利用基因演算法解決設施定位問題

李怡慧　　洪瑞文　　曾國鈞　　郭凱銘
嶺東科技大學資訊管理系
講師　　學生
{sanity, harng}@mail.ltu.edu.tw　　{nk32325959,a77228ford}@yahoo.com.tw

摘要

設施定位問題在作業研究領域應用地相當的廣泛，多數研究採用一些限制規則將問題簡化，然後透過傳統數學上的線性規劃之類的解法，但由於設施定位問題為 NP-hard，此類解法的困難度及花費時間相當高。而 Meta-heuristic 乃是在啟發式方法上，加入一些用亂數產生解的過程，相當適合用來解決具有 NP 性質的問題。因此，本研究透過 Meta-heuristic 中應用最廣的基因演算法來求得設施定位問題最佳解。

關鍵詞：基因演算法、NP-hard、Meta-heuristic、設施定位問題

1. 前言

設施定位問題(Facility Location Problem, FLP) 是要從一些位置中，選出幾個會使總成本為最低的位置來放置設施。其在作業研究領域應用地相當的廣泛，例如，貨物銷售點的設置或是汽車維護服務廠的設置地點選擇，感測節點最佳設置位置[1][2]等等。此問題為典型 NP-hard [3]問題，要以傳統數學公式來解題，將會耗費太多時間且難度非常高，於是本研究採用 Meta-heuristic 技巧來解設施定位問題。Meta-heuristic 乃是在 heuristic 方法上，加入一些用亂數產生解的過程，相當適合用來解決具有 NP 性質的問題。而基因演算法是目前最常被運用來求最佳解的技術，如 [4][5][6]。因此，本研究利用 Meta-heuristic 中應用最廣的基因演算法，來求得設施定位問題最佳解。

1.1 問題定義

設施定位問題(Facility Location Problem; FLP): 給定一個大小為 m 的設施(facility)集合 F，跟一個大小為 n 的城市(city)集合 C，每個設施 i 的固定開啟成本(fixed open cost) 為 f_i，設施 i 對城市 j 的服務成本(service cost) 為 c_{ij}，c_{ij} 代表 i 與 j 之間的距離。此問題的目的是要找到一個設施集合 F 的子集合，以及一組此子集合中之設施與所有城市之間的分配關係，使得此關係下的固定開啟成本與服務成本總合(Total_Cost)為最小。

設施定位問題有許多常見的問題變型，包括 (1) 無容量限制的設施定位問題(Uncapacitated Facility Location Problem; UFLP) [3]: 是指每個設施可以服務的城市數目(capacity)沒有限制。(2) 有容量限制的設施定位問題(Capacitated Facility Location Problem): 是指每個設施可以服務的城市數目有所限制。每個設施可以服務的城市數目又有兩種分類: 相同或是不同。(3) 設施跟城市可以是屬於相同或不同的集合。(4) 每個城市可以是確實地只與一個設施連接，或是至少與一個設施連接 [7]。(5) 每個設施的開啟成本可以是相同或是不同，可以固定或是與服務的城市數目成一個函數關係，或是完全不考慮開啟成本(即沒有開啟成本) [7]。(6) 每個城市到設施之間的服務成本可以兩者之間的距離，或是距離再乘上一個需求量(demand)。(7) 限定被選取的設施總數量為 k 個 [3]，或是最多 k 個 [7]。

本研究將解決的是第(1)類型，每個設施可以服務的城市數目沒有限制的無容量限制的設施定位問題(UFLP) ，定義如下:

假設 F 為具有 m 個設施的集合，C 為具有 n 城市的集合。令 $FCost_j$ $(1 \le j \le m)$ 為設施 j 的 open cost，$SrvCost_{ji}$ $(1 \le i \le n, 1 \le j \le m)$ 為設施 j 的到城市 i 的 service cost。我們的目標是找到一個擁有最小總成本的連線狀況，即必須滿足以下的目標函式(即適存函式):

$$\min\left(\sum_{j=1}^{m}\sum_{i=1}^{n}\mathrm{SRVCost}_{ji}x_{ji} + \sum_{j=1}^{m}\mathrm{FCost}_j \cdot y_j\right) \qquad (1)$$

subject to:

$$\sum_{j=1}^{m} x_{ji} = 1, \qquad x_{ji} \in \{0,1\}, \text{ for every } i \in C;$$

$0 \le x_{ji} \le y_j$ and $y_j \in \{0,1\}$, for every $j \in F$ and every $i \in C$;

其中:

y_j 表示設施 j 是否被開啟。$y_j = 1$ 為開啟，否則 $y_j = 0$。

x_{ji} 表示城市 i 是否與設施 j 連線。每個城市必須恰好連線到某一個設施。

本研究主要是應用基因演算法，求取設施定位問題的最佳解，除了對設施定位問題的文獻進行回顧外，更針對基因演算法相關文獻進行探討。並透過模型修正及程式模擬，得到一些結果與探討分析，並做出結論與提出後續研究的方向、建議。

本論文的其他章節架構如下: 我們在第 2 節做一些相關研究的介紹，包括基因演算法以及設施定位問題的常見解決方法。第 3 節詳述本研究如何利用基因演算法來解決設施定位問題。第 4 節為實驗與結果分析。最後一節為結論與未來展望。

2. 文獻探討

2.1 基因演算法

基因演算法(Genetic Algorithms, GAs)的基本理論由 Holland 於 1975 年首先提倡，是基於生物進化大自然選擇過程，運用在電腦上的一種最佳化搜尋機制。其出自達爾文進化論的「物競天擇，適者生存」概念作為基礎，模擬基因組合的變化，每串基因即為生物的染色體(個體)，經由選擇族群中具有較佳特性的母代，並且隨機交換基因(交配、突變)；以期望能產生出比母代更優秀之子代，再透過多次交替，從新一代的染色體中保留優良基因，而劣質染色體逐漸淘汰，最終產生適應性最佳的個體。基因演算法三個主要程序，說明如下：

(1)複製(Reproduction)

複製，廣義來看，是指生物體對自身的複製，複製出來的子代可以跟母代完全一樣，也可以有部份的不同。其目的就是讓母代的特徵保留到子代，以免隨著時間的演進，生物演化出來的長處反而消失不見了。由於複製是挑選表現較佳的個體來取代較差的個體，因此這個動作也被稱為「選擇」(selection)，而選擇數量的多寡，則是由「選擇率」來決定。

(2)交配(Crossover)

交配過程是隨機地選取所有個體中的兩個為母代，透過彼此基因的交換，進而組成另外兩個新的子代個體，藉著累積前代的優秀基因以期望能產生更優秀的子代。交配個體的多寡是由「交配率」來決定的。交配方式可為「單點交配」(One-point Crossover)，即隨機的選取一個交配點，交配點兩側的基因做互換的動作，此外，也有另外一種「均勻交配」(Uniform Crossover)方法，主要是隨機的選取母代基因來組成子代染色體。

最後，在二個母代與二個子代間，透過適應函式的評估，保留兩個最佳的個體，並填回原有母代的位置，以便完成交配的動作。

(3)突變 (mutation)

突變過程是隨機的選取一個染色體，隨機地選取突變點，加以改變物種染色體裡的基因，突變的數量交由「突變率」所控制。

2.2 FLP 相關研究探討

在多數研究都是採用一些限制或規則將問題簡化，然後透過數學上的線性規劃之類的解法來求解，常見解題技巧為 Linear Programming Branch and Bound、LP rounding、Dual、Primal-dual 以及 Filtering 等。另有一些研究則是發展 Heuristic 的演算法，如 Moses 在 [3] 提出的方法; 或是利用一些 Local Search 技術，如 Greedy Methods [7]等，甚至將上述的解法做整合，以求得近似解(Approximation Solution)。而各種方法對設施定位問題的解題結果的驗證，多數是透過比較近似率(Approximation Ratio)、時間複雜度(Time Complexity)，或是利用知名的 Benchmark 來驗證是否找到最佳解。

以下是本研究在實驗比較時，使用到的兩個利用 deterministic 演算法求 UFLP 近似解的相關研究，分別描述如下:

(1) Moses

由 Moses Charikar [3] 所提出的演算法，其做法分為兩個階段。

第一階段為初始階段，將所有設施依 open cost 進行由小到大的排序。然後開啟 open cost 最小的設施，並將所有城市連線到該設施。接著依序開啟設施，每開啟一個設施時，依序把城市一次一個連線到新開啟的設施，並進行總成本比較，若此城市改變連線並不會降低成本則不改變原本的連線，當所有城市比較完後，讓有被城市連線的設施保留開啟，沒有被城市連線的設施則關閉。

第二階段為精煉階段，即對每一個開啟的設施，考慮將連接到該設施的所有城市改連接

到其他選擇距離最短且已經開啟的設施，然後將此設施關閉。若這樣做可以減少總成本，即產生更好的解，則將此設施關閉，且將連接到該設施的所有城市改連接到其他已經開啟的設施。

(2) The net benefit heuristic, NBH

NBH 演算法[8]可以產生很不錯的解且執行效率佳，其做法分為兩個階段:

第一階段為初始階段，即每個城市連接到服務成本最少的設施，然後讓有被城市連線的設施開啟，沒有被城市連線的設施則關閉。

第二階段為精煉階段，即對每一個開啟的設施，考慮將連接到該設施的所有城市改連接到其他已經開啟的設施(選擇距離最短者)，然後將此設施關閉。若這樣做可以減少總成本，即產生更好的解，則將此設施關閉，且將連接到該設施的所有城市改連接到其他已經開啟的設施。

Meta-heuristic 是在 heuristic 方法上，加入一些用亂數產生解的方式，使得求解過程十分彈性，有很高的機會可以在短時間內求得最佳解，相當適合用來解決具有 NP 性質的問題。因此，近期已有許多研究利用 Meta-heuristic 來解決設施定位問題，例如採用 Tabu Search [8][9] 或是基因演算法[10]等等。

3. 基因演算法的設計

本研究將透過 Meta-heuristic 中應用最廣的基因演算法為主要解題方式，並參考相關文獻的做法，來設計基因演算法的各個實驗方法，以求得設施定位問題最佳解。我們亦將以 Benchmark 來驗證。

3.1 基因演算法設計

過高的突變率與交配率會使得整個群組的收斂(Intensification)過程震盪過於激烈，但過低的突變率與交配率會使得發散(Diversification)不易，造成整個群組中，每條染色體過於相似而無法找出全域的最佳解，容易陷於區域最佳解，於是在收斂與發散兩股力量中間需要找到一個平衡，才能使問題解決。

本研究如 Kratica [10] 一樣地將 UFLP 的任意一組解以一個 binary string 來表示，亦即為一組染色體。如圖 1 為 16 個設施的染色體，0 代表該設施關閉，1 則為開啟該設施。並以一般的基因演算法求解流程為處理架構如，如圖 2 [11] 所示。本研究首先讀入 OR Library [12][13]的設計的一組 benchmark。利用一個 deterministic 演算法找出基本初解，再透過亂數的方式產生共 100 組的染色體。接著利用 1.1 小節的公式(1)計算出這些解的總成本，並利用 Insertion Sort 進行排序，以加速後續的計算動作；然後由 Crossover Operator 將染色體作基因交換，產生新的染色體；接著再經由 Mutation Operator 將基因作突變，產生新的染色體；接著利用公式(1)計算出這些解的總成本，即進行適存函式的評估；若尚未達到最大限制的代數，或是沒發生連續 100 代未產生更好的解，則透過 Selection Operator 保留上一代的某些染色體到新的一代，並重覆進行 Crossover Operator 及 Mutation Operator 繼續演化出新的一代。詳細作法如下一小節所描述。

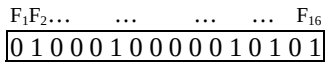

圖1 染色體編碼

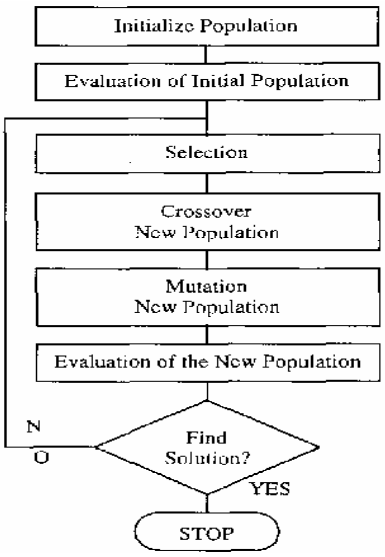

圖2求解流程圖 [11]

3.2 Operators 設計

本研究利用不同的方式來實作基因演算法的各個 Operator，且每個 Operator 都有幾種不同的實作方法。本研究將互相搭配各種 Operator 的做法，以找出可以快速求得最佳解的最佳搭配組合。Initialize Population、Selection Crossover 及 Mutation 四種 Operator 實作方法分別描述如下:

(1) Initialize Population

1. 使用亂數產生初解，本文稱為基本形式。
2. 在初始解中加入用 Moses [3]方式產生

不錯的一組染色體，其他 99 組由亂數產生。

3. 在初始解中加入用 NBH [8]方式產生不錯的一組染色體，其他 99 組由亂數產生。

(2) Selection Operator

1. Rank 選擇法: 按照排列過後的基因組挑選最佳的幾組，目的是要能夠加速收斂。
2. 隨機選擇法: 利用任選兩組染色體比較後留下較佳一組的方法，目的是要與 Rank 選擇法比較，看是否此方法較好。

(3) Crossover Operator

1. 使用隨機單點的切段交換基因資訊的方法，本文稱 R1。
2. 使用固定單點(中心點)的切段交換基因資訊的方法，目的是要與隨機單點比較收斂和發散平衡的差異，本文稱 A1。
3. 使用隨機雙點的切段交換基因資訊的方法，目的是要比較切段數不同對於收斂與發散之差異，本文稱 R2。
4. 使用固定雙點(平均分散點)的切段交換基因資訊的方法，目的是要比較隨機與固定的差異，本文稱 A2。
5. 利用 AND 對兩組染色體作運算，將兩組解中都有開啟的設施保留下來，在設計上希望能加速收斂。

(4) Mutation Operator

1. 搭配一般的 Crossover Operator 做變化。每一次作完 A1 或 A2 或 R1 或 R2 之後，10%機率產生 2 bits 突變，20% 機率產生 1 bit 突變。
2. 利用 OR 對兩組已做過 AND Crossover Operator 的染色體作運算，將兩組解中都有所有開啟的設施保留下來，再使用亂數選擇某些已開啟的設施將其關閉，此 Mutation 設計只搭配前面列出的 Crossover AND，目的是要加入發散與 Crossover AND 取得平衡。

4. 實驗結果與分析

本研究使用 Microsoft Visual C++ 6.0 實作基因演算法的各種 Operator 及整個基因演化的過程。並以 OR Library [12][13]的 benchmark 71-74、101-104 與 131-134 來驗證。每個 benchmark 會被執行 10 次，求得其(1)找到最佳解的次數(2)平均找到最佳解的代數(3)平均找到最佳解的時間。本研究利用第 3 節設計的各種 Operator 互相搭配出 10 種主要解法組合，分別描述如下: (實驗結果見表 2)

表 1: OR Library Benchmark

Problem instance	Size	File Size
41 – 44, 51, 61 – 64, 71 - 74	16 x 50	10 KB
81 – 84, 91 – 94, 101 – 104	25 x 50	15 KB
111 – 114, 121 – 124, 131 – 134	50 x 50	31 KB
A-C	100 x 1000	1.2 MB

(1) 基本形式+ Rank 選擇法

此實驗目的是要了解用基因演算法解 UFLP 的基本狀況，其作法如下：

- Initialize Population：使用亂數產生之 100 組解。
- Crossover Operator：從最佳的 90 組解，隨機挑 2 組做「隨機單點切段交換」做 50 次。
- Mutation Operator：每一次作完「隨機單點切段交換」之後，10%機率產生 2 bits 突變，20% 機率產生 1 bit 突變。
- Selection Operator：算出 100 組解各自的 總成本，再透過 Insertion Sort 挑出較佳的 90 組，並紀錄此次最好的 10 組解與其總成本，以及整體最佳的那一組解及其總成本。

實驗結果：當設施個數較少時，幾乎都可以找到最佳解，當設施個數變多時，找到最佳解的狀況較不穩定。

(2) 基本形式+ 隨機選擇法

此實驗目的是要與實驗(1)比較不同 Selection Operator 作法的效果，其作法如下：

- Initialize Population：同實驗(1)。
- Crossover Operator：同實驗(1)。
- Mutation Operator：同實驗(1)。
- Selection Operator：任挑兩個解留下較好的那個解，放回去再繼續挑。

實驗結果：與 Rank 選擇法比起來，隨機選擇法執行效果並不好，且當設施個數變多時，幾乎找不到最佳解。

(3) Moses +AND/OR+ Rank 選擇法

此實驗目的是要了解用 AND 與 OR 來實作 Crossover 與 Mutation Operator 的效果，其作法如下：

- Initialize Population：使用亂數產生 99 解，用 Moses 產生 1 組解當作一開始的

最佳解。

- Crossover Operator：從最佳的 90 組解，隨機挑出兩組作 AND。
- Mutation Operator：從 Crossover 之後的 90 組解，隨機挑出兩組作 OR，然後經過 10% 機率將 3 個 bit 變為 0 ，20% 機率將 2 個 bit 變為 0，重複做 60 次。
- Selection Operator：同實驗(1)。

實驗結果：整體的執行效果很不好。其中 benchmark 104 的結果代表透過 Moses 產生最佳解的初解，就是整體的最佳解。

(4) Moses +AND/OR+隨機選擇法

此實驗目的是要與實驗(3)比較不同 Selection Operator 作法的影響，其作法如下：

- Initialize Population: 同實驗(3)
- Crossover Operator：同實驗(3)。
- Mutation Operator：同實驗(3)。
- Selection Operator：同實驗(2)。

實驗結果：與實驗(3)類似，即當 Crossover 與 Mutation Operator 的設計不恰當時，Selection Operator 作法的影響不大。

(5) Moses +基本形式+ Rank 選擇法

此實驗目的是要了解在實驗(1)加入 Moses 解法當一組初解的影響，其作法如下：

- Initialize Population：使用亂數產生 99 組解，用 **Moses** 產生 1 組解當作一開始的最佳解。
- Crossover Operator：同實驗(1)。
- Mutation Operator：同實驗(1)。
- Selection Operator：同實驗(1)。

實驗結果: 與實驗(1)類似，但是找到最佳解的平均代數跟時間的效能都有所提升。即若在各種 Operator 的設計效果不差的情況下，加入 Moses 解法當一組初解，對整體的效果確有助益。

(6) NBH+基本形式+ Rank 選擇法

此實驗目的是要了解在實驗(1)加入 NBH 解法當一組初解，與實驗(5)比較的結果，其作法如下：

- Initialize Population：使用亂數產生 99 解，用 NBH 產生 1 組解當作一開始的最佳解。
- Crossover Operator：同實驗(1)。
- Mutation Operator：同實驗(1)。
- Selection Operator：同實驗(1)。

實驗結果：在設施數量變多的 benchmark 131-134 顯示結果比加入 Moses 的整體效果更好，於是後續的設計便以 NBH 取代 Moses。

(7) NBH+R1 +Rank 選擇法

此實驗目的是要了解在實驗(6)中 Mutation 只針對 Crossover 後的基因做變化與全域 Mutation 的差異性。

- Initialize Population：同實驗(6)。
- Crossover Operator：從最佳的 90 組解，隨機挑 2 組做「隨機單點切段交換」做 50 次。
- Mutation Operator：每一次作完「隨機單點切段交換」之後，隨機挑選 2 組，10% 機率產生 2 bits 突變，20% 機率產生 1 bit 突變。
- Selection Operator：同實驗(1)。

實驗結果: 如表 2 所示，結果雖然與實驗(6)類似，但卻有些許的落差，應是因為 Mutation 後的基因接續做 Crossover 的收斂與發散平衡較佳。

(8) NBH+A1+Rank 選擇法

此實驗目的是要了解在此實驗中，Crossover 的切段點改變為固定點的差異性。

- Initialize Population：同實驗(6)。
- Crossover Operator：從最佳的 90 組解，隨機挑 2 組做「中點切段交換」做 50 次。
- Mutation Operator：每一次作完「中點切段交換」之後，隨機挑選 2 組，10% 機率產生 2 bits 突變，20% 機率產生 1 bit 突變。
- Selection Operator：同實驗(1)。

實驗結果: 與實驗(7)相比較，最佳解的平均代數跟時間的效能都下降，應是因為切段點設定為固定點的結果，將會造成發散力量大幅減小。

(9) NBH+R2+Rank 選擇法

此實驗目的是要了解在實驗(7)中，Crossover 的交換段改變為兩個隨機點的中段之差異性。

- Initialize Population：同實驗(6)。
- Crossover Operator：從最佳的 90 組解，隨機挑 2 組做「雙隨機點中段交換」做 50 次。
- Mutation Operator：每一次作完「雙隨

機點中段交換」之後，隨機挑選 2 組，10% 機率產生 2 bits 突變，20% 機率產生 1 bit 突變。

- Selection Operator：同實驗(1)。

實驗結果: 與實驗(7)相比較下，找到最佳解的平均代數跟時間的效能都有些微的提升。

(10) NBH+A2+Rank 選擇法

此實驗目的是要了解在實驗(8)中，Crossover 的交換段改變為中段的差異性、以及如同實驗(9)再次求證隨機與定點切段的差異。

- Initialize Population：同實驗(6)。
- Crossover Operator：從最佳的 90 組解，隨機挑 2 組做「雙定點中段交換」做 50 次。
- Mutation Operator：每一次作完「雙定點中段交換」之後，隨機挑選 2 組，10% 機率產生 2 bits 突變，20% 機率產生 1 bit 突變。
- Selection Operator：同實驗(1)。

實驗結果: 找到最佳解的平均代數跟時間的效能都大幅下降，證明了定點切段的成果比起隨機切段還差。

上述各項實驗數據，統計之後列於表 2。經由上述實驗結果整體分析可知:

(1) Selection Operator 的設計上，Rank 選擇法比隨機選擇法效果好。

(2) 設計用 AND 實作 Crossover Operator 的想法是基於要把各組解都會開啟的設施，視為重要的、有一定影響程度的設施，其出現在最佳解的機率應該不低。而用 OR 來實作 Mutation Operator 的想法是基於擔心保留下來的只是某一區域的重要的開啟設施，所以再做 OR 運算將每一區域的重要的開啟設施集合起來，但又擔心 OR 之後的開啟設施過多，因此再用亂數將某些已開啟的設施關閉。實驗結果顯示這些想法考慮不夠嚴謹，需再加以修正，於是在後續實驗開始捨棄 AND/OR 做法。

(3) 在初解中加入一組利用 Moses 或 NBH 做法，對於找到最佳解的代數有明顯的下降。且 NBH 解法又比 Moses 解法佳。

(4) 在 Crossover Operator 切割點上，結果顯示固定的切割點非常容易造成發散力量不夠，造成求解無法更進一步找尋最佳解。

(5) Mutation Operator 變化的力量不夠大，不易跳開區域解的範圍，使得在尚未找出最佳解時，收斂速度加快。

此外，由表 2 可知，不管用哪一種組合的解法，每一種無容量限制之設施定位問題的 benchmark 都曾有被找出最佳解的狀況，且其他非最佳解也都相當近似最佳解。亦即，本研究驗證了基因演算法能在合理的時間內求得近似最佳解，相當適合用來解決 NP 性質的問題。

5. 結論

本研究採用 Meta-heuristic 中目前應用最廣的基因演算法來解決無容量限制之設施定位問題，因為 Meta-heuristic 加入一些亂數的方式來尋找最佳解，若運氣好一下子跳到正確的最佳解區域，馬上可以找到最佳解。但也有可能隨處亂跳到不同的解區域，甚至是已經到了最佳解區域，然後又跳離開該區域。亦即，解答的品質與尋找時間較不一定。

從實驗結果得知，在初解中加入一組利用 deterministic 演算法找出來的近似解，可以加速演化，並防止發散力量太大，導致不亦收斂，對於找到最佳解的代數有明顯的下降。且在求解前半段流程較仰賴 Crossover 的變化，後半段因為基因群都演化到非常相似，於是比較需要良好的 Mutation，但突變率過小的時候，往往造成基因演化不易，較難找尋到最佳解，且在演化過程中，現階段最佳解如果已多代未演化，結果通常是陷於前段區域解，較無法跳到最佳解或是鄰近最佳的解。

所以，收斂與發散兩股力量若設計不恰當，將使得找最佳解的過程無原則的隨處亂跳，可能會很快找到最佳解，但也可能一直找不到或需要找很久，使得平均尋找最佳解時間變得很長。反之，整個解題方法中收斂與發散兩股力量若設計恰當，將會比較快速的找到整體的最佳解。而在尋找最佳解時，若透過已經接近最佳解的初始解，如品質不錯的NBH當初解，亦可縮短大多數尋找最佳解的時間。

6. 未來展望

在可以找到最佳解的前提下，也可透過 Cache 或一些加速計算的資料結構與演算法，來減少最佳解的尋找總時間。此外，採用 Meta-heuristic 來解決無容量限制之設施定位問題，尚有許多其他的方法，例如 Tabu Search、Simulated Annealing 等等，未來將嘗試透過其他方法或是將各種 Meta-heuristic 方

法做適當的整合，來解決無容量限制之設施定位問題，並分析其解題效果。例如，可以將 Neighborhood 的概念，轉換為基因演算法的 Crossover Operator 或 Mutation Operator；或利用基因演算法負責 Global Search，透過 Tabu Search 負責 Local Search 等等作法。另外，設施定位問題的各種變型問題的解決，亦是可以深入探討的議題。

參考文獻

[1] Krivitski, D., Schuster, A. and Wolff, R., "Local Hill Climbing in Sensor Networks," ***SIAM International Conference on Data Mining***, pp. 38- 47, April 21-23, 2005.

[2] Krivitski, D., Schuster, A. and Wolff, R., "A Local Facility Location Algorithm for Sensor Networks," ***International Conference on Distributed Computing in Sensor Systems (DCOSS)***, Marina del Rey, CA, USA, pp. 368-375, 2005.

[3] Charikar, M. and Guha, S., "Improved combinatorial algorithms for facility location problems," ***Society for Industrial and Applied Mathematics***, Vol. 34, No. 4, pp. 803-824, 2005.

[4] Aissiou, M. and Guerti, M., "Genetic Algorithms Application for the Automatic Recognition of the Arabic Stop Sounds," ***Journal of Applied Sciences Research,*** Vol. 3, No.5, pp. 358-366, 2007.

[5] Araujo, L. and Merelo, J. J., "A Genetic Algorithm for Dynamic Modelling and Prediction of Activity in Document Streams," ***Genetic and Evolutionary Computation Conference, Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation***, London, England, United Kingdom, pp. 1896 - 1903, 2007.

[6] Li, I H., Liao, I E. and Pang, W. Z., "MINING CLASSIFICATION RULES IN THE PRESENCE OF CONCEPT DRIFT WITH AN INCREMENTAL GENETIC ALGORITHM", ***Journal of Theoretical and Applied Information Technology***, Vol. 4, No. 7, pp.608-623, 2008.

[7] Jain, K., Mahdian, M. and Saberi, A., "A New Greedy Approach for Facility Location Problems," ***Proceedings of the 34th Annual ACM Symposium on Theory of Computing***, pp. 731-740, 2002.

[8] Al-Sultan, K. S. and Al-Fawzan, M.A., "A tabu search approach to the uncapacitated facility location problem," ***Annals of Operations Research***, Vol. 86, pp. 91-103, 1999.

[9] Michel, L. and Hentenryck, P. V. "A simple tabu search for warehouse location, " ***European Journal of Operational Research***, Vol. 157, pp. 576-591, 2004.

[10] Kratica, J., Tosic, D., Filipovic, V. and Ljubic, I., "Solving the Simple Plant Location Problems by Genetic Algorithm," ***RAIRO Operations Research***, Vol. 35, pp. 127-142, 2001.

[11] Codreanu, I., Codreanu, C., Obreja, V. V. N. and Avram, M. "Use of genetic algorithms in heat transfer problems," ***Semiconductor Conference, 2004. CAS 2004 Proceedings. 2004 International***, Vol.2, 2004, pp. 499 – 502, 2004. DOI 10.1109/SMICND.2004.1403058

[12] Beasley, J. E., "OR-Library: Distribution test problems by electronic mail," ***Journal of Operational Research Society***, Vol. 41, pp. 1069-1072, 1990.

[13] Beasley, J. E., "Lagrangian heuristics for location problems," ***European Journal of Operational Research***, Vol. 65, pp. 383-399, 1993.

表 2: 10 種主要解法組合的實驗結果

Benchmark		1		2		3		4		5		6		7		8		9		10	
		平均值	找到最佳解次數	平均值	找到最佳解次數	平均值	找到最佳解次數	平均值	找到最佳解次數	平均值	找到最佳解次數	平均值	找到最佳解次數	平均值	找到最佳解次數	平均值	找到最佳解次數	平均值	找到最佳解次數	平均值	找到最佳解次數
71	代數	13	10	53.7	10	42.3	3	57.8	6	14	10	12.1	10	12.9	10	17.4	10	12.6	10	18.6	10
	時間	0.9		3.9		3.3		4.3		1		8.1		6.9		7.3		8.5		9.2	
72	代數	11.1	10	32.8	10	110	1	79	1	11.1	10	13.6	10	14	10	16.5	10	14.4	10	24	10
	時間	0.9		2.3		9		6		0.7		9		7.1		7.3		8.3		7.4	
73	代數	19.1	10	103.3	4	Fail	0	Fail	0	14.2	10	27.2	10	16.4	10	16.5	6	14.2	10	19.8	5
	時間	1.3		7		Fail		Fail		1		10		7.6		7.3		8.6		8.8	
74	代數	12.2	10	109.6	5	Fail	0	Fail	0	8.2	10	14.1	10	12.6	10	22.9	10	12.5	10	20.2	10
	時間	0.9		7.2		Fail		Fail		0.8		9.4		9.1		8.2		8.5		7.7	
101	代數	22.7	9	145.8	4	Fail	0	Fail	0	68.4	7	57.4	7	39.3	7	47.6	5	24.4	9	38.5	4
	時間	1.9		15.5		Fail		Fail		7		13.9		10.7		12.8		9.8		8.8	
102	代數	29	10	174.5	2	Fail	0	Fail	0	33.9	10	34.2	10	53.4	10	55.1	10	27.8	10	51.9	10
	時間	2.5		15		Fail		Fail		3.8		9.7		12.6		13.5		10.8		10.5	
103	代數	30.6	5	295	1	Fail	0	Fail	0	52	3	45.2	8	57.3	3	23.7	6	24.8	6	30.5	2
	時間	2.8		33		Fail		Fail		5.7		12		8.3		6.5		9.5		8.5	
104	代數	20.5	10	231.7	3	1	10	1	10	1	10	36.8	10	20.9	10	35.1	10	19.3	10	41.4	8
	時間	1.9		24.7		0		0		0		12.1		10		10.9		9.9		10.8	
131	代數	84.8	5	Fail	0	Fail	0	Fail	0	98.4	5	118.6	3	105	2	174	2	119	4	Fail	0
	時間	13.4		Fail		Fail		Fail		16.6		23.7		21.5		36		23.3		Fail	
132	代數	101.1	8	Fail	0	Fail	0	Fail	0	66.6	9	100.3	8	118.3	4	114.8	5	98.7	7	175	2
	時間	16.9		Fail		Fail		Fail		10.8		21.9		22.8		14.6		20.3		30.5	
133	代數	106.5	2	Fail	0	Fail	0	Fail	0	Fail	0	118.3	3	44	1	Fail	0	34	1	Fail	0
	時間	17.5		Fail		Fail		Fail		Fail		24.7		14		Fail		11		Fail	
134	代數	76.9	9	Fail	0	Fail	0	Fail	0	13.4	5	54	8	53.8	8	97	4	57.1	7	129	5
	時間	13.0		Fail		Fail		Fail		2		18.3		10.4		21.3		14.7		20.4	