網狀網路上可容錯之直線式史坦那樹試誤型演算法

范啟明
國立臺灣海洋大學資訊科學系
kevincmfan@hotmail.com

詹景裕　林英仁
國立臺灣海洋大學資訊科學系
{b0199, ijlin}@mail.ntou.edu.tw

摘要

直線式史坦納樹(rectilinear Steiner tree; RSP)係指於一網格狀平面上，僅利用垂直線段(vertical segment)與水平線段(horizontal segment)，針對所給定的節點集合 *Z*，將 *Z* 中的各個節點以最短的總連接路徑相互連接後所形成的樹。其與最小伸展樹(minimal spanning tree; MST)不同之處在於最小伸展樹僅允許由節點連接至節點的路徑型態，而直線式史坦納樹尚可允許由節點連結至一現存路徑上的某一點。本文是先將利用 Lee 的連結演算法來求取出任意指定兩點之間的最短路徑。再透過距離等高線累加的觀念來求出欲連結的各個節點之間的關鍵節點(critical vertex)；最後，再利用 Lee 演算法中的路徑回溯過程取得直線式史坦納樹。本文提出的試誤型演算法其空間複雜度為 $O(pN)$，而其時間複雜度為 $O(p^2N)$，其中 *N* 為非障礙物的節點總數，*Z* 為具有 *p* 個欲相互連接成一網路的節點所成的集合。

關鍵詞：容錯、積體電路設計、多點傳送、多處理器排程、直線式史坦納樹。

一、緒論

(一) 史坦那樹問題之定義

史坦納樹問題之更正式的數學定義為：若給定一個具有 $|V|$ 個節點的節點集合 *V*、$|E|$ 個邊的邊集合 *E* 以及成本函數 $C: E \rightarrow R$ (real number)的無向網路圖 $G = (V, E, C)$，以及 *V* 的子集合 *Z*，*Z* 包含於 *V*，*Z* 為具有 *p* 個欲相互連接成一網路的節點所成的集合。則史坦納樹問題可以表示為：求解 *G* 的子網路 G_Z 使得在 *Z* 中的每一節點對 (z_i, z_j) $0 \le i, j \le p$，均存在一條路徑，並使 G_Z 的總成本 $C(G_Z)$為最小。G_Z 中共有兩種節點型態，包含於 *Z* 中的節點稱為 *Z* 節點(*Z*-vertices)，其餘的節點則稱為 *S* 節點(*S*-vertices)。子網路 G_Z 則稱為於 *G* 上對 *Z* 而言的最小史坦納網路。

(二) 直線式史坦那樹演算法

直線式史坦納樹(rectilinear Steiner tree; RST)為史坦納樹(Steiner Problem in Network; SPN)的一個特例。其係指於一網格狀平面上，僅利用垂直與水平線段，將節點集合 *Z* 中的各個節點以最短的總連接路徑相互連結後所形成的樹。該問題首先由 Hanan 加以數化並證明其為史坦納樹的一個特例[4]。令 $G(Z)$ 為由通過每個 *Z* 節點之水平與垂直線所產生的一個網格平面圖，則對 *Z* 而言，存在一個僅使用 $G(Z)$ 中之線段而形成的最小史坦納樹(Steiner minimal tree; SMT)。因此，在每個線段所形成的交點若非為 *Z* 節點的話，則其為 *S* 節點(即史坦納節點)。對於每一個 RST 的問題，均可以 SPN 的演算法來加以解決[22]。但若針對直線式網路(rectilinear network)的特性而加以設計的演算法將能更有效率地解決 RST 的問題。與 SPN 問題相同，RST 問題亦不存在可於多項式時間(polynomial time)內解決的確切演算法(exact algorithm)，Garey 等人已證明了 RST 問題亦為 NP-complete[2]。因此，許多解決 RST 問題的試誤型演算法(heuristic algorithm)亦已被提出。

Aho 等人針對一些特殊 RST 提出了兩個演算法[1]，其第一個演算法的時間複雜度係與 *Z* 集合中點的個數 *p* 成線性(linear)關係，但與 $G(Z)$圖型中的行數或列數(rows or columns)成指數(exponential)關係，其可用來解決當行數(或列數)較少時的 RST 問題。Aho 等人所提出的第二個演算法則適用於當所有 *Z* 集合中的節點均位於 $G(Z)$圖中的邊界上時之 RST 問題，其時間複雜度為 $O(rs^2+sr^2)$，假設 $G(Z)$含有 *r* 個列數及 *s* 個行數。

Hanan 提出了一個 $O(p^2)$的試誤型演算法[3]，其中 *p* 為節點集合 *Z* 中的節點個數，其將 Prim 的最小伸展樹演算法[17]應用至 *Z* 上，並將 *Z* 集合中的各個點以 X 座標值來加以排序後得出最小的 RST。Smith 等人則提出一個 $O(p^4)$的試誤型演算法[20]，Lee 等人則提出 $O(p^2)$的試誤型演算法[15]，該兩個演算法均是透過與 Hanan 相同的 MST 演算法為基礎的方式來求得 RST。Hwang 等人，將 Lee 等人的演算法加以修改而得出一個 $O(p\log p)$的試誤型演算法[7]。Smith 等人又以與 Hwang 相近的觀念提出了利用 MST 來重標(re-label)*Z* 節點的方式，並透過循序引入最佳的 *S* 節點對(*S*-vertices pairs)的方式來轉換 MST 至可伸展 *Z* 之最小成本樹演算法，其時間複雜度亦為 $O(p\log p)$ [19]。此外，Hwang 並証明了以 MST 方式來找尋史坦納最小樹其最差的狀況將不會超過正確解的 1.5 倍，即 MST 的成本比上 SMT 的成本將不會超過 3/2[6]。Ho 等人並於最近利用 L 形(L-shape)及 Z 形(Z-shape)來替換 MST 上之向量路徑的方式於 MST 上找尋 SMT[10]，其時間複

雜度介於 $O(p^{1.5})$至 $O(p^{2})$。另外，基於確切型演算法以特定限制條件加以精簡而得出的遺傳演算法(genetic algorithm)，如 Hesser[5]、Julstrom[10]、Kapsalis[11]等提出的均屬之。

二、可容錯之直線式史坦納樹試誤型演算法

(一) Lee 的最短路徑連接演算法

1961 年 Lee 提出的最短路徑連接演算法(簡稱 Lee 演算法) 是一個被廣泛運用的網格式空間資料結構演算法[14]，於網格平面上給定起點及終點及若干障礙物，若於起點及終點間存在一組最短且具障礙物避碰之路徑集合，則 Lee 演算法可保証於 $O(N)$時間內找到該路徑集合中的其中一條最短路徑，其中 N 為網格平面上自由網格數(number of free cells)。Lee 演算法分為三個階段，第一個階段稱為洪氾階段(wave propagation phase)，其由起點開始，依序向其四向鄰近網格擴散，每擴散一次則其與起點的距離加 1，直至全部自由空間網格均被探索過。第二階段稱為路徑回溯階段(back trace phase)，其由終點 *Dest* 開始，依序找尋其四個鄰近網格中距離起點之距離值最小者，一路回溯至到達起點為止。第三個階段則為路徑反轉階段，將回溯階段時所得的路徑，依序反轉而得到由起點直至終點的一條最短路徑。

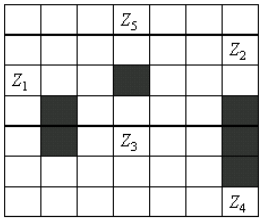
Lee 演算法中洪氾階段所產生的結果，當其每格移動速度皆為固定時，於各點中的距離值，可視為由起點至網格圖中其餘各點的到達時間(arriving time)。其觀念類似於地理圖上之等高度曲線圖。對 Lee 演算法中洪氾階段所產出的結果，吾人稱其為等距離圖(distance level curve)，本文中擬提出的直線式史坦納樹試誤型演算法將利用等距離圖累加(distance level curve accumulation)的觀念求出距離各個所求節點平均值最近的關鍵節點(critical vertices)。

(二) 可容錯直線式史坦納樹試誤型演算法

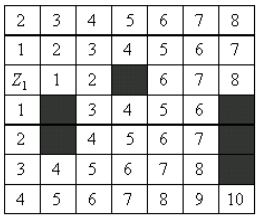
求得關鍵節點後，即可依距離關鍵節點最近之 Z 節點開始，以 Lee 演算法中的回溯步驟，由關鍵節點開始，回溯至距離關鍵節點最近之一 Z 節點。其所得的路徑為第一條關鍵路徑(critical path)。其餘的各點，則可採線性搜尋(linear search)的方式，再輔以哨兵(sentinel)的資料結構技巧求出距關鍵路徑最近的剩餘 Z 節點。依序利用 Lee 演算法中的回溯步驟將其加入關鍵路徑中。最後，直至所有 Z 節點已被計算過為止，所得的關鍵路徑即為給定 Z 節點的直線式史坦納樹。所謂可容錯即是將錯誤節點(faulty nodes)視為網狀架構上的障礙物(obstacle)。

(三) 可容錯直線式史坦納樹試誤型演算法

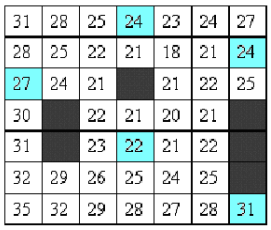
限於篇幅，本演算法細節詳述於[23]。本演算法之步驟圖如圖一所示。



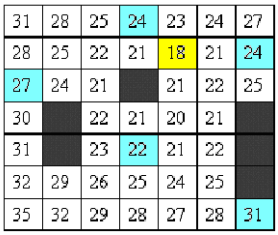
(a) 起始狀態



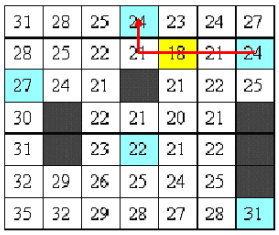
(b) Z_1 節點的等距離圖



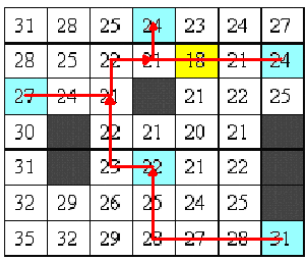
(c) Z_1 節點至 Z_5 節點等距離圖累加後之結果



(d) 總和最小之關鍵節點 Z



(e) 完成第一個節點的連結



(f) 最後的結果

圖一 史坦納樹試誤型演算法步驟圖

Step 1: Initailization 圖一(a)

Z 集合中共有 p 個終端節點，且包含許多的障礙物，將最初始的狀態設定。

Step 2: Individual Costs 圖一(b)

針對每一個終端節點使用 Lee 的演算法

作洪氾計算各節點的個別等距離圖。

Step 3: Cost Accumulation 圖一(c)

將 Step1 中所產生的所有等距離圖作累加產生一組累加總合的 Table，從這個 table 中取出最小值的節點當作關鍵節點。

Step 4: Initial Critical Path 圖一(d)

在 Z 集合中搜尋距離關鍵節點最近的終端節點，由此節點當做起點，距此節點最近的另一終端節點當作終點，使用 Lee 的演算法產生的第一條路徑稱為關鍵路徑。並從 Z 集合中移除上述兩個節點。

Step 5: Iterations 圖一(e)

在剩餘的 Z 節點成本表中，搜尋距離關鍵路徑最近的 Z 節點，將此節點以 Lee 的回溯步驟產生另一條路徑，將此路徑加入到原有的路徑中，並將剛才的節點從 Z 中移除，再依序尋找下一個離路徑最近的 Z 節點，重複作上述的動作，直到 Z 集合為空集合為止。

Step 6: Output Steiner Tree 圖一(f)

將產生的路徑輸出即為直線式試誤型史坦納樹。

(四) 可容錯之直線式史坦納樹試誤型演算法的空間複雜度分析

關於可容錯之直線式史坦納樹試誤型演算法的空間複雜度分析，很明顯地該演算法使用了 1 個 $m \times n$ 二維陣列 $v_{i,j}$ 來儲存網狀網路中各個節點的狀態、1 個 $m \times n$ 二維陣列 D^{global} 用來儲存等距離累加圖、p 個 $m \times n$ 二維陣列 $D^{local}(k)$ 用來儲存各個 Z 節點個別的等距離圖以及數個輔助計算用的資料結構。其中二維陣列的空間需求為 N 即 $m \times n$ 個，輔助計算用的資料結構中，空間需求最大的 LL_Z 及 LL_{RST} 在最差情況下，其空間需求亦均不會超過 N 個。所以本演算法的空間複雜度 (space complexity) SC_{total} 為：

$$SC_{total} = SC_v + SC_{D\_global} + SC_{D\_local} + SC_{auxiliary}$$
$$\le N + N + pN + 2N = (p+4)N = O(pN)$$

(五) 可容錯之直線式史坦納樹試誤型演算法的時間複雜度分析

關於可容錯之直線式史坦納樹試誤型演算法的時間複雜度分析，可分為三大階段來探討，第一階段(Step 2)為計算各個 Z 節點分別的等距離圖及其等距離累加圖，其時間複雜度於計算個別的等距離圖時，受限於 Lee 演算法的洪氾程序，其時間複雜度為 $O(N)$，但因為需計算 p 個，因此時間複雜度為 $O(pN)$；而在等距離累加圖的計算上，共需對 N 個節點計算 p 個等距離圖的累加值，因此其時間複雜度亦為 $O(pN)$。因此第一階段的時間複雜度 TC_{phase_1}：

$$TC_{phase\_1} = O(pN)$$

在第二階段(Step 4)找尋關鍵節點時，其所使用的的方式為對 LL_Z 做線性搜尋，找出其於等距離累加圖上最小值者，由於 LL_Z 的長度最多不超過 p 個，因此第二階段的時間複雜度 TC_{phase_2} 為：$TC_{phase\_2} = O(p)$

在第三階段(Step 5)時，由於每次均須找尋距離關鍵路徑最近的一個節點，而每次須要比對的次數為 $O(pN)$，由於採對 LL_Z 做線性搜尋的方式，因此第三階段的時間複雜度 TC_{phase_3} 為：$TC_{phase\_3} = O(p^2 N)$。

綜合三個階段的時間複雜度可知道該演算法的時間複雜度的最大需求係受限於第三階段，因此該演算法的時間複雜度為 $O(p^2 N)$，即：$TC_{total} = TC_{phase\_1} + TC_{phase\_2} +$

$$TC_{phase\_3} = O(p^2 N)$$

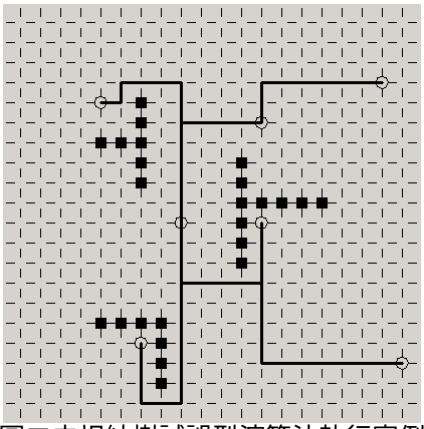
(六) 演算法執行實例

圖二為本文擬提出之試誤型演算法的模擬實例。其為 20X20 的網狀網路，圖中之四方格點代表障礙物，圓形格點表欲相連接之 Z 節點。其中共有 24 個障礙物，7 個 Z 節點，利用本計劃所提出的演算法求得之最小史坦納樹之總成本為 56，其與確切型演算法所求得值相差為零。

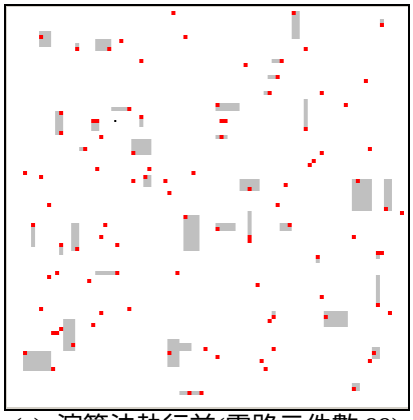
(七) 試誤型演算法於 VLSI 及 PCB 上之應用實例演示

利用史坦那樹演算法可以在 $O(p^2N)$的時間內快速地找出連接各個電路區塊(block)中欲相互連接成一個網路最短路徑的近似解。其中 N 為電路配置區域(layout)上的所有節點數，p 則為欲相互連接成一個網路的節點數。圖三則為較大型之 VLSI 模擬，其中灰色部份表電路元件（component），紅點則為該電路元件之接點。圖三中欲相連接之電路元件數為 99 個，經由上述之演算法計算後所得之史坦納樹路徑長度為 3372 單位。

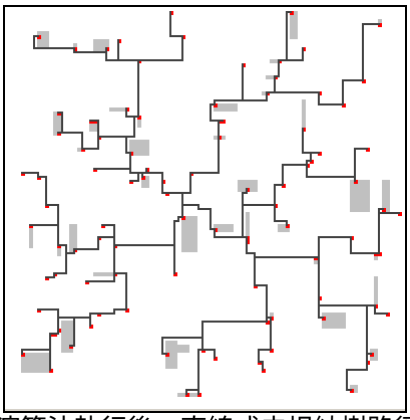
於圖三中，由於其自由節點數過大，若以第三節中欲提出之確切型演算法來加以計算，則其所需的計算時間將為天文數字。若以人力對圖三所得之結果加以觀察時，可很容易地發現其所得之結果仍有很明顯的改善空間、即其路徑長度非為最佳值。在電路元件數較少時，人力可以勝任計算 VLSI 中各元件之路由路徑之規劃工作。但倘若電路元件超過一個定數時，以人腦來加以求取路由路徑時，將造成極大的負擔。且亦無法保證可以求出最佳值。在 VLSI 的路由路徑找尋時，最難的即為從無到有



圖二史坦納樹試誤型演算法執行實例



(a) 演算法執行前(電路元件數 99)



(b)演算法執行後，直線式史坦納樹路徑長 3372 單位。

圖三 RST 演算法於 VLSI Routing 上之應用實例

的路徑規劃，即倘若以人力從無到有地規劃出類似本演算法所提出之近似解，其所耗之人力需求極大。但若以本演算法置於一般處理能力等級的電腦上執行時，其所需之執行時間，約為數秒鐘。

因此本節所提出之演算法，雖然無法在電路元件數多時找出史坦納樹路徑之確切值，但其可於極短的時間內求解出近似解。此對於 VLSI 工程師來說極為有用。因其可利用本節所提之演算法快速找出一個欲連接網路的近似解，而後，根據本演算法所計算出之結果，輔以調整規劃之工具，讓工程師得以對演算法執行結果加以微調，依據其經驗值與個人之判斷快速地找出更佳的路徑。此為本演算法應用於 VLSI 之路由路徑找尋時的最大貢獻。

三、可容錯之直線式史坦納樹確切型演算法

由於在可容錯之直線式史坦納樹的研究領域目前仍無可供做效能比較的指標，因此第二節中所提之試誤型演算法其執行的效能無法加以驗證。本節將提出一個可容錯之直線式史坦納樹確切型演算法來與第二節中之試誤型演算法之執行結果加以比較

(一) 可容錯直線式史坦納樹確切型演算法

本節所提之可容錯之直線式史坦納樹確切型演算法係以一種完全探索（exhausted searching）的方式，對所有的自由節點加以變化，即設定其為障礙節點或自由節點兩種可能性。而後再測試在自由節點變化後，所給定的 Z 節點是否可以完全連接。若可以完全連接時則依序遞減自由節點數，直至所指定的自由節點數為可完全連接所有 Z 節點達到最小值時則停止，此即為所求之史坦納樹的確切解。

在簡化計算時，本文所提之演算法亦利用了一些技巧來改進計算效能，其一為設定開始計算時的自由節點數之上限(upper bound)，即為試誤型演算法所求得之路徑長度。其二，由於欲連接所有的 Z 節點，至少需要 Z 節點個數的自由節點，因此設定此為計算時最後的下限(lower bound)。此一計算時的簡化，雖未能改變演算法執行時的時間複雜度，但在實務上卻能有效地大幅減少計算時所需的時間，因為第二節中所提出之試誤型演算法其執行結果與確切解相差不多。在直線式史坦納樹確切型演算法中，其所使用到的空間資料結構可分為兩部份來加以討論。首先，在儲存網狀網路的資訊部份，其使用了一個 $m \times n$ 二維陣列 $v_{i,j}$ 來儲存原始的網狀網路，並使用了一個 $m \times n$ 二維陣列 $v'_{i,j}$ 來儲存計算時所暫存用的網狀網路，因此其空間複雜度為 $O(m \times n)=O(N)$，其中 N 為網狀網路的節點數。

而在第二部份，在儲存自由節點之可能性變化時，因其計算過程中並不需儲存之前所計算過的可能性組合字串，所以吾人可用遞迴演算法來加以隨機取出指定序號的可能性，因此其所需的空間資料結構需求為一字串變數，其大小不超過所有自由節點 N_{free} 的個數。即其空

間複雜度為 $O(N_{free})= O(N)$。綜合以上兩部份之分析，可容錯之直線式史坦納樹確切型演算法之空間複雜度為 $O(N)$。

本節所提之可容錯之直線式史坦納樹確切型演算法之時間複雜度為 $O(p^2N^4)$，其中 N 為網狀網路的節點數，p 為欲連接的節點個數。確切型演算法本身以及複雜度分析詳情請參照[23]。

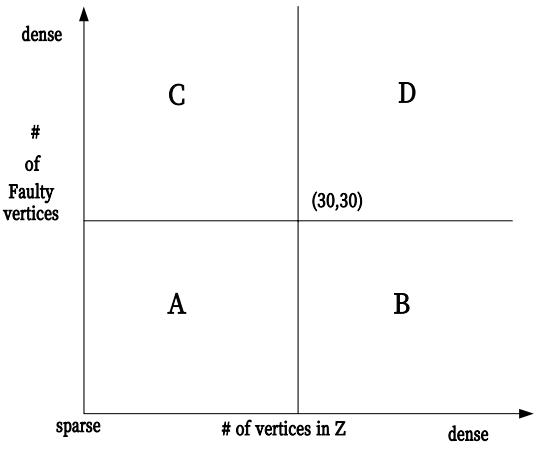
四、演算法模擬執行結果與效能驗證

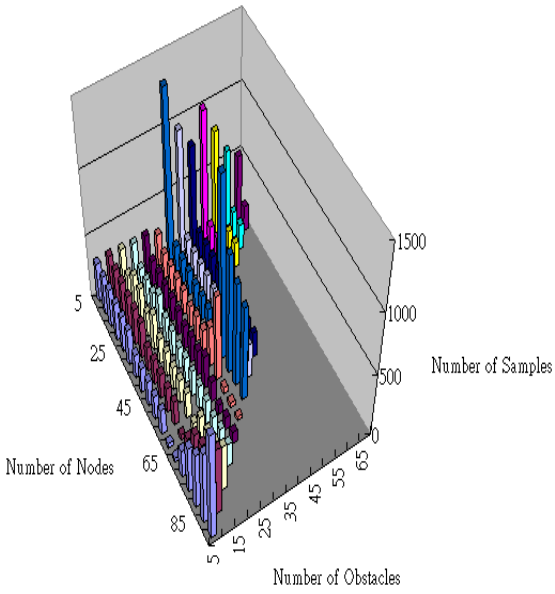
本章將利用電腦隨機產生網狀網路樣本來加以驗證第二章所提之可容錯之直線式史坦納樹試誤型演算法之執行效能。

(一) 隨機樣本產生原則

於電腦模擬時，隨機樣本的產生係依 Z 節點的個數及障礙節點的個數來做為給定參數，而網狀網路的大小則為 10×10。在 Z 節點個數及障礙節點個數的產生時，吾人根據 Z 節點或障礙節點的密集度將其分為四個區域，如圖四所示。以 Z 節點數及障礙節點數各為 30 個做為分區的依據，將其分為區域 A、B、C、D。區域 A 內的樣本均為 Z 節點及障礙節點均為 sparse 的情況；而區域 B 則是 Z 節點為 dense，而障礙節點為 sparse 的情況； 區域 C 則為 Z 節點為 sparse，障礙節點為 dense 的情況；最後，於區域 D 中則為 Z 節點及障礙節點均為 dense 的情況。而後，依四個區域分別產生隨機樣本以供驗證演算法之效能。而在樣本產生時，由於某些樣本所需的計算時間需求係為天文數字，因此在樣本產生後，吾人將針對其所需的計算時間加以限制為五小時，於五小時內可以計算出來的樣本，則將其寫入資料庫，反之則將該樣本丟棄。

統計學上常用由目標群體(population)中某個選定的樣本(sample)來做出對目標群體的推論(estimation)，較常見的推論方式有兩種，一個參數推論(parameter estimation)以及(hypothesis estimation)。其中參數推論係以給定





圖四 全部樣本分佈圖

樣本的某些參數，據此參數的統計分析後，來對目標群體做出推論。

根據所推論出之結果的不同，又有兩種推論結果，其為點推論(point estimation)及區間推論(interval estimation)。其中點推論係指根據給定樣本的參數表現，對群體行為做出某一固定值的推論，如推論台灣地區男性平均壽命為 72 歲，此即為點推論的結果。而區間推論則為根據給定樣本的參數表現，對目標群體行為做出某一固定區間的推論，如推論台灣地區男性平均結婚為 25-29 歲，為區間推論。在做統計學上的推論時，必須針對推論的正確性提出推論的信心水準(confi- dence level)，據此信心水準，才能讓觀看統計數據的人可以了解，在統計時所採用的樣本數及是否足以代表目標群體的行為表現，並可了解做此推論時，其與目標群體的實際表現的差距究竟有多大。因此在統計學上常用信賴區間的方式，來對推論做出在某一信心水準下，其與標準值的標準差究竟有多大。信賴區間的計算方式係依以下步驟進行。先計算每一個樣本差：$X_i = length_{heuristic} - length_{exact}$，其中 $0 \le i < n$，n 為總樣本數。而後再計算總樣本差為：$\overline{X} = \sum_{i=0}^{n-1} X_i \Big/ n$。最後，利用 95%的信賴區間(confidence interval)公式：$CI = \pm 1.96 \times \left(\overline{X} \Big/ \sqrt{n} \right)$ 加以判斷樣本數是否已進入所要的信賴區間。

(二) 模擬結果分析及說明

如表一所示，電腦模擬結果在有效樣本 42809 個隨機產生的網狀網路下，其在 95%的信心水準下，其總樣本差為 1.34，即信賴區間

表一 全部模擬結果

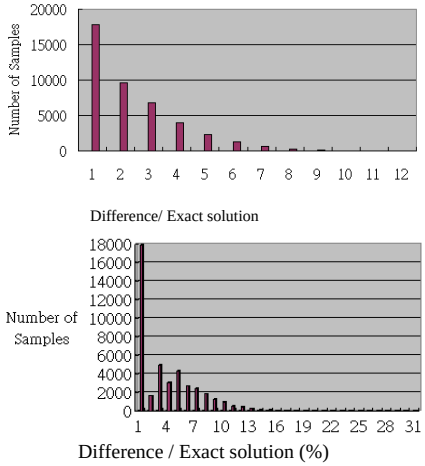
差值	樣本數	差值	樣本數
0	17878	7	270
1	9563	8	82
2	6752	9	18
3	3967	10	5
4	2341	11	1
5	1304	> 11	0
6	628		
標準差 1.34			

表二 分區模擬結果

樣本數 / 差值	A	B	C	D	**全部**
0	5023	1610	8965	2280	**17878**
1	2281	3469	1317	2496	**9563**
2	1284	3663	441	1364	**6752**
3	439	2918	99	511	**3967**
4	156	1983	24	178	**2341**
5	28	1240	8	28	**1304**
6	8	606	0	14	**628**
7	1	268	0	1	**270**
8	1	81	0	0	**82**
9	0	18	0	0	**18**
10	0	5	0	0	**5**
11	0	1	0	0	**1**
12	0	0	0	0	**0**
標準差	**0.75**	**2.52**	**0.24**	**1.12**	**1.34**

表三 樣本/誤差百分比分析表

分區	誤差值小於 5%樣本數	超過 80%的樣本數, 其最大之誤差值
全部	80.68%	5%
區域 A	74.58%	6%
區域 B	72.19%	6%
區域 C	91.61%	0%
區域 D	91.25%	4%



圖五 全部樣本差值與百分比結果圖

為 ±1.34 單位長度 $CI/\sqrt{x} = 0.0195 \leqq 0.025$。即代表本電腦模擬已進入 95%信賴區間。其中差值為試誤型演算法與確切型演算法所得到的路徑長度差。由於試誤型演算法所計算出之結果，必然會大於確切型演算法所得出之結果。

因此信賴區間中之 0 -1.34 部份區間將不適用。由此吾人可說本文所提出之試誤型演算法在 95%的信心水準下，根據所採用之隨機樣本依試誤型演算法所計算出之直坦納樹的長度與確切值之差為 0~1.34。即可宣稱在樣本空間下，試誤型演算法的執行結果與確切解相差不大於 1.34 個單位長度。表三中則顯示出分區的模擬結果。由其結果可知除了在區域 B 中的標準差較大外，其餘三區均小於全部統計結果的標準差。分析其主要原因，係第二節中所提出之試誤型於計算節點數較多時將會與確切解有較大的誤差，但其最差情況下亦不會超過 2.52 個單位長度。

統計學中常用到的 80-20 原則，其意謂著統計的結果有 80%的樣本都能落在 20%的條件下。本節亦將依此一原則對全部樣本及分區樣本之結果分佈情形做一分析。如表三所示，在全部樣本的統計結果中，誤差值小於 5%的樣本數佔全部總樣本數的 80.68%；而超過 80%的樣本，其所產生的誤差值均小於 5%。

五、結論

本文提出了一個可適用於網狀網路上的可容錯直線式史坦納樹試誤型演算法，其空間複雜度為 $O(pN)$，時間複雜度為 $O(p^2N)$，其中 p 為網狀網路中欲相互連接成一個網路的節點總數，N 為網狀網路上所有節點的個數。另外，為驗證試誤型演算法的效能，本文亦提出了一個可於指數時間內完成給定 Z 的史坦納樹的確切型演算法。經由統計、分析後，試誤型演算法於 10×10 大小的網狀網路上以電腦模擬試誤型演算法及確切型演算法的結果。在有效樣本數 42809 個隨機產生樣本下，試誤型演算法在 95%的信賴區間下所找出之路徑長度與確切型演算法之標準差為 1.34 單位。且在全部的有效樣本中超過 80%的樣本，其以試誤型演算法執行所求得之近似解與確切解之誤差率小於等於 5%。因此本演算法可以極小的時間、空間成本, 快速地找出直線式史坦納樹的近似最佳解。

六、參考文獻

[1] A. V. Aho, M. R. Garey, and F. K. Hwang, "Rectilinear Steiner trees: efficient special case algorithms," *Networks*, vol. 7, pp. 37-58, 1977.

[2] M. R. Garey and D. S. Johnson, "The rectilinear Steiner problem is NP- complete," *J. SIAM Appl. Math.*, vol. 32, pp. 826-834, 1977.

[3] M. Hanan, “Net wiring for large scale integrated circuits,” *IBM Res. Report* RC 1375, 1965.
[4] M. Hanan, “On Steiner’s problem with rectilinear distance” *J. SIAM Appl. Math.*, vol. 14, pp. 255-265, 1966.
[5] J. Hesser, R. Manner, and O. Stucky “Optimization of Steiner trees using genetic algorithms” *Proc. 3rd Int. Conf. Genetic Algorithms*, pp. 231-236, 1989.
[6] F. K. Hwang, “On Steiner minimal trees with rectilinear distance,” *SIAM J. Appl. Math.*, vol. 30, pp. 104-114, 1978.
[7] F. K. Hwang, “An *O*(*n log n*) algorithm for suboptimal rectilinear Steiner trees,” *IEEE Trans. Circuits and Systems*, CAS-26, pp. 75-77, 1979.
[8] Gene Eu Jan and Ki-Ying Chang, “An *O*(*N*) Shortest Path Algorithm on Raster,” *Technical Report* CS01001, Department of Computer Science, National Taiwan Ocean University, Taiwan, Jul. 2001.
[9] Gene Eu Jan, Ming-Bo Lin and Yung-Yuan Chen, “Computerized Shortest Path Searching for Vessels,” *Journal of Marine Science and Technology*, Vol. 5, No. 1, 1997, pp. 95-99.
[10] B. A. Julstrom, “A genetic algorithm for the rectilinear Steiner problem,” *Proc. 5th Int. Conf. Genetic Algorithms*, pp. 474-480, 1993.
[11] A. Kapsalis, V. J. Rayward-Smith, and G. D. Smith, “Solving the graphical Steiner tree problem using genetic algorithms,” *Journal of the Operational Research Society*, vol. 44, no. 4, pp. 397-406, 1993.
[12] R. M. Karp, “Reducibility among combinatorial problems,” in R. E. Miller, J. W. Thatcher (eds.), *Complexity of Computer Computations*, Plenum Press, New York, pp. 85-84, 1972.
[13] P. Korhonen, “An algorithm for transforming a spanning tree into a Steiner tree,” *Proc. 9th Int. Math. Progr. Symp.*, Budapest *1976*. North-Holland, Amsterdam, pp. 349-357, 1979.
[14] C. Y. Lee, “An algorithm for path connections and its applications,” *IRE Trans. Electron. Computer*, vol. EC-10, pp. 346-365, Sept. 1961.
[15] J. H. Lee, N. K. Bose, and F. K. Hwang, “Use of Steiner’s problem in suboptimal routing in rectilinear metric,” *IEEE Trans. Cir. and Sys.*, CAS-23, pp. 470-476, 1976.
[16] Z. A. Melzak, “On the problem of Steiner,” *Canad. Math. Bull.*, vol. 4, pp. 143-148, 1961.
[17] R. C. Prim, “Shortest connection networks and some generalizations, *Bell System Tech. J.*, vol. 36, pp. 1389-1401, 1957.
[18] J. M. Smith, “An *O*(*n log n*) heuristic for Steiner minimal tree problems on the Euclidean metric,” *Networks*, vol. 11, pp. 23-39, 1981.
[19] J. M. Smith, D. T. Lee, and J. S. Liebman, “An *O*(*n log n*) heuristic algorithm for the rectilinear Steiner minimal tree problem,” *Eng. Optimi.*, vol. 4, pp.179-192, 1980.
[20] J. M. Smith and J. S. Liebman, “Steiner trees, Steiner circuits and the interference problem in building design” *Eng. Optimization*, vol. 4, pp. 15-36, 1979.
[21] P. Winter, “An algorithm for the Steiner problem in the Euclidean plane” Networks, vol. 15, pp. 323-345, 1985.
[22] P. Winter, “Steiner problem in networks: a survey,” *Networks*, vol. 17, pp. 129-167, 1987.
[23] 范啟明，〝網狀網路上可容錯之直線式史坦那樹試誤型演算法〞，海洋大學資訊科學研究所碩士論文，民國 91 年 7 月。