# 導引式遺傳演算方法解決旅行銷貨員問題之研究

# Solving the Traveling Salesman Problem through a Genetic Algorithm Approach with Guided Information

江傳文、潘邦積
國立高雄第一科技大學電腦與通訊工程系
ccw@ccms.nkfust.edu.tw

## 中文摘要

本文中，我們提出了一個可以有效解決旅行銷貨員問題的導引式遺傳演算方法。我們同時也分別設計了一個交配與突變的運算子，用以平衡本文所提出演算法之開發與探索的能力。我們以 TSPLIB 國際題庫中的問題案例為效能量測的工具。實驗結果顯示，本文所提出之方法在解題效能上有非常顯著且優異的表現。

**關鍵詞**：旅行銷貨員問題；遺傳演算法；組合最佳化；

## Abstract

In this study, the well-known Traveling Salesman Problem (TSP) is solved by using a novel genetic algorithm approach with guided information. Two sophisticated crossover and mutation operators are introduced to explicitly balance the exploration and exploitation abilities of the proposed algorithm. The performance of the proposed algorithm is evaluated by comparing it against existing techniques in terms of overall schedule length for a set of problem instances obtained form the traveling salesman problem library (TSPLIB). Experimental results indicate that the proposed algorithm is a significant improvement compared with other approaches.

**Keywords**：Traveling Salesman Problem; Genetic Algorithms; Combinatorial Optimization;

## 一、簡介

旅行銷貨員問題 (Traveling Salesman Problem，TSP) 係組合最佳化研究領域中一個非常著名的經典問題 [9]。其可被描述為：如何以最小成本方式在恰好繞經所有城市一次之後並回到原出發點。由於此一問題的定義簡單易懂，因此常被用來作為測試演算法效能的工具。此外，用以解決旅行銷貨員問題的技術具有廣泛的應用層面。舉凡生物科學領域的基因序列比對、應用於物流管理的車輛途程決策、電腦網路連線之配線佈置、工廠之生產排程作業以及電子產業的電路板設計等問題，皆可藉由直接或間接方式引用旅行銷貨員問題的解決方法求解之 [3]。因此，發展可以有效解決旅行銷貨員問題的技術便成為極具研究價值的重要議題。

一般而言，解決旅行銷貨員問題的技術可概分為確切演算法 (Exact algorithms) 以及近似演算法 (Approximation algorithms) 兩種類型。確切演算法包括窮舉搜尋 (exhaustive search)、分枝界限 (branch and bound)、動態規劃 (dynamic programming) 以及切割平面 (cutting plane) 等技術。此一類型技術的最主要特色在於保證可以得到所求解問題案例的最佳解。然而，由於旅行銷貨員問題的一般

形式已被證實為是 NP-完備 (NP-Complete) [12]，因此對於大規模問題案例之求解，使用確切演算法將因為需要耗用大量的 CPU 計算時間而顯得不切實際。有鑑於此，一般吾人皆以近似演算法為解題工具，試圖在合理的時間之內獲得令人滿意的近似解。

近似演算法基本上包含了傳統的啟發式與泛用啟發式 (meta-heuristic) 兩種技術。其中，傳統的啟發式技術又可分為以下三種類型：

- 途程建構法 (Tour Construction)：根據頂點與頂點間的距離成本建構一條路徑。例如：最近鄰點法 (Nearest Neighbor Procedure)、插入法 (Insertion)、節省法 (Savings Method)、貪婪法 (Greedy Algorithm)、最小擴張樹法 (Minimal Spanning Tree Approach) 等。
- 途程改進法 (Tour Improvement)：將一條路徑以鄰域搜尋的方式進行改善。例如：K-opt 節線交換法、Or-opt 節線交換法以及 Lin & Kernighan 節線交換法等。
- 綜合法 (Composite Procedure)：混合途程建構法以及途程改進法。

以上所提及之傳統啟發式技術，其最主要的缺點在於缺乏跳脫陷入區域最佳困境的能力。因此，便有學者探討如何以泛用啟發式技術解決旅行銷貨員問題。泛用啟發式技術通常源自於對真實世界之生物習性或物理特性的觀察與模擬。例如：模擬退火 (Simulated Annealing，SA) [1]、螞蟻族群最佳化 (Ant Colony Optimization) [7-8, 14] 以及遺傳演算法 (Genetic Algorithms，GAs) [4-5, 10] 等。

模擬退火的觀念最早是由 Metropolis 等人於 1953 年所提出。一般而言，當物質處於較高溫度環境時，其內部粒子亦具有較大能量，因此物質狀態的改變具有較高的隨機性。然而隨著溫度的緩慢下降，物質的狀態亦將漸趨穩定。基於此一特性，模擬退火演算法藉由一個有效的問題解來模擬物質目前的狀態，而物質狀態的改變則是利用此一現行解的鄰近空間隨機產生另一個可行解的過程來加以表示。在每一次的迭代中，如果鄰近解的品質優於現行解，則將該鄰近解取代現行解。反之，則由機率決定是否接受該鄰近解為現行解。此一演算法利用一個溫度參數控制迭代的進行，該參數的值與接受機率的大小成正比。隨著溫度的逐次遞減，整個搜尋過程在迭代之初有較高的機率可跳脫區域最佳困境，而迭代後期則可有效地保留較佳品質的解。此一迭代過程稱為退火程序。模擬退火雖然是一種可用以解決組合最佳化問題的方法，然而由於其求解的過程係以一個解在某一溫度的系統狀態下進行擾動搜尋，因此可搜尋的範圍較小且屬於單點式的搜尋。

另一方面，基於對自然界真實螞蟻覓食行為的觀察，Marco Dorigo 等學者則是於 90 年代初期提出著名的螞蟻系統 (Ant System，AS) [8]。在真實世界中，螞蟻之間會利用一種名為費洛蒙的化學物質相互溝通，進而以協同合作方式找出巢穴與食物之間的最短路徑。藉由此一行為模式的模擬，螞蟻系統成功地解決了著名的旅行銷貨員問題。螞蟻系統與傳統泛用啟發式方法在設計上的最主要不同之處在於：此一系統的設計精神著重在建構問題解的過程。更進一步地說，人工螞蟻係藉由學習以往建構解之過程所獲得的經驗，以協同合作方式達成改善目前所獲得解之品質的目的。基於此一設計概念，近年來陸續有許多文獻提出新的螞蟻族群最佳化演算法 [7, 14]。由於螞蟻族群最佳化演算法係以逐步建構方式產生問題解，因此對於大規模旅行銷貨員問題案例之求解，將耗費過多的計算時間。此外，人工螞蟻在問題解空間的搜尋決策行為會同時受到費洛蒙與啟發函式的影響，但二者間之相對權重不易決定。

除了模擬退火與螞蟻族群最佳化之外，遺傳演算法也是一種被廣泛用於解決組合最佳化問題的泛用啟發式技術。遺傳演算法係由 John Holland 於 1975 年所

提出 [5]。此一演算法係植基於達爾文「物競天擇」演化理論的概念所發展而成，其試圖藉由模擬自然界物種演化的過程以搜尋問題領域的狀態空間，進而獲得問題的解答。遺傳演算法首先會將吾人所欲求解之問題以字串的方式編碼成為所謂的「染色體 (chromosome)」。每個染色體中包含了數個基因 (gene)，不同的基因排列表現出此一染色體的不同特徵，而這些特徵在經過評估 (evaluation) 之後便會反映出該染色體對所在環境的適存程度 (fitness)，適存值越高的染色體表示其所相對應解的品質越高。遺傳演算法一般具有「交配 (crossover)」、「突變 (mutation)」以及「選擇 (selection)」等三個運算子。交配運算子的主要目的在於將不同染色體的部份基因予以交換，進而使得相對應解的優良特性可以被保留至下一個迭代；突變運算子則是隨機改變染色體中部份基因的內容，用以在搜尋過程中跳脫局部最佳解的陷阱；至於選擇運算子則是用以決定哪一個染色體可以存活至下一個迭代。對於環境的適存程度越高的染色體而言，其在演化過程中得以存活的機率當然也就越高。如此反覆運作許多迭代之後，我們預期將可以獲致高品質的問題解。本文中，我們將以遺傳演算法的架構為基礎，提出全新的交配與突變運算子，用以更有效解決旅行銷貨員問題。此外，為了改善遺傳演算法容易「早熟 (過早出現搜尋停滯的現象)」的缺點，我們則是藉由限制最佳染色體的繁衍數量，以達到維持族群多樣性的目的。

本文在後續的內容中，首先會在第二節針對所探討的旅行銷貨員問題加以描述且定義。其次，我們將在第三節中就國內外過去的相關研究做概要的介紹。接著我們會在第四節詳細說明本文中所提出方法的設計細節。至於評估演算法效能的實驗結果則是在第五節中呈現。最後，我們將在第六節為本文內容進行總結。

## 二、問題塑模

一般而言，吾人可利用一個無向圖 $G = \{V, E\}$ 來描述旅行銷貨員問題。在此圖形中，$V = \{v_1, v_2, ..., v_{|V|}\}$ 為所有節點的集合。其中，符號 $|V|$ 表示所有節點的數目。此外，集合 $E = \{e_{ij}\}$ 中之無向邊 $e_{ij}$ 係用以表示節點 $v_i$ 與節點 $v_j$ 之間的成本。基於此一表示方法，旅行銷貨員問題可被定義為找出一條最小成本的周遊路線 (tour)，此一路線係以任一節點 $v_i$ 為出發點，繞行圖形中所有的節點恰好一次，最後再回到節點 $v_i$。根據以上的定義，我們可以得到以下的目標函式 [6]：

$$Minimize \sum_{i=1}^{|V|}\sum_{j=1}^{|V|} c_{ij}x_{ij} \qquad (1)$$

限制式：

$$\sum_{i=1}^{|V|} x_{ij} = 1, (j = 1,...,|V|) \qquad (2)$$

$$\sum_{j=1}^{|V|} x_{ij} = 1, (i = 1,...,|V|) \qquad (3)$$

$$x_{ij} = 0 \text{ or } 1, \forall (i, j) \in E \qquad (4)$$

其中，$c_{ij}$ 表示銷貨員在行經路徑 $(i, j)$ 時所付出的成本。此外，若銷貨員曾行經路徑 $(i, j)$，則 $x_{ij}$ 的內容值為 1；反之，則為 0。

## 三、相關研究

### (一) 遺傳演算法的運作流程

一般而言，適當的問題編碼係吾人利用遺傳演算法解決組合最佳化問題的首要步驟。針對不同特性的問題，其編碼方式亦多所不同。當編碼方式決定之後，遺傳演算法接著會建立一個由數個初始染色體所組成的族群。通常，以隨機的方式產生初始解是最直接的方法。一旦族群建構完

成，吾人便可以從族群中隨機選出兩個染色體，並利用交配程序產生新的子代。除了交配程序之外，另一種可用以產生新的子代的操作為突變程序。最常見的突變操作是從染色體中以隨機的方式選取一個基因並改變其內容或所在位置。在經過了交配和突變的過程之後，吾人便可藉由評估每個染色體的適存度以決定該染色體是否得以繼續存活而進入下一迭代。我們可以使用隨機抽樣 (stochastic sampling)、決定性抽樣 (deterministic sampling) 或是混合以上兩種方法來選擇組成下一迭代族群的染色體成員。遺傳演算法便是藉由重複執行交配、突變、評估、選擇等四個程序試圖找出欲求解問題的最佳解。

(二) 常見的交配運算子

遺傳演算法之交配運算子的目的在於保留親代染色體中好的基因特徵，進而使得族群個體朝向解空間中的最佳方向搜尋。基本上，解決旅行銷貨員問題的交配運算子可概分為以下三種類型：

- 保留基因區段資訊的交配運算子。例如：順序交配運算子 (order crossover，OX) 以及局部對應交配運算子 (partially-mapped crossover，PMX)。
- 保留基因位置資訊的交配運算子。例如：循環交配運算子 (cycle crossover，CX)。
- 保留邊之資訊的交配運算子。例如：啟發式交配運算子 (heuristic crossover，HX) [11]。

OX 的作法是先以隨機的方式從族群中選取 $C_{p1}$ 和 $C_{p2}$ 兩個染色體並決定切點 (cut-off point) 的位置，然後將 $C_{p1}$ 中位於切點左側所有的基因複製並加入加入到子代 $C_{offspring}$ 之中，最後再將 $C_{p2}$ 中尚未在 $C_{offspring}$ 出現的基因依序加入至 $C_{offspring}$。

PMX 則是先以隨機的方式從族群中選取 $C_{p1}$ 和 $C_{p2}$ 兩個染色體並決定切點的位置，接著找出 $C_{p1}$ 與 $C_{p2}$ 在切點右側所有基因的對應關係，然後再根據此對應關係交換 $C_{p1}$ 中的基因位置，最後所得到的結果即為新產生的子代 $C_{offspring}$。

至於 CX 則是先以隨機的方式從族群中選取 $C_{p1}$ 和 $C_{p2}$ 兩個染色體。以 $C_{p1}$ 的第一個基因為出發點，找出在兩個父染色體中相對位置基因的一個循環，然後將此循環中屬於 $C_{p1}$ 的基因複製到子代 $C_{offspring}$ 的相對位置。之後，再以 $C_{p2}$ 中未包含在上一循環中的第一個基因為出發點，以同樣的方式找出第二個循環，然後將此循環中屬於 $C_{p2}$ 的基因複製到子代 $C_{offspring}$ 的相對位置。如此交互替換直到 $C_{offspring}$ 的所有基因均被填滿為止。

除此之外，HX (heuristic crossover) 則是一種以建構方式為基礎的交配運算子。其基本作法係以隨機方式決定一個起始頂點，然後再根據事先定義的規則選取一條可行進的邊並走訪至下一個頂點，如此反覆進行直到建構出一個完整的解為止。由於在走訪頂點的過程中，必須避免循環 (cycle) 的形成，因此 HX 採用以下程序解決此一問題：

步驟 1. 隨機選取一個頂點 $u$ 為目標頂點。

步驟 2. 根據親代染色體的基因內容，將所有與目標頂點相連且不會在子代染色體中構成循環的邊組合成為一個候選邊的集合 $S$。

步驟 3. 基於以下規則，由集合 $S$ 中選出一個可行進的邊：

- 若 $S$ 為空集合，則根據最鄰近串列 (nearest-neighbor list，NNL) 的順序，選出一條具有最短距離且不會在子代染色體中構成循環的邊。
- 若 $S$ 中只有一個候選邊，則直接選取之。
- 若 $S$ 中具有數個候選邊，則根據以下規則決定之：
  - 若集合 $S$ 中恰好存在兩個相同的候選邊，則以該候選邊為選取目標；反之，則以具有最

短距離的候選邊為選取目標。

步驟 4. 根據步驟 3 所選出之可行進的邊，走訪至下一個頂點 $v$，並將頂點 $v$ 視為是新的目標頂點。

步驟 5. 重複執行步驟 2 直到建構出一個完整的解為止。

(三) 常見的突變運算子

遺傳演算法之突變運算子的目的在於使族群個體維持適度的多樣性、重新拾回遺失的基因資訊以及跳脫陷入區域最佳的困境。最常見用以解決旅行銷貨員問題的突變運算子包括互換突變 (swap mutation) 與反向突變 (inversion mutation) 等運算子。

假設目前擬進行突變運算的染色體 $s$ 的基因內容為 1-2-3-4-5-6-7-8-9-1。互換突變運算子的運作過程是先以隨機方式選取染色體 $s$ 中的兩個基因 (假設分別為 2 與 6)，然後交換這兩個基因的內容。因此，經互換突變後之染色體 $s'$ 的基因內容為 1-6-3-4-5-2-7-8-9-1。另一方面，反向突變運算子的運作過程也是先以隨機方式選取染色體 $s'$ 中的兩個基因 (假設分別為 2 與 6)，然後再將包含基因 2、6 的基因區段內容予以反轉。因此，經反向突變後之染色體 $s''$ 的基因內容為 1-2-5-4-3-6-7-8-9-1。

互換突變運算子與反向突變運算子的優點是作法簡單，不需要考慮突變後所得的問題解是否合法。然而其缺點是互換突變運算子執行一次突變只能改變 4 個邊，而反向突變運算子一次突變只改變 2 個邊。此對於使用以保留邊之資訊的運算子而言，跳離的解空間距離並不遠。而且，以隨機選點的方式進行突變，將使突變後的解很可能變的很差而不易存活到下一迭代。

(四) 常見的選擇機制

遺傳演算法中常見的選擇機制包括輪盤選擇 (Roulette Wheel Selection)、競爭式選擇 (Tournament Selection) 以及評等選擇 (Ranking Selection)。選擇機制對於遺傳演算法的求解效能有著舉足輕重的影響。一個設計不當的選擇機制將會使族群中的個體很快地喪失多樣性，進而造成搜尋停滯的情況發生。

輪盤選擇方式係根據每一個染色體之適存值在整個族群中所佔的比重來決定該染色體的存活率。令 $|P|$ 為族群大小、$f_i$ 與 $r_i$ 分別為族群中第 $i$ 個個體的適存值與存活率，則 $r_i$ 可被定義如下

$$r_i = f_i \times (\sum_{j=1}^{|P|} f_j)^{-1} \qquad (5)$$

輪盤選擇機制的優點在於具有公平性，具有越大適存值的個體其存活率便越高。然而，若族群中存在一個個體具有比其它所有個體大很多的適存值，則將造成其它個體的存活率相對降低，導致族群中之個體的多樣性就此消失。

為了避免此一情況的發生，吾人可以改用競爭式選擇機制，以決定族群中之個體何者能存活至下一迭代。此一機制的基本運作方式是由族群中以隨機方式選取數個個體，然後再從這些個體中找出具有最大適存值的染色體為選取目標。競爭式選擇機制的實作非常簡單，但也存在著一個嚴重的問題。那便是：族群中最差的個體將永遠不會被選取。

除了輪盤選擇機制與競爭式選擇機制之外，評等選擇機制也是一個常用的方法。基本上，評等選擇機制與輪盤選擇機制類似，都是一種以機率方式進行選擇的機制。唯一不同的是，評等選擇機制係以適存值的排名來決定個體的存活率。適存值排名越前面的個體，其存活率便越高。此一機制的優點在於可避免超級強者的出現；但其缺點則是在於將適存值的影響間接化，使個體間適存值的差值變得不再重要。

除了輪盤選擇機制與競爭式選擇機制之外，評等選擇機制也是一個常用的方

法。基本上，評等選擇機制與輪盤選擇機制類似，都是一種以機率方式進行選擇的機制。唯一不同的是，評等選擇機制係以適存值的排名來決定個體的存活率。適存值排名越前面的個體，其存活率便越高。此一機制的優點在於可避免超級強者的出現；但其缺點則是在於將適存值的影響間接化，使個體間適存值的差值變得不再重要。

## 四、設計方法

(一) 編碼

針對一個具有 $n$ 個城市的旅行銷貨員問題，本文係以一個元素個數為 $n + 1$ 的一維串列來表示相對應於該問題的解。若某一問題解 $s$ 的基因內容如下：

$$\{c_1, c_2, \ldots, c_i, \ldots, c_n, c_{n+1}\} \qquad (6)$$

則表示銷貨員係由城市 $c_1$ 出發，依序經過 $c_2$、$c_3$、 ... 、$c_i$、 ... 、$c_n$、$c_{n+1}$ 等城市。由於銷貨員最後必須回到原出發點，此表示 $c_1 = c_{n+1}$。令符號 $d_{c_i, c_{i+1}}$ 表示第 $i$ 個位置之城市與第 $i+1$ 個位置之城市間的距離，則問題解 $s$ 的排程長度 $l_s$ 可利用以下算式計算而得

$$l_s = \sum_{i=1}^{n} d_{c_i, c_{i+1}} \qquad (7)$$

於是，問題解 $s$ 的適存值 $f_s$ 便可被定義如下：

$$f_s = \frac{1}{l_s} \qquad (8)$$

(二) 交配

有鑑於自然界中某些具有社會組織活動的生物 (例如：螞蟻、蜜蜂等)，其領導者往往具有交配的主導權。因此在本文所提出的演算法中，我們定義了一個名為交配優勢的參數，並藉由以下程序決定出每次參與交配運算的兩個親代：

步驟 1. 以隨機方式產生一個介於 [0,1) 之間的數值 $c$。

步驟 2. 若數值 $c$ 小於等於事先訂定的交配優勢，則以輪盤選擇方式自交配池中選取一個基因內容與全域最佳染色體 $gb$ 相異的個體，並使之與 $gb$ 進行交配；反之，則以輪盤選擇方式自交配池中選取兩個染色體進行交配。

當決定出參與交配運算的兩個親代染色體之後，便可進行以下的交配程序：

步驟 1. 隨機選取一個頂點 $u$ 為目標頂點。

步驟 2. 根據親代染色體的基因內容，將所有與目標頂點相連且不會在子代染色體中構成循環的邊組合成為一個候選邊的集合 $S$。

步驟 3. 執行以下步驟，由集合 $S$ 中選出一個可行進的邊：

步驟 3.1. 若 $|S| \le 1$，則利用一機率值決定是否根據導引表格 (guided table) 的順序，選出一條不會在子代染色體中構成循環的邊，並將此一可行進的邊加入集合 $S$ 中。

步驟 3.2. 若 $S$ 為空集合，則根據最鄰近串列 (nearest-neighbor list，NNL) 的順序，選出一條具有最短距離且不會在子代染色體中構成循環的邊。

步驟 3.3. 若 $S$ 不為空集合，則根據所有候選邊之距離資訊計算出一個門檻值 $t_s$。此一門檻值 $t_s$ 被定義為所有候選邊之距離的標準差除以所有候選

邊之平均距離所得的結果。此外，產生一個介於 [0,1) 之間的數值 $r$，並比較 $r$ 與 $t_s$ 之間的關係。若 $r < t_s$，則以具有最短距離之候選邊為選取目標；反之，則以輪盤選擇方式自集合 $S$ 中選出一個可行進的邊。

步驟 4. 根據步驟 3 所選出之可行進的邊，走訪至下一個頂點 $v$，並將頂點 $v$ 視為是新的目標頂點。

步驟 5. 重複執行步驟 2 直到建構出一個完整的解為止。

(三) 突變

對於每一個經由交配程序所產生之子代染色體，其皆有一定的機率 (突變率) 會進行突變運算。在傳統的遺傳演算法中，突變運算子往往只是簡單地針對現階段所得到的問題解進行擾動，因此常會導致經擾動後所得到新問題解的品質不佳而無法存活至下一迭代，進而造成計算資源的浪費。因此，本文所提出之演算法便利用一個大小為 $|V|\times|V|$ 且名為導引表格 (guided table) 的二維矩陣記錄染色體經突變後所得之結果，用以作為後續進行交配與突變運算時的參考，進而導引演算法往問題解空間中的正確方向進行搜尋。

此一導引表格中第 $u$ 列第 $v$ 行元素的初始值被設定為頂點 $u$ 與頂點 $v$ 彼此間之距離的倒數。導引表格中之元素值的更新方式為將突變後所得解之改善值與該解的總長度之比例值依邊的貢獻度分配累加於導引表格中的相關元素。以下步驟染色體 $s$ 執行突變程序之過程：

步驟 1. 隨機選擇一個頂點 $u$。

步驟 2. 根據導引表格中第 $u$ 列的內容順序，選取出另一個頂點 $v$。在此，$u$ 與 $v$ 在染色體 $s$ 中不得為彼此相鄰的基因。

步驟 3. 判斷 $u$ 與 $v$ 在擾動後的兩種情形：

- 將 $v$ 移至 $u$ 的右邊
- 將 $u$ 移至 $v$ 的右邊

選擇上述兩種情形解路徑總長最短的一種加到候選清單中。

步驟 4. 若候選清單尚未滿，則執行步驟 2。

步驟 5. 將候選清單中依權重用輪盤選取一個頂點。在此，權重表示擾動後的改善值。

步驟 6. 更新導引表格的內容值。

步驟 7. 若尚未到達突變長度，則將頂點 $u$ 取代所選擇的頂點 $v$ 並繼續執行步驟 2。

(四) 選擇

本階段的主要工作在於決定哪些染色體可繼續存活至下一迭代。由於傳統的遺傳演算法在選擇階段中往往只根據每一個染色體的適存值作為判定該染色體能否繼續存活至下一迭代的唯一參考，因此導致族群中之染色體的基因內容在極短時間內便會趨於一致，於是交配運算子的功能減弱，演算法亦呈現搜尋停滯的現象。

為了改善此一缺點，我們利用一個介於 [0,1) 之間的全域最佳門檻值 $t_{gb}$ 為管控工具，用以限制所有優於目前全域最佳染色體之個體繼續存活至下一迭代的數量。此一選擇階段之執行程序如下：

步驟 1. 將迭代最佳的染色體複製到下一迭代。

步驟 2. 令目前迭代中適存值大於目前全域最佳染色體之個體總數為 $m$。若原始族群大小 $|P|$ 與 $t_{gb}$ 之乘積大於 $m$，則根據適存值大小依序複製 $m$ 個染色體到下一迭代；反之，則讓前 $|P| \times t_{gb}$ 個具有較大適存值的染色體繼續存活。

步驟 3. 以目前迭代中所有適存值小於全域最佳染色體之個體為對象，藉由輪盤選擇方式反覆選取可繼續存活的染色體，直到下一迭代的

族群大小回復至 $|P|$ 為止。

由於許多最佳解的基因片段極可能存在於適存值較差的個體中，因此確保族群的多樣性將可以有較大的機會把最佳解的基因保留在族群中。本文所提出之選擇機制一方面可以避免輪盤選擇方式中過好的基因佔據整個族群而減少多樣性，另一方面又可以維持族群中較好的染色體的主導性，因此可以平衡演算法之開發與探索的能力。

## 五、實驗結果

為了評估本文所提出之演算方法的效能，我們由 TSPLIB [2] 國際題庫中選出了 25 個案例作為測試樣本。針對每一個測試案例，我們皆以本文所提出的演算方法進行 10 次的求解實驗。演算法在執行時所採用的族群大小為 100、交配率為 0.9、突變率為 0.1。此外，當全域最佳解經過連續 200 迭代後皆未改善便終止演算法的執行。我們以 Java 程式語言為實作演算方法的工具。所有的實驗均在 Intel Pentium 4 個人電腦上執行。此一實驗平台的 CPU 時脈為 2.6 GHz、記憶體大小為 2 GB、作業系統為 Microsoft Windows XP Professional Pack 2。

我們首先針對各種演算法在效能上的表現進行全面的比較。針對每一個測試案例，我們以平均誤差率 *Error* 為效能比較的指標 [13]。此一指標被定義為

$$Error = \frac{AVG - OPT}{OPT} \times 100\% \qquad (9)$$

其中，*AVG* 表示該案例在經過 10 次求解實驗後所得之平均解的品質，而 *OPT* 則是該案例目前已知的最佳排程長度。表 1 所列示之內容即為各演算法在求解所有測試案例時所得到的平均誤差率。實驗結果顯示，本文所提出之演算方法在解題效能上的表現確實優於其它傳統的方法。

除了在效能上的全面比較之外，我們也針對交配運算子的效果進行觀察。表 2 所列示之數據係本文所提出之交配運算子與 HX 運算子在解題效能上的表現。顯而易見地，本文所提出的交配運算子具有較佳的解題效能。其主要原因在於 HX 運算子在選取可行進邊的過程中，係以親代染色體中較好的邊為選取目標，一旦演算法陷入區域最佳，將無法順利跳脫此一困境。反觀本文所提出之方法，其不僅利用導引表格與最鄰近串列的資訊增加候選邊集合的多樣性，同時也將所有候選邊之距離的標準差納入考慮，使得演算法有較大機會可以跳脫陷入區域最佳的困境。

基本上，本文所提出之交配運算子由候選邊集合中選取可行進邊的方式有以下三種策略：

- 選取候選邊集合中具有最短距離的邊。
- 以每一個候選邊的距離為基礎，藉由輪盤選擇方式選取之。
- 混合上述所提及之兩種方法。

表 3 所列示之內容係各種選取策略對演算法在解題效能上之影響的比較。由實驗結果觀之，使用最短距離選取策略所得之演算法效能與 HX 相近，這是因為 HX 運算子的選邊方式也是以最短距離邊為基礎。另一方面，使用機率模式選取策略所得之演算法效能比使用最短距離選取策略為佳的主要原因在於前者有較大機會可以避免陷入區域最佳的困境。至於混合模式則是因為結合了最短距離選取策略與機率模式選取策略二者的優點，因此具有最佳的效能。

## 六、結論

本文中，基於遺傳演算方法的基本原理，我們提出了一個可用以有效解決旅行銷貨員問題的演算法。相較於其它以泛用啟發式技術所發展而成的演算法，本文所提出之方法具有以下幾點特色：

- 在選擇機制中加入門檻值，確保族群中

之個體的多樣性。

- 在交配程序中加入參考導引表格的機制，使交配程序兼具開發與探索的能力。
- 將突變運算後所得之結果記錄於導引表格中，使得導引表格之內容可作為後續交配與突變運算時的指引。

## 七、參考文獻


[1] E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*, John Wiley & Sons, 1989.

[2] G. Reinelt, "TSPLIB-A Traveling salesman problem library," *ORSA Journal on computing*, Vol. 3, pp. 376-384, 1991.

[3] G. Gutin and A. P. Punnen, *The Traveling Salesman Problem and Its Variations*, Springer, 2002.

[4] H. -K. Tsai, J. -M. Yang, Y. -F. Tsai, and C. -Y. Kao, "An Evolutionary Algorithm for Large Traveling Salesman Problems," *IEEE Trans. on Systems, Man and Cybernetics: Part B Cybernetics*, Vol. 34, No. 4, pp. 1718-1729, August 2004.

[5] J. H. Holland, *Adaptation in Natural and Artificial Systems*, Univ. of Michigan Press, Ann Arbor, 1975.

[6] L. Bodin, B. L. Golden, A. Assad, and M. Ball, "Routing and Scheduling of Vehicle and Crew: The State of Art," *Special Issue of Computers and Operations Research*, Vol. 10, No. 2, pp. 63-211, 1983.

[7] M. Dorigo and L.M. Gambardella, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem," *IEEE Trans. on Evoluationary Computation*, Vol. 1, No. 1, pp. 53-66, April 1997.

[8] M. Dorigo, V. Maniezzo and A. Colorni, "Ant System: Optimization by a Colony of Cooperating Agents," *IEEE Trans. on Systems, Man and Cybernetics, Part B*, Vol. 26, No. 1, pp. 29 - 41, February 1996.

[9] M. M. FLOOD, "The Travelling Salesman Problem," *Operations Research*, Vol.4, No. 1, pp. 61-75, 1956.

[10] M. Srinivas and L. M. Patnaik, "Genetic Algorithms: A Survey," *IEEE Computer*, pp. 17-26, June 1994.

[11] Q. Hu, H. Wu, Q. Chen, and Y. Chen, "Study on a New Heuristic Crossover for the Traveling Salesman Problem," *Chinese Control Conference*, pp. 1442-1447, 2006.

[12] R. Karp, "Reducibility among Combinatorial Problem," *Complexity of Computer Computations*, Plenum Press, pp. 85-104, 1972.

[13] S. T. Mahdi, K. Morteza, Mohammad-R. A. -T and Hamid, A. M.,"A Novel Constructive-Optimizer Neural Network for the Traveling Salesman Problem," *IEEE Trans. on Systems, Man and Cybernetics: Part B Cybernetics*, Vol. 37, No. 4, pp. 745-770, August 2007.

[14] T. Stutzle and H. H. Hoos, "MAX-MIN Ant System," *Journal of Future Generation Computer Systems*, Vol. 16, No.8, pp. 889-914, 2000.

表 1. 各演算法求解所有測試案例之平均誤差率

| Problem Instance | OPT | This work | CONN | SA2 | KD | KL | KG | Budinich's SOM | ESOM | eISOM | Co-adaptive Net | MVHN 2-optimal |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Eil51 | 426 | 0.45 | 2.6 | 2.3 | 3.5 | 2.9 | 2.9 | 3.1 | 2.1 | 2.6 | 2.9 | 7 |
| St70 | 675 | 0.46 | 3 | 2.3 | 3.7 | 1.5 | 2.3 | 1.7 | 2.1 | | 1.7 | 7.8 |
| Eil76 | 538 | 0.19 | 5 | 5.5 | 6.5 | 5 | 5.5 | 5.3 | 3.9 | | 4.4 | 9.4 |
| Rd100 | 7910 | 0.38 | 3.6 | 3.3 | 4.9 | 2.1 | 2.6 | 3.2 | 2 | | 3.6 | 10 |
| Kroa100 | 21281 | 0.18 | 2.6 | 5.9 | | | | 3.7 | 1 | 0.6 | 1.3 | 13.9 |
| Eil101 | 629 | 1.34 | 5.1 | 5.7 | 6.8 | 4.7 | 5.6 | 5.2 | 3.4 | 3.6 | 3.8 | 10.1 |
| Lin105 | 14379 | 0.11 | 0.38 | 1.9 | 2.2 | 2 | 1.3 | 1.7 | 0.25 | | 1.1 | 8.3 |
| Pr107 | 44303 | 0.41 | 2.8 | 1.5 | 10.8 | 0.73 | 0.42 | 1.3 | 1.5 | | 4.4 | 6.1 |
| Pr124 | 59030 | 1.01 | 1.7 | 1.3 | 3.2 | 0.08 | 0.49 | 1.6 | 0.67 | | 2.9 | 6.1 |
| Bier127 | 118282 | 0.69 | 2.5 | 3.5 | 5.8 | 2.8 | 3.1 | 3.6 | 1.7 | | 3 | 9.1 |
| Pr136 | 96772 | 0.48 | 2.3 | 4.9 | 1.9 | 4.5 | 5.2 | 5.2 | 4.3 | | 4.7 | 9.9 |
| Kroa150 | 26524 | 0.84 | 3.5 | 4.3 | | | | 4.4 | 2 | 1.8 | 3.1 | |
| Pr152 | 73682 | 0.21 | 0.79 | 2.6 | 3.2 | 0.97 | 1.3 | 2 | 0.89 | | 2.1 | 5.6 |
| Rat195 | 2323 | 0.87 | 5.6 | 13.3 | 8.4 | 12.2 | 11.9 | 11.5 | 7.13 | | 7.5 | 12.2 |
| Kroa200 | 29368 | 0.91 | 5.7 | 5.6 | 5.7 | 5.7 | 6.6 | 6.1 | 2.9 | 1.6 | 3.3 | |
| Lin318 | 42029 | 1.51 | 7.3 | 7.6 | | | | 8.2 | 4.1 | 2.1 | 4.3 | |
| Pcb442 | 50778 | 2.05 | 5.8 | 9.2 | 8 | 11.1 | 10.4 | 8.4 | 7.4 | 6.1 | 7.6 | 12.8 |
| p654 | 34643 | 1.70 | 4.1 | | 5 | | 5.6 | | | | 4.6 | |
| rat783 | 8806 | 2.47 | 7.8 | | 9.1 | | 9.6 | | | | 6.6 | |
| pr1002 | 259045 | 3.04 | 7.6 | 6 | 7.1 | | 7.6 | 8.8 | 6.4 | 4.8 | 5.3 | |
| pcb1173 | 56892 | 3.05 | 9.2 | 11.1 | 12.7 | | | 11.4 | 9.9 | | 9.6 | |
| d1291 | 50801 | 2.66 | 11.3 | | 16.4 | | | | | | 12 | |
| rl1323 | 270191 | 2.31 | 10.9 | | 13.8 | | | | | | 11.8 | |
| fl1400 | 20127 | 2.40 | 4.1 | 4.7 | 6 | | | 5.6 | 4.1 | | 4.3 | |
| u1432 | 152970 | 3.61 | 6.6 | 2.3 | 9 | 2.9 | 2.9 | 3.1 | 2.1 | 2.6 | 6.3 | 7 |
| Average | | 1.33 | 4.87 | 5.13 | 6.99 | 4.02 | 4.85 | 5.10 | 3.39 | 2.90 | 4.89 | 9.16 |

表 2. 不同的交配運算子對於演算法在解題效能上之影響

| Problem Instance | OPT | This work | HX |
|---|---|---|---|
| Eil51 | 426 | 0.45 | 0.19 |
| St70 | 675 | 0.46 | 1.01 |
| Eil76 | 538 | 0.19 | 0.87 |
| Rd100 | 7910 | 0.38 | 0.47 |
| Kroa100 | 21281 | 0.18 | 0.58 |
| Eil101 | 629 | 1.34 | 1.86 |
| Lin105 | 14379 | 0.11 | 0.23 |
| Pr107 | 44303 | 0.41 | 0.36 |
| Pr124 | 59030 | 1.01 | 0.95 |
| Bier127 | 118282 | 0.69 | 0.88 |
| Pr136 | 96772 | 0.48 | 1.67 |
| Kroa150 | 26524 | 0.84 | 1.66 |
| Pr152 | 73682 | 0.21 | 0.33 |
| Rat195 | 2323 | 0.87 | 1.23 |
| Kroa200 | 29368 | 0.91 | 0.92 |
| Lin318 | 42029 | 1.51 | 2.31 |
| Pcb442 | 50778 | 2.05 | 2.57 |
| p654 | 34643 | 1.70 | 3.70 |
| rat783 | 8806 | 2.47 | 4.03 |
| pr1002 | 259045 | 3.04 | 3.85 |
| pcb1173 | 56892 | 3.05 | 4.51 |
| d1291 | 50801 | 2.66 | 4.61 |
| rl1323 | 270191 | 2.31 | 4.95 |
| fl1400 | 20127 | 2.40 | 4.80 |
| u1432 | 152970 | 3.61 | 4.92 |
| Average | | 1.33 | 2.24 |

表 3. 不同的選邊模式對於演算法在解題效能上之影響

| Problem Instance | OPT | 混合模式 | 最短距離 | 機率模式 |
|---|---|---|---|---|
| Eil51 | 426 | 0.45 | 0.56 | 0.59 |
| St70 | 675 | 0.46 | 0.95 | 1.16 |
| Eil76 | 538 | 0.19 | 0.58 | 1.21 |
| Rd100 | 7910 | 0.38 | 1.31 | 1.38 |
| Kroa100 | 21281 | 0.18 | 0.64 | 1.35 |
| Eil101 | 629 | 1.34 | 2.48 | 2.16 |
| Lin105 | 14379 | 0.11 | 0.64 | 0.96 |
| Pr107 | 44303 | 0.41 | 0.39 | 0.75 |
| Pr124 | 59030 | 1.01 | 0.56 | 0.72 |
| Bier127 | 118282 | 0.69 | 1.04 | 1.61 |
| Pr136 | 96772 | 0.48 | 3.02 | 2.13 |
| Kroa150 | 26524 | 0.84 | 2.41 | 1.59 |
| Pr152 | 73682 | 0.21 | 0.49 | 0.63 |
| Rat195 | 2323 | 0.87 | 1.55 | 2.01 |
| Kroa200 | 29368 | 0.91 | 1.71 | 1.66 |
| Lin318 | 42029 | 1.51 | 2.78 | 2.80 |
| Pcb442 | 50778 | 2.05 | 2.77 | 2.55 |
| p654 | 34643 | 1.70 | 1.85 | 3.08 |
| rat783 | 8806 | 2.47 | 3.41 | 3.64 |
| pr1002 | 259045 | 3.04 | 4.27 | 4.19 |
| pcb1173 | 56892 | 3.05 | 4.80 | 4.37 |
| d1291 | 50801 | 2.66 | 3.34 | 2.95 |
| rl1323 | 270191 | 2.31 | 4.47 | 2.78 |
| fl1400 | 20127 | 2.40 | 3.94 | 3.81 |
| u1432 | 152970 | 3.61 | 4.80 | 4.16 |
| Average | | 1.33 | 2.19 | 2.17 |