次 5-opt 運作都會完全換掉 5 條邊。

以下我們簡略地利用 LKH 運作過程來說明何謂 LKH 重覆修正的特性。基本上，在 LKH 開始執行節點順序重排機制時，LKH 會依序以路徑解的每個節點作為嘗試節點順序重排動作的出發點，在以該節點作完一次節點順序重排動作後，如能成功改進路徑解的話，該節點未來將會被重新執行節點順序重排動作，如圖 14 所示。此外 LKH 的節點順序重排機制依換邊規則的不同，分成循序節點順序重排與非循序節點順序重排二種獨立的運作階段，每個運作階段都是一次完整的節點順序重排機制運作，在 LKH 運作的一開始先會進入循序節點順序重排階段，在循序節點順序重排階段完成之後，才接著運作非循序節點順序重排階段，此時如果在非循序節點順序重排階段裡能改善路徑解的話，LKH 將又會回到循序節點順序重排階段的運作，否則才會停止此次 LKH 的運作。所以 LKH 乃是藉由上述不斷重覆的節點順序重排機制來達到極佳的路徑解改善效果，因此本研究也利用 LKH 此一特性來使得較佳邊集合能發揮最大效益。

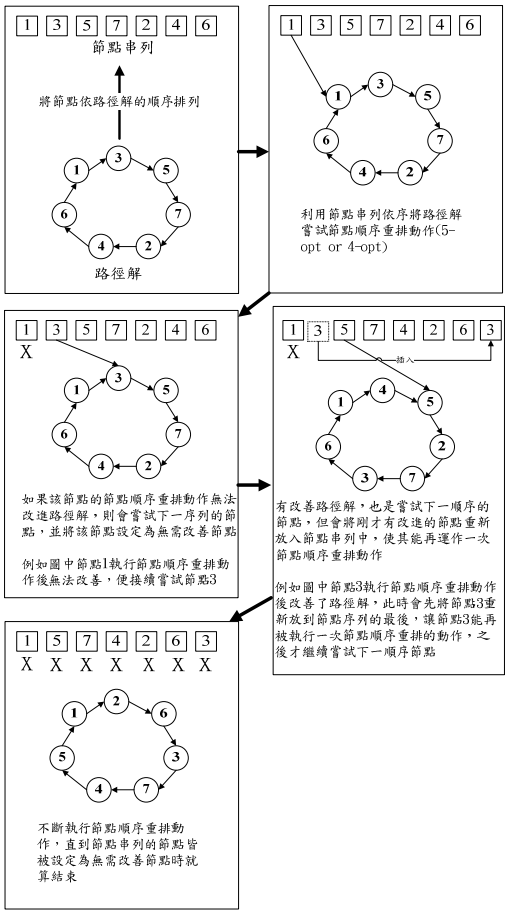


圖 14　LKH 之節點順序重排機制運作過程

4.5　較佳邊集合的進退場機制

較佳邊集合在本研究中代表著較小而有效的搜尋範圍，而這個有效的搜尋範圍在初始時卻難以涵蓋所有最佳解邊，藉由第三節中表 2 的數據可看出，雖然應用有限次數的取邊動作來取得大部分的最佳解邊，可是仍會有遺漏的最佳解邊未能在有限次數裡被收集到。因此如何收集一開始遺漏的最佳解邊，來充實較佳邊集合的可靠度是十分重要的。

在此為使加入集合的邊能保持一定品質的前提之下，我們藉由以下兩種條件的進場機制來收集可能的較佳邊：

1. 突破本演算法到目前所找到之最佳解的新路徑解。
2. 基因演化過程中每隔 t_1 代，最佳且不重覆之百分比 t_2 數量的路徑解。

所以只要符合上述任一條件的路徑解都會加入較佳邊集合中，並會依加入情況的不同來調整原本較佳邊子集合的序列。所謂加入情況不同是指加入的邊為新邊或是已存在邊。如果為新邊的話，將會放置於該相關節點的較佳邊子集合裡的第一順位，其餘的較佳邊依原本順序退後一位。如果是已存在邊

，便會提升一次該相關節點的較佳邊子集合裡的原本順位，只要升到第一順位後，便不會再被提升。特別注意的是，新邊的放置優先權大於已存在邊的提升權。

另一方面，較佳邊集合在經過多次增邊動作之後，必定容易存在過多不必要的邊而拖延搜尋的時間。因此為了能保持較佳邊集合的數量，我們設定以下條件的退場機制來刪去集合中較無貢獻的較佳邊：

1. 基因演化過程中每隔 t_3 代，在這 t_3 代之間皆無任何貢獻度的較佳邊。

而為了減少可能發生的誤刪機率，要設法讓集合裡的較佳邊都能被充分檢驗其貢獻度，因此 t_3 的設定就要能充份保留適當的檢驗緩衝時間。

4.6　強化搜尋架構的設定

為了提升本研究搜尋架構的求解效能，以下我們針對搜尋效能與突破區域最佳解等二方面作一簡單的補強設定。

在搜尋效能方面，為了避免較佳邊集合在初始建構階段取得過多不必要的邊使得搜尋速度遲緩。在產生初始路徑解時，只針對當中最好的路徑解執行較佳邊集合的進場機制，而非所有的初始路徑解都會執行。

而突破區域最佳解的設定是如果連續 u_1 代無法改進時，在該代執行完選擇機制之後，會依據以下二個條件來作變動後，才會進入下一代的演化過程：

1. 如果較佳邊集合總量小於 u_2 倍的問題規模時，會將初始時的較佳邊集合加入現在的較佳邊集合。
2. 除了保留最好且不重覆之百分比 u_3 數量的路徑解，剩餘各半皆由較佳邊優先

之隨機建構法與參考較佳邊之隨機建構法重新產生。

五、實驗結果與討論

本研究已將前述所提出之演算法利用C語言實作成單CPU核心運行的程式，測試平台為個人電腦，其備配為Intel Core2 Due E6320 1.86GHz CPU、4GB主記憶體，作業系統：Windows XP、並針對TSPLIB[28]之35題進行實驗。

5.1 實驗參數

實驗參數這一小節將條列出本研究架構中所有的參數設定：

1. 初始族群數量(initial population size)：數量 m=50。
2. 較佳邊集合(promising edge set ; PES)：最小生成樹取邊次數 p=3。
3. 初始解建構法：
   A. 較佳邊優先之貪婪建構法：初始族群數量百分比 n_1=20%
      i. 優先嘗試較佳邊子集合裡排序第一的候選節點機率 k_1=70%。
      ii. 隨機嘗試較佳邊子集合裡候選節點機率 k_2=30%。
   B. 較佳邊優先之隨機建構法：初始族群數量百分比 n_2=40%，其餘設定與較佳邊優先之貪婪建構法相同。
   C. 參考較佳邊之隨機建構法：初始族群數量百分比 n_3=40%
      i. 隨機選擇一個位於較佳邊子集合裡的候選節點機率 k_3=80%。
      ii. 隨機選擇一個未被選節點機率 k_4=20%。
4. 交配機制(crossover)：
   A. 採用順序交配法機率 r_1=90%。
   B. 採用參考較佳邊之順序交配法機率 r_2=10%。
5. 突變機制(mutation)：
   A. 節點鄰近關係 x_1=3。
   B. 節點邊有鄰近關係但依舊刪除機率 x_2=20%。
   C. 突變機制發動機率 x_3=10%。
6. 選擇機制(selection)：
   A. 優勢族群的數量比例 g_1=20%。
   B. 一般族群的數量比例 g_2=40%。
   C. 新生族群的數量比例 g_3=40%。
7. 較佳邊集合進退場機制：
   A. 進場機制：
      i. 進場代數 t_1=4。
      ii. 最佳且不重覆的路徑解數量比例 t_2=10%。
   B. 退場機制：
      i. 退場代數 t_3=8。
8. 強化搜尋架構的設定：
   A. 突破區域最佳解：
      i. 持續未改善代數 u_1=4。
      ii. 較佳邊集合總量小於問題規模倍數 u_2=1.6。
      iii. 保留最好且不重覆的路徑解數量比例 u_3=20%。

5.2 實驗結果

本節將針對本研究所提出的由較佳邊集合引導之基因區域搜尋法(Promising Edge Set Guided Genetic Local Search Algorithm ; PESGLS)進行實驗，本實驗分成兩部分，第一為測試PESGLS是否比LKH演算法更具求解能力，第二為實驗PESGLS應用於旅行推銷員問題(Traveling Salesman Problem ; TSP)求解品質與效能。

在表3中，我們利用較大規模的TSP問題5題，於本研究測試平台下執行PESGLS與LKH，由於PESGLS與LKH的代數設定機制不同，無法簡單以執行一代的結果作為比較基礎，因此改由以PESGLS執行一代的時間為基準，LKH也以同等時間執行後再比較其結果。以下依序說明表3中各欄

表 3.　PESGLS 與 LKH 測試 TSBLIB 大題組的實驗結果

		PESGLS			LKH		
Instance	Opt Length	Tour Length	%Opt	CPU Time	Tour Length	%PEGLS	Block Time
rl5915	565530	565942	0.073	31	566727	0.139	23
rl5934	556045	556793	0.134	56	556922	0.023	46
pla7397	23260728	23264544	0.016	79	23265576	0.004	70
rl11849	923288	924408	0.121	94	924584	0.019	88
usa13509	19982885	19990187	0.037	206	19991703	0.008	202

位代表的意義，Instance 欄位是代表 TSPLIB 中的題目代號。Opt Length 欄位是指該題目的最佳解路徑長度。PESGLS 欄位下的 Tour Length 欄位是在 PGLES 執行完一代後最好的路徑解長度，%Opt 欄位為 PESGLS 的 Tour Length 超過 Opt Length 的百分比，%Opt 正數代表 PESGLS 的 Tour Length 大於 Opt Length。CPU Time 欄位為 PESGLS 執行完一代的時間。LKH 欄位下的 Tour Length 欄位是 LKH 在執行 PESGLS 一代的時間內最好的路徑解長度。%PESGLS 欄位為 LKH 的 Tour Length 超過 PESGLS 的 Tour Length 的百分比，%PESGLS 正數代表 LKH 的 Tour Length 大於 PESGLS 的 Tour Length。Block Time 欄位是指 LKH 在 PESGLS 執行一代的時間內最好的路徑解出現的時間。表 3 的時間單位以秒為單位。

從表 3 中發現 PESGLS 與 LKH 在同等的時間求解限制下，PESGLS 在測試的 5 題中能皆表現出較佳的效果，從 LKH 的 Block Time，我們也可發現 LKH 容易掉入區域最佳解而難以跳脫，而 PESGLS 雖然應用 LKH 作為區域搜尋法，但結合全域搜尋的機制與較佳邊集合的優點，使得 PESGLS 表現出較好的求解能力。

表 4 是利用 TSPLIB 問題規模大於 500 且小於 4000 的題目 30 題作為評比求解效能與計算時間的標準，除了 PESGLS 與 LKH[17] 的實驗結果比較之外，在此我們也與目前品質與效能比較佳的

表 4.　針對 TSPLIB 題目規模大於 500 小於 4000 的題組，應用 PESGLS、ANGEL 與 LKH 之實驗結果

		Success			Gap_{avg}			Gap_{max}			$Time_{min}$			$Time_{avg}$		
Instance	n	ANGEL	LKH	PEGLS	ANGEL	LKH	PEGLS	ANGEL	LKH	PEGLS	ANGEL	LKH	PEGLS	ANGEL	LKH	PEGLS
att532	532	100/100	98/100	100/100	0	0.001	0	0	0.072	0	0	0.1	0	0	1.2	0
si535	535	100/100	33/100	100/100	0	0.006	0	0	0.017	0	0.01	1.7	0.8	3.37	10.0	4.3
pa561	561	100/100	99/100	100/100	0	0.001	0	0	0.072	0	0	0.1	0	1.07	1.2	0.73
u574	574	100/100	91/100	100/100	0	0.007	0	0	0.081	0	0	0.1	0	0.56	1.1	0.61
rat575	575	100/100	77/100	100/100	0	0.004	0	0	0.03	0	1	0.1	0.07	2.2	2.7	0.54
p654	654	100/100	100/100	100/100	0	0	0	0	0	0	0.07	0.4	0.21	0.29	3.2	0.91
d657	657	100/100	100/100	100/100	0	0	0	0	0	0	0	0.1	0.06	0	1.1	0.2
gr666	666	100/100	30/100	100/100	0	0.026	0	0	0.04	0	2	0.7	0.5	2	6.3	2.7
u724	724	100/100	98/100	100/100	0	0	0	0	0.005	0	0.12	0.2	0.2	3.19	2.3	1.63
rat783	783	100/100	100/100	100/100	0	0	0	0	0	0	0.05	0.0	0	0.42	0.1	0.21
dsj1000	1000	10/10	7/10	10/10	0	0.035	0	0	0.116	0	3	4.4	1.62	6.3	18.4	3.9
pr1002	1002	10/10	10/10	10/10	0	0	0	0	0	0	1	0.3	0.7	2.4	1.1	1.83
si1032	1032	10/10	2/10	10/10	0	0.057	0	0	0.071	0	0.45	1.1	0.74	0.3	6.7	2.67
u1060	1060	10/10	9/10	10/10	0	0	0	0	0.003	0	3	0.5	0.6	4	12.7	3.82
vm1084	1084	10/10	7/10	10/10	0	0.007	0	0	0.022	0	3	0.6	1.91	23.3	9.5	7.1
pcb1173	1173	10/10	8/10	10/10	0	0.002	0	0	0.009	0	15	0.2	0.5	18.1	4.9	4.1
d1291	1291	10/10	8/10	10/10	0	0.033	0	0	0.167	0	0.36	7.7	4.7	3.7	19.9	11.5
rl1304	1304	10/10	8/10	10/10	0	0.019	0	0	0.161	0	0.3	0.4	1.4	3	11.9	8.61
rl1323	1323	10/10	1/10	10/10	0	0.018	0	0	0.048	0	0.5	3.6	2.6	13.1	17.2	15.8
nrw1379	1379	10/10	3/10	10/10	0	0.006	0	0	0.009	0	0.7	6.7	0.2	23.4	23.1	5.79
fl1400	1400	10/10	1/10	10/10	0	0.162	0	0	0.199	0	0.5	4.4	6.2	14.5	194.4	24.7
u1432	1432	10/10	10/10	10/10	0	0	0	0	0	0	0.3	0.3	0.61	2.3	2.7	1.88
fl1577	1577	10/10	2/10	10/10	0	0.046	0	0	0.063	0	0.1	40.1	1.81	17.1	365.8	27.2
d1655	1655	10/10	10/10	10/10	0	0	0	0	0	0	0.4	3.5	1.3	40.8	13.7	14.3
vm1748	1748	10/10	4/10	10/10	0	0.023	0	0	0.054	0	8	2.4	2.1	24.5	338.7	19.3
u1817	1817	10/10	2/10	10/10	0	0.078	0	0	0.124	0	15	66.5	9.8	135.8	84.3	56.9
rl1889	1889	10/10	4/10	10/10	0	0.002	0	0	0.004	0	1.5	0.4	3.8	16.2	37.9	28.9
pr2392	2392	10/10	10/10	10/10	0	0	0	0	0	0	2.3	2.9	1.81	25.1	18.2	12.9
pcb3038	3038	10/10	9/10	10/10	0	0	0	0	0.004	0	49	13.0	27.2	1125.8	107.9	88.7
fl3795	3795	10/10	4/10	10/10	0	0.072	0	0	0.191	0	51.78	816.3	108	748.5	1888.8	976.8

ANGEL[30]演算法之實驗數據作一比較，藉由目前表現較好的演算法來有效評估PESGLS。PESGLS於本研究測試平台下執行，ANGEL與LKH運作測試平台為Intel Pentium 2.0 GHz CPU和 1GB主記憶體，雖然兩者之測試平台略有差異，但利用本研究測試平台上測試LKH求解所得的實驗數據與ANGEL報告中LKH的實驗數據進行比較，發現兩者時間的差異不到 5%，因此本研究的實驗結果並不作任何的效能差異修正。以下依序說明表 4 中各欄位代表的意義，Instance欄位是代表TSPLIB中的題目代號。n欄位是指該題目的規模大小。Success欄位下的分母代表解題的總次數，分子則代表成功的次數。Gap_{avg}與Gap_{max}欄位是指最佳路徑解與最佳解之間差距佔最佳解長度百分比於總解題次數下的平均值與最大值，上述最佳路徑解是指完成一次題目求解結束時最好的路徑解。$Time_{min}$與$Time_{avg}$欄位是指在總解題次數中單次求解達到停止條件的最短時間和各次求解達到停止條件的平均時間。表 4 的時間單位以秒為單位。

從表 4 結果中可發現，針對測試題組而言，PESGLS 的求解成功次數已可達百分之百，與ANGEL 的求解能力相當，而 LKH 則有相當多次無法求解成功，由此可證實 PESGLS 的求解穩定度較佳於 LKH。在求解效率的比較結果中，PESGLS 針對測試題組幾乎皆能比 LKH 的求解效率結果來得更好。而與 ANGEL 相比之下則是互有優劣，於是本研究也針對PESGLS表現較差的題組作了分析，發現 PESGLS 對於在初始階段有較多最佳解邊沒被取得的題目，在效能表現上有較多需要努力的空間，不過相對地只須透過適量的廣域搜尋的題目，PESGLS 的效能表現就較為優異。由上述分析的結果彰顯出較佳邊集合之進退場機制的有效性，將足以左右 PESGLS 的求解效率。不過總體而論，PESGLS在應用於求解TSP問題的效能與品質表現仍是相當不錯的。

六、結論與未來研究方向

本論文所提出之由較佳邊集合引導之基因區域搜尋法，在求解旅行推銷員問題(Traveling Salesman Problem ; TSP)上得到不錯的成效。經由實驗結果證明，較佳邊集合導引之基因區域搜尋法能夠更集中搜尋的力量以提升求解速度，此外也藉由較佳邊集合有效連結了全域搜尋與區域搜尋的合作機制，使其兩者相輔相成達到更完善的搜尋效果與求解品質。

然而本論文須要改善的部分是求解時間的方面，在實驗結果中我們可發現對於不少一開始較佳邊集合較差的題目，其解題時間與ANGEL[30]相比仍花費較多的搜尋時間。因此我們針對本研究架構中能加強效能的部分，提出以下四點作為未來研究方向：

1. 建構出更好的進場機制，以免容易因一開始設定的門檻限制過高而使得後續的收集邊的困難度增加。因為本研究架構主要依賴目前找到的最佳解被突破時執行收邊的進場機制，收到的邊雖然可能較佳，但也容易收不到可能是不錯的邊。
2. 設計較為精準的退場機制，以減少非最佳解的邊進入較佳邊集合中存在過久，而增加無效的搜尋時間。
3. 分析設計能預測較佳邊落點的機制，來加速找到必須強化的搜尋範圍。
4. 組合不同特性的區域搜尋法或啟發式演算法，來改善單一搜尋法可能產生的盲點。

相信未來研究如能強化上述四點，必能提升本研究之基因區域搜尋法求解 TSP 問題的效率。

最後，我們未來也將嘗試應用本研究之基因區域搜尋法於求解排程問題(Scheduling Problem)、車輛巡迴問題(Vehicle Routing Problems)等 TSP 問題的延伸題型上，因為透過本研究的實驗結論，我們更有信心能將本研究的求解經驗與設計機制轉移到上述相似性質的問題，期待最後能提供上述各類問題一個更有效且更穩定的求解機制。

致謝

本論文之研究經費由國科會在計畫編號 NSC96-2628-E-005-074-MY3 下部分補助，作者在此特別致上感謝。

七、參考文獻

[1] E. H. L. Aarts and H. P. Stehouwer, "Neural networks and the traveling salesman problem," Proceedings of the International Conference on Artificial Neural Networks, Springer-Verlag, 1993, pp. 950-955.

[2] J. R. A. Allwright, and D. B. Capenter, "A distributed implementation of simulated annealing for the traveling salesman problem," Parallel Computing, vol. 10, pp. 335-338, 1989.

[3] R. Baraglia, J. I. Hidalgo, and R. Perego, "A hybrid heuristic for the traveling salesman problem," IEEE Transactions on Evolutionary Computation, vol. 5, no. 6, pp. 613-622, December 2001.

[4] F. Bock, ‘‘An algorithm for solving traveling-salesman and related network optimization problems,’’ unpublished manuscript associated with talk presented at the 14th ORSA National Meeting, 1958.

[5] M. Barrie, Baker, and M.A. Ayechew, "A genetic algorithm for the vehicle rounting problem," Computer and Operational Research., Vol. 30, pp. 787-800, 2003.

[6] F. Croce, R. Tadei, and G. Volta, "A genetic algorithm for the job shop problem," Computers and Operational Research., Vol. 22, pp. 15-24, 1995.

[7] N. Christofides, ‘‘Worst-case analysis of a new heuristic for the traveling salesman problem,’’ Report No. 388, GSIA, Carnegie-Mellon University, Pittsburgh, PA, 1976.

[8] G. Clarke AND J. W. Wright, ‘‘Scheduling of vehicles from a central depot to a number of delivery points,’’ Operations Research, vol. 12, pp. 568-581, 1964.

[9] G. A. Croes, ‘‘A method for solving traveling salesman problems,’’ Operations Research , vol. 6, pp. 791-812, 1958.

[10] M. Dorigo, L. M. Gambardella, ‘‘Ant colony system: A cooperative learning approach to the traveling salesman problem, ’’ IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 53-66, April 1997.

[11] R. Durbin, R. Szeliski, and A. Yuille, "An analysis of the elastic net approach to the traveling salesman problem," Neural Computation, vol. 1, pp. 348-358, 1989.

[12] L. Fiechter, "A parallel Tabu search algorithm for large traveling salesman problems," Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science, vol. 51, pp. 243-267, 1994.

[13] B. Freisleben and P. Merz, ‘‘A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems,’’ Proceedings of the 1996 IEEE International Conference on Evolutionary Computation, 1996.

[14] B. Freisleben and M. Schulte. "Combinatorial optimization with parallel adaptive threshold accepting". Proceedings of the 1992 European Workshop on Parallel Computing, Barcelona, IOS Press, 1992, pp. 176-179.

[15] L. M. Gambardella and M. Dorigo, "Ant-Q: A reinforcement learning approach to the traveling salesman problem," Proceedings of the Twelfth International Conference on Machine Learning, 1995, pp. 252-260.

[16] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979.

[17] K. Helsgaun, "An effective implementation of the Lin–Kernighan traveling salesman heuristic," European Journal of Operations Research, vol. 126, pp. 106-130, 2000.

[18] J.H. Holland, *Adaptation in Nature and Artificial Systems*, University of Michigan Press. 1975.

[19] L. Homaifar, C. Guan, and G. Liepins, "A new approach to the traveling salesman problem by genetic algorithms," Proceedings of the 5th International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, 1993, pp. 460-466.

[20] O. Jellouli, E. Chatelet, "A dynamic programming approach to sequencing problems," SLAM Review.10, pp.196-210, 2000.

[21] D. S. Johnson, L. A. McGeoch, "The traveling salesman problem: A case study in local optimization," in *Local Search in Combinatorial Optimization*, E. H. L. Aarts and J. K. Lenstra, Eds. , New York: Wiley: New York, 1997.

[22] E. Lawler, J. K. Lenstra and D. B. Shmoys, *The Traveling Salesman Problems: A Guided Tour of Combinatorial Optimization*. Wiley, 1985.

[23] S. Lin, ''Computer solutions of the traveling salesman problem,'' Bell System Technical Journal, vol. 44, pp. 2245-2269, 1965.

[24] S. Lin , B. W. Kernighan, ''An effective heuristic algorithm for the traveling-salesman problem,'' Operations Research, vol. 21, pp. 498-516, 1973.

[25] P. Merz and B. Freisleben, ''Genetic local search for the TSP: New results,'' Proceedings of the 1997 IEEE International Conference on Evolutionary Computation, 1997.

[26] J. W. Pepper, B. L. Golden, and E. A. Wasil, "Solving the traveling salesman problem with annealing-based heuristics: a computational study," IEEE Transactions on Systems, Man, and Cybernetics, Part A, vol. 32, no. 1, pp. 72-77, January 2002.

[27] J. Perttunen, "On the significance of the initial solution in traveling salesman heuristics," Journal of the Operational Research Society, vol. 45, pp. 1131-1140, 1994.

[28] G. Reinelt, "TSPLIB – A traveling salesman problem library," ORSA Journal of Computing, vol. 3, no. 4, pp. 376-384, 1991.

[29] T. Starkweather, D. Whitley, C. Whitley, K. Mathial, "A comparison of genetic sequencing operators," Proceedings of the 4th International Conference on Genetic Algorithm, Morgan Kaufmann, 1991, pp. 69-76.

[30] L. Y. Tseng and S. C. Liang, "A Hybrid Metaheuristic and Its Application to the Traveling Salesman Problem," Submitted to IEEE Transactions on Evolutionary Computation. (Revised) (SCI), 2007.

[31] T. Volgenant, R. Jonker , "A branch and bound algorithm for the symmetric traveling salesmane problem based on the 1-tree relaxation," European Journal of Operational Research.9, pp. 83-89,1982.