試誤型史坦那樹演算法及電子設計自動化應用
PART I：2D 平面中試誤型可容錯八向史坦那樹演算法
Obstacle-Avoiding Heuristics for Steiner Tree Problem in EDA
PART I：Heuristics for Steiner Tree in 2D 8-directional Algorithm

林琮徨
國立臺灣海洋大學
資訊工程學系
m92570013@mail.ntou.edu.tw

詹景裕
國立臺北大學
資訊工程學系
gejan@mail.ntpu.edu.tw

黃元欣
國立臺灣海洋大學
資訊工程學系
shin@mail.ntou.edu.tw

摘要

找尋史坦納樹問題一直以來都是計算機科學領域中一個很重要的課題。其應用領域廣泛，本文所提出 X 架構上 2D 平面之試誤型史坦那樹演算法演算法主要改良 Luo 與 Hanan 的方法，並加上 Prime 最小伸展數演算法的應用，成為八向米字型線段延伸的新演算法。本演算法在無障礙空間下的時間與空間複雜度為 $O(p^3)$以及 $O(pN)$，在有障礙物空間下的時間與空間複雜度為 $O(N+p^3)$以及 $O(pN)$，其中 p 為 Z 集合總數，N 自由節點總數。

關鍵字：X 架構、最小伸展樹、史坦納樹、八向米字型線段延伸。

ABSTRACT

Heuristics for Steiner Tree Problem in Design Automation of System for a Steiner minimal tree for a set Z of vertices on X architectures is a tree, which interconnects Z using horizontal, vertical and oblique segments of shortest possible total length. In this paper, The proposed algorithm bases on the concepts of Luo's and Hanan's algorithms and Prim's Minimal spanning tree(MST) algorithm to obtain the heuristic Steiner minimal tree(SMT) on X architectures.

The SMT algorithm without obstacles has the time and space complexities of $O(p^3)$ and $O(pN)$, respectively, The SMT algorithm with obstacles has the time and space complexities of $O(N+p^3)$ and $O(pN)$, respectively, where N and p are the numbers of free and terminal vertices, respectively, $p<N$.

Keywords: X architectures, Minimal spanning tree, Steiner minimal tree.

一、緒論

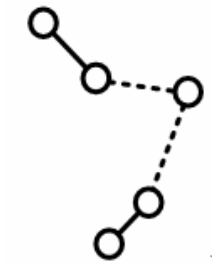
舉凡網路上的多重廣播路徑之找尋或網狀架構(mesh)上的容錯路徑、印刷電路版(printed circuit board, PCB)的電流網路連接、以及超大型積體電路內的路由路徑(VLSI routing)找尋等 [28]，均為史坦納樹問題的應用領域。

過去數十年來，許多數學家、資訊科學領域的研究者，均針對史坦納樹問題投入不少心力。與史坦納樹相關問題之研究也有不少的成果被發表，但是在史坦納樹問題的特例中：X 架構史坦納樹問題(Steiner tree problem on X architecture)，舉例而言，在印刷電路版及超大型積體電路的應用上，極需要在已存在現有網路的情形下，找出另一組電流網路的連接路徑，此時，可容錯或可避開障礙物(即現存網路)的斜線式史坦納樹演算法便顯得非常急迫。

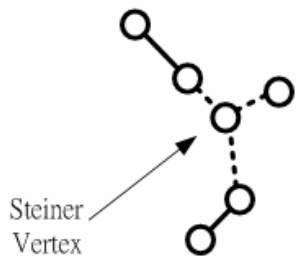
因此，本文即將針對在一具有障礙物的 X 架構中，找出可容錯的 X 架構史坦納樹。在找尋的方式上，係提出試誤型演算法來減少解決該問題所需的時間。

二、相關文獻回顧

網路中的史坦納樹問題(Steiner problem in networks; SPN)係指於一個歐式平面(Euclidean plane)中，找出可展開(spanning)給定節點集合(set of vertices)的最短路徑網路。其與最小伸展樹 MST 的問題大致相似，但其不同點為最小伸展樹中構成路徑的邊集合(set of edges)中的邊(edges)僅允許由節點(vertices)連接至節點，而構成史坦納樹路徑的邊型態除了可由節點連接至節點的邊構成外，尚可由其他任意自由節點來當作輔助節點，使得連接整個



(a) Minimal Spanning Tree



(b) Steiner Minimal Tree

圖 1 最小伸展樹與史坦納樹之差異示意圖

樹後的總長度最小，吾人稱其為史坦納節點(SV)。

如圖一所示，圖一中圓點表示欲相連接之節點，實線及虛線則為連接該節點集合的邊集合。圖一(a)為以最小伸展樹定義中所允許的路徑連接方法將給定的節點連接後所形成之連接圖(connected graph)；而圖 1(b)則為給定與圖 1(a)相同的節點集合，以最小史坦納樹定義的路徑形態所得出之連接圖形，比較圖中虛線部份，可明顯看出最小伸展樹與史坦納樹之差異處。即最小伸展樹僅可以允許路徑為節點連結至節點的形態，史坦納樹則允許新增節點連結至現有路徑之路徑形態。

史坦納樹問題之正式的數學定義為：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: $E \rightarrow R$ 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)，$0 < i, j \le p$，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。如圖 1(b)中之史坦納節點即為 S 節點，其餘則為 Z 節點。

關於 SPN 的問題有許多確切型演算法(exact algorithm)已被提出，如 Melzak [21], Cockayne [6], Boyce 及 Seery [4], Boyce 與 Winter 等 [3]，[26]。不過，由於 SPN 的問題太過於複雜，使得前述的諸多確切型演算法僅能在合理的時間內解決連接節點數 p 較少的問題，當連接節點數過多的時候，其所需的處理時間將為天文數字而無法令人接受。

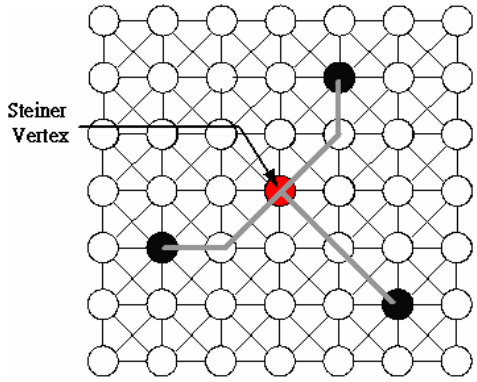


圖 2 Steiner minimal tree on X architecture, length of tree= 7.656。

Karp 以及 Foulds 分別證明了 SPN 問題係屬於 NP-complete [7]，[17]。

因此，為解決較多的連接節點數，試誤型演算法變得極為重要。目前，已有許多針對 SPN 問題而提出的試誤型演算法已被提出。其中可獲得較佳結果且執行效率也較佳的 SPN 試誤型演算法係由 Smith 等人所提出 [23]，其他的試誤型演算法則有 Chang 及 Korhonen 等 [5]，[18]。

X 架構式史坦納樹(Steiner tree on X architecture; STX)為史坦納樹(Steiner Tree; ST) 的一個特例。其係指於一個 X 架構平面上，利用垂直與水平線段以及斜線線段，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。令 $G(Z)$ 為由通過每個 Z 節點之水平垂直與斜線所產生的一個網格狀平面圖(如圖 2 所示)，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 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)。如圖 2 中之黑色節點為 Z 節點，而其兩線相交之點則為 S 節點。於本例中子網路 G_Z 之總成本即為連結所有 Z 節點之線段總長度，即為 7.656。

對於每一個(Rectilinear Steiner Tree; RST)的問題，均可以 SPN 的演算法來加以解決 [27]。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，Garey 等人已證明了 RST 問題亦為 NP-complete [8]。因此，許多解決 RST 問題的試誤型演算法亦已被提出。

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是與 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 提出的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 [9]，其中 p 為節點集合 Z 中的節點個數，其將 Prim 的最小伸展樹 MST 演算法應用至 Z 上 [22]，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 [25]，Lee 等人則提出 $O(p^2)$的試誤型演算法 [19]，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 [13]。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 $O(p\,logp)$ [24]。除此之外，Hwang 並証明了以 MST 方式來找尋史坦納最小樹 SMT 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 [12]。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 [11]，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、Julstrom、Kapsalis 等所提出的演算法均屬於之 [10]，[15]，[16]。

Jan 於近年提出一個有效率的 SMT 試誤型演算法 [14]，利用 Lee 最短路徑演算法並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)，並以遞迴方式連結所有的終端節點(Terminal Vertices)，並將此演算法的時間複雜度控制在 $O(p^2N)$，其演算法名稱為 Higher Geometry Maze Routers Algorithm (HGMR)。

Lou 等人最近提出一個不同於先前 Areibi 的觀念 [2]，[20]，Areibi 的方法是在直線式無障礙物自由空間中，先尋找所有的 Hannan grid，再嘗試將其中每一個 Hanan vertices(HV)當作整個 ST 的潛在可能的成員之一，去計算新節點集合的 MST 樹長，並且將每一個 HV 增益量記錄下來，當所有的 HV 計算完成後，選取其增益量最大的點當作 SV，重複上述步驟產生 SV 有 HV 的增益量為負為止。不同於先前 Areibi 的目的，Lou 演算法是在 X 架構上並且可存在障礙物的。

三、參數與變數表

因為本文結構複雜，所使用之變數符號之定義，就如表 1 所示。

表 1 參數與變數表

變　數	說　明
G	為由節點(vertices)及邊(edges)所連接而成的圖(graph)。
V	節點集合(set of vertices)，其中包含有 $\lvert V\rvert$ 個節點。。
E	邊集合(set of edges)，其中包含有 $\lvert E\rvert$ 個邊。
Z	為 V 中欲相互連接成一個網路(network)的節點所成之節點集合。其中包含有 p 個節點。在 Z 中之節點稱為 Z 節點(Z-vertices)。且 Z 包含於 V。
p	Z 節點集合中之節點總數。
C	成本函數(cost function)，C: $E\rightarrow R$ 為一由邊集合 E 映至非負實數 (non-negative real number)集合 R 的函數。
G_Z	屬於 G 之子圖(sub graph)，並包含所有 Z 節點。
S	史坦納節點 (Steiner vertices)所成的節點集合。
$C(G)$	G 的總成本，其成本因子為邊的長度(length of edges)，所以總成本 $C(G)$ 為 G 中所有邊長度的總和。
$V_{i,j}$	vertices，儲存 $I\times J$ 網狀網路內的節點狀態，其中 $(i,\ j)$ 為二維陣列之索引值，$0\le i\le I$ ，$0\le j\le J$
$V_{i,j}^{Steiner}$ $V_{i,j}^{Steiner}(l)$	Steiner vertices，經由米字型延伸所得之交叉點，儲存 $I\times J$ 網狀網路內的節點狀態，其中 (i,j) 為二維陣列之索引值，當以 $V_{i,j}^{Steiner}(l)$ 表示時，表其為存取 $V^{Steiner}(l)$ 上索引值 $(i,\ j)$ 位置上之值，$0\le i\le I$ ，$0\le j\le J$ 。
Atb $Atb_{i,j}$	Attribution，描述整個地圖各點的屬性，其值介於{0, 1, 2 }之間。其中值為 0 時表示該節點為自由節點，1 表其為障礙節點，2 則為欲相互連接的節點。其中 $(i,\ j)$ 為二維陣列之索引值，$0\le i\le I$ ，$0\le j\le J$ 。
$V_{i,j}^{SM}$	vertices of Subset Mesh，描述終端點延伸範圍內的屬性，預設為 False，其值設為 True,則表示在指定範圍之內，其中 (i,j) 為二維陣列之索引值，$0\le i_{Min}^{SM}\le i\le i_{Max}^{SM}\le I$ ，$0\le j_{Min}^{SM}\le j\le j_{Max}^{SM}\le J$ 。
	Array of Distance for *Z*-vertices，儲存以 Z 中第 l 個節點之位置為起點，經由所有終端點米字型

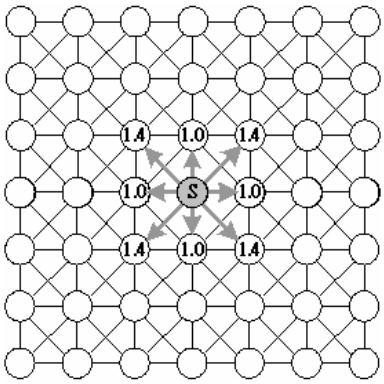
$AD(l)$ $AD_{i,j}(l)$	延伸後所得之等距離圖，$0 \le l < p$。$AD(l)$陣列之大小為 $I \times J$。當以 $AD_{i,j}(l)$表示時，表其為存取 $AD(l)$上索引值(i, j)位置上之值，$0 \le i \le I$，$0 \le j \le J$。
$AD(\eta,\iota)$ $AD_{i,j}(\eta,\iota)$	Array of Distance for vertices，儲存以 Z 中以(i,j)節點之位置為起點，經由所有終端點米字型延伸後所得之等距離圖。$AD(\eta,\iota)$ 陣列之大小為 $I \times J$。當以 $AD_{i,j}(\eta,\iota)$ 表示時，表其為存取 $AD(\eta,\iota)$ 上索引值 (i, j) 位置上之值，$0 \le i \le I$，$0 \le j \le J$，$0 \le \eta \le I$，$0 \le \iota \le J$。
LL^Z $LL^Z(l)$	Linked List of Z-vertices，儲存記錄 $v_{i,j}$ 中值為 2 之節點，即 Z 節點。當以 $LL^Z(l)$表示時，係指取出 LL^Z 中第 l 個節點 $v_{i,j}$。
$e_{v_{i,j},v_{i',j'}}$	edges，儲存由 $v_{i,j}$ 連結至 $v_{i',j'}$ 的邊。
LL^{RST} $LL^{RST}(l)$	Linked List of RST，為一儲存計算後史坦納路徑的鏈結串列。其內之資料型態由邊 $e_{v_{i,j},v_{i',j'}}$ 所組成。當以 $LL^{RST}(l)$表示時，係指取出 LL^{RST} 中第 l 個邊 $e_{v_{i,j},v_{i',j'}}$。
$v^{SM}_{(imin,jmin)}$ $v^{SM}_{(imax,jmin)}$ $v^{SM}_{(imin,jmax)}$ $v^{SM}_{(imax,jmax)}$	為設定終端點延伸範圍的座標位置
$(i_{Steiner}(l), j_{Steiner}(l))$	*Steiner* 節點的二維陣列之索引值，當以$(i_{Steiner}(l),j_{Steiner}(l))$表示時，係指取出第$(l)$個$(i_{Steiner}, j_{Steiner})$，$0<i_{steiner}<I$，$0<j_{steiner}<J$。
$*v^{temp}$	儲存計算時所用的暫存節儲存計算時所用的暫存節點 $v_{i,j}$ 之位址。
LL_{index}	Linked List，記錄排序之相鄰網格串列
index	LL 串列指標陣列的索引值，$0<index<2$。
$Atb'_{i,j}$	vertices，係為輔助計算用時的暫存資料結構，其內儲存 $I \times J$ 網狀網路內的節點狀態，其值介於{0, 1, 2 }之間。其中值為 0 時表示該節點為自由節點，1 表其為障礙節點，2 則為欲相互連接的節點。其中(i, j)為二維陣列之索引值，$0 \le i \le I$，$0 \le j \le J$。
$Num^{obstacle}$	Number of obstacles，儲存 $v_{i,j}$ 中 Atb 值為 1 的節點總數，即為所有障礙節點之總數。
$Num^Z = p$	Number of Z-vertices，儲存 Atb 值為 2 的節點總數，即為所有欲相互連接的 Z 節點之總數。
$LMST_{initial}$	initial Length of Minimal Spanning Tree，對 Z 集合中所有點作 Prim 的最小伸展樹演算法所得到的總樹長。
$LMST_{i,j}$	temporal Length of Minimal Spanning Tree，將點(i,j)暫時加入 Z 集合中，而以 Prim 的最小伸展樹演算法計算 Z 所得到的總樹長值。
$DLMST_{i,j}$	Difference of Lengths of Minimal Spanning Trees，其值為 $LMST_{initial} - LMST_{i,j}$。
IT $IT_{i,j}$	Improvement Table，為儲存 $DLMST_{i,j}$ 所用，當以 $IT_{i,j}$ 表示時，係表存取 IT 上索引值為(i, j)位置上之值。其中(i, j)為二維陣列之索引值，$0 \le i \le I$，$0 \le j \le J$。

四、無障礙物史坦那樹演算法

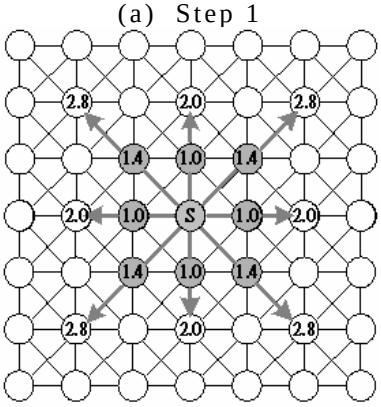
(一)無障礙物八向史坦那樹演算法介紹

本文所使用的演算法利用觀念利用 Hanan 演算法的觀念，將原本平面四向延伸做了修正，更改為平面八向米字型延伸，如圖 3 所示，圖中 S 為起點(source)，其距離值為 0，地圖的盡頭則為終點(destination)。於圖 3(a)Step 1 時，以 S 為基點向外延伸，所得之鄰格之距離值為基點之距離值（此時為 0）加 1 以及 $\sqrt{2}$ (斜線方向)；圖 3(b) Step 2 時則以距離值以 Step 1 所延伸的結果為網格為基點，分別向其鄰格同方向呈現米字型擴散；圖 3(c)與圖 Steps 3 以相同的方式進行，直至擴散至整張地圖完為止。

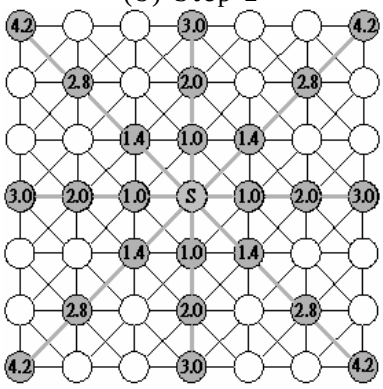
此時終點的距離值經由米字延伸的累加，得知所有即在 X 架構上距離起始點的所有米字延伸距離單位。關於多點的米字型延伸，如圖 4(a)中 $Z1$ 與 $Z2$ 為我們所欲連接的節點(Terminal vertices)，$Z1$ 的米字型延伸區域在圖中網格顯示為上半圓橫斜



(a) Step 1



(b) Step 2



(c) Finish

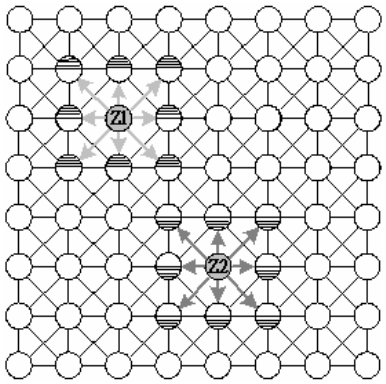
圖 3 本文所提出八向米字型延伸示意圖。

線，*Z*2 則是圖中網格顯示為下半圓橫斜線，於 Step1 時其 *Z*1 與 *Z*2 分別向其鄰格共八個方向延伸，如圖 4(b)中 Step 2 時則以距離值以 Step 1 之 *Z*1 與 *Z*2 所延伸的結果為網格為基點，分別向其鄰格同方向呈現米字型擴散，在地圖中網格全部顯示為全滿的橫斜線部分，則表示為 *Z*1 與 *Z*2 的重疊節點，並且得知 *Z*1 與 *Z*2 的最短路徑，這些重疊節點也是我們可能潛在的史坦那節點。

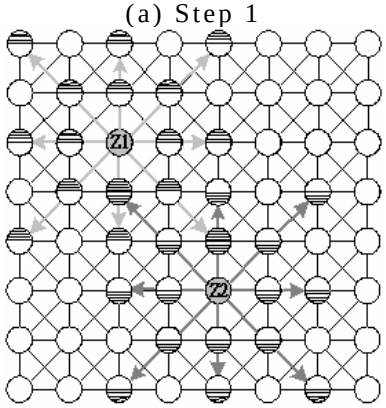
本文所提的演算法在無障礙空間時，只需要使用座標軸公式即可求的節點之間的距離;
距離公式：

$$min(|i-i'|,|j-j'|)\sqrt{2}+max(|i-i'|,|j-j'|)-min(|i-i'|,|j-j'|)$$

如圖 5 所示，圖中 *Z*1 座標值為(2,2)，*Z*2 座標值為(6,3)，將兩座標值代入公式計算過程結果如下



(a) Step 1



(b)Step 2

圖 4 八向米字型之延伸說明圖。

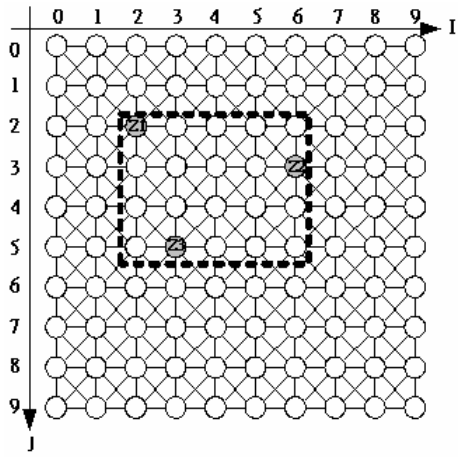


圖 5 所有 *Z* 節點的範圍圖

$$\text{距離} = min(|2-6|,|2-3|)\sqrt{2} + max(|2-6|,|2-3|) - min(|2-6|,|2-3|)$$
$$= min(4,1)\sqrt{2} + max(4,1) - min(4,1)$$
$$= (1)\sqrt{2} + 4 - 1$$
$$= 4.4$$

由於是無障礙空間下的關係，因此我們可以利用所給定的 *Z* 節點來進行設定一個固定的收尋範圍，好處是可以節省收尋時間以及空間，其設定範圍座標公式如下：

$V^{SM}_{(imin,jmin)}$ $V^{SM}_{(imax,jmin)}$ $V^{SM}_{(imin,jmax)}$ $V^{SM}_{(imax,jmax)}$
其收尋範圍位於此四點座標所連接而成的矩陣之中，如圖 5 所示，圖中有 Z1、Z2、Z3 三個節點，利用此三點來設定米字型延伸與所要收尋 SV 的範圍，將這三個節點代入公式即可求出收尋範圍，因此圖 5 所設定的米字型延伸收尋範圍則在(2,2),(2,5),(6,2),(6,5)此四節點內。

本演算法使用 Hanan 演算法的觀念，將原本平面四向延伸做了修正，更改為平面八向米字型延伸，本演算法是應用於地圖無障礙情形下，因此只要使用座標軸，即可設定每個 Z 節點的米字型延伸範圍，減少不必要的延伸與重疊節點，並且可求出各 Z 節點之間的最短距離，針對每個 Z 節點間第一個重疊的米字型延伸節點當作潛在的史坦納節點來一一計算其個別的 MST，進而求得 IT，僅產生一個 IT，利用 IT 中的數值當成史坦那測試節點，將這些節點使用 MST 遞回產生出史坦那節點，直到 IT 中所有的節點計算完為止，本演算法即結束。

(二)無障礙物八向史坦那樹演算法

BEGIN

Step 1 設定所有終端節點矩陣及米字型延伸的範圍

Step 1.1 利用座標軸設定範圍：

$V^{SM}_{(imin,jmin)}, V^{SM}_{(imax,jmin)}, V^{SM}_{(imin,jmax)}, V^{SM}_{(imax,jmax)}$
此四點所形成的矩陣之內是我們座標軸所設定的範圍。

Step 1.2 For ($j=V^{SM}_{jmin}$; $j<=V^{SM}_{jmax}$;j++)

For ($i=V^{SM}_{imin}$; $i<=V^{SM}_{imax}$; i++)

{

$V^{SM}_{i,j}$ =True

}

End {For}

End {For}

/* 假設 $V^{SM}_{i,j}$ 為 True 時，終端點的延伸重疊節點僅在此範圍內才計算*/

Step 2 使用座標軸求所有終端節點間最短距離與可能的 Steiner 節點

For $j=V^{SM}_{jmin}$ to V^{SM}_{jmax}

For $i=V^{SM}_{imin}$ to V^{SM}_{imax}

當 $v_{i,j}$ 屬於 Z 時，先產生一個相對應的 $AD(\eta,\iota)$，然後將此 $v_{i,j}$ 依序存入 LL^Z，在此 $AD(\eta,\iota)$ 中，以 $v_{i,j}$ 為起點執行米字型延伸。

End {For}

End {For}

Step 3 計算出 Z 集合的最小伸展樹長

從 Z 集合中 $AD(l)$中，可以得知每一個 Z_l 點到其他所有 $Z_{l'}$點的距離。所以可以根據 Prim 的最小伸展樹演算法求得最小伸展樹的樹長 $LMST_{initial}$。

Step 4 求出 Improvement Table

For $j=V^{SM}_{jmin}$ to V^{SM}_{jmax}

For $i=V^{SM}_{imin}$ to V^{SM}_{imax}

Step 4.1

若 $V^{Steiner}_{i,j}$ 為指定範圍內重疊節點，則暫時將 $V^{Steiner}_{i,j}$ 當作是 Z 集合中的一點，然後再利用 Prim 的最小伸展樹演算法計算求出加入新節點後的樹長 $LMST_{i,j}$。

Step4.2

$DLMST_{i,j}=(LMST_{initial}-LMST_{i,j})$的值存入相對應的 $IT_{i,j}$ 位置中，然後將 $V^{Steiner}_{i,j}$ 移除原本的 Z 集合。若此值為正，則代表此點 $V^{Steiner}_{i,j}$ 具有使原本的 Z 集合的總樹長降低的意義。

End {For}

End {For}

Step 5 產生 Steiner vertex

搜尋整個 IT 所有的值，取其最大的值，若 $(i_{Steiner}(l),j_{Steiner}(l))$ 為其相對應的位置，即為 Steiner vertex 所在的座標位置，暫將所產生的 Steiner vertex 加入原本的 Z 集合中，使用原本 Z 集合的總樹長減去 Z 集合加入 Steiner vertex 的總樹長，結果大於 0，則修改 Z，然後刪除 $IT_{i,j}$，結果小於 0，直接刪除 $IT_{i,j}$，重複此步驟，遞迴直到 IT 所有的值都計算完為止。

Step 6 產生 Steiner Minimal Tree

將 Step 5 的所得到的 Z 集合，執行一次 Prim 最小伸展樹演算法，所得到的結果利用米字型延伸的方法，將整個樹的所有的路徑追蹤完成，整個演算法即完成。

END{無障礙物八向史坦那樹演算法}

(三) 效能分析

1.空間複雜度分析

關於本文所提出 SMT 的空間複雜度分析，最原始的地圖必須使用一個 $I\times J$ 二維矩陣 $v_{i,j}$。在初始化的過程中，我們使用了 p 個 $I\times J$ 二維矩陣 $v_{i,j}$ 來儲存各個節點所洪氾的狀態，另外還需要一個 $I\times J$ 的二維矩陣儲存 IT 的值。再者輔助計算用的資料結構中，空間中要求最大的 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：

$$SC_{total} = SC_v + SC_{AD} + SC_{IT} + SC_{auxiliary}$$
$$= N + pN + N + 2N$$
$$= O(pN)$$

2.時間複雜度分析

關於本文所提出 SMT 的時間複雜度分析，可分為五個階段來探討，Step 1 為設定座標軸求每個米字型延伸的範圍必須設定在 $v^{SM}_{(imin,jmin)}$，$v^{SM}_{(imax,jmin)}$，$v^{SM}_{(imin,jmax)}$，$v^{SM}_{(imax,jmax)}$ 四個座標點之內，由於四個座標僅作四次設定，因此時間複雜度 TC_{Step1}：

$$TC_{Step1} = O(1)$$

Step 2 為求每個米字型延伸的距離矩陣，計算各個 *Z* 節點的等距離圖，其時間複雜度於計算個別的等距離圖時，其時間複雜度為 *O*(*p*)，但因為需計算 *p* 次，因此時間複雜度為 *O*(p^2)。因此 Step 2 的時間複雜度 TC_{Step2}：

$$TC_{Step2} = p \times p = O(p^2)$$

在 Step 3 中計算出 Z 集合中的最小伸展樹，根據 Prim 的 MST 演算法所需要的時間複雜度為 $O(p^2)$，所以 Step 3 的時間複雜度 TC_{Step3}：

$$TC_{Step3} = O(p^2)$$

在 Step 4.1 求出 Improvement Table 的過程中，因為每一次執行 Prim 的最小伸展樹演算法必須花費 $O(p^2)$的時間，而其過程為假設每個進行米字型延伸的第一個重疊節點為 *Z* 集合中其中一點，所以最差的情形之下，總共必須執行 8×O(*p*)次，因此 Step 4.1 的時間複雜度 $TC_{Step4.1}$ 為：

$$TC_{Step4.1} = 8p \times (p+1)^2 = O(p^3)$$

在 Step 4.2 中，從 Improvement Table 中搜尋 Steiner vertex，由於每次搜尋必須 $v^{SM}_{(imin,jmin)}$，$v^{SM}_{(imax,jmin)}$，$v^{SM}_{(imin,jmax)}$，$v^{SM}_{(imax,jmax)}$ 從四個座標點之內的空間中搜尋，因此 Step 4.2 的時間複雜度 $TC_{Step4.2}$ 為：

$$TC_{Step4.2} = 8 \times p = O(p)$$

因此 Step4 的總時間複雜度 TC_{Step4} 為：

$$TC_{Step4} = TC_{Step4.1} + TC_{Step4.2}$$
$$= O(p^3) + O(p) = O(p^3)$$

第 Step 5 中，必須將 Steps 3-4 的所計算得到的結果，將所產生的 Improvement Table 中的數值當成史坦那測試節點，在最多的情況下會有 8(*p*)個史坦那測試節點，這些節點使用 MST 遞回產生出史坦那節點，因此 Step 5 的時間複雜度 TC_{Step5} 為：

$$TC_{Step5} = (TC_{Step3} + TC_{Step4}) + (8p \times p^2)$$
$$= O(p^2) + O(p^3) + (p^3)$$
$$= O(p^3)$$

綜合 Steps 1-5 的時間複雜度可知道本演算法的時間複雜度應該為：

$$TC_{total}$$
$$= TC_{Step1} + TC_{Step2} + TC_{Step3} + TC_{Step4} + TC_{Step5}$$
$$= O(1) + O(p^2) + O(p^2) + O(p^3) + O(p^3)$$
$$= O(p^3)$$

(四)演算法執行範例與比較

1.演算法範例

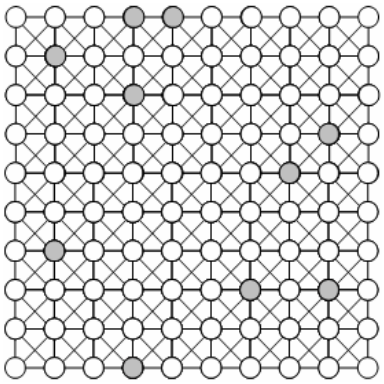
圖 6(a)是我們的地圖初始狀態，這是一張 10×10 的矩陣，淺灰色節點所代表的是欲連結的終端節點，本範例中初始設定為 10 個隨機點，而白色的部分是可以通行的自由節點，圖 6(b)~(d)中黑色的線段代表根據演算法所計算的結果，我們使用了相同的例子來執行 3 種不同的演算法，在這此分析 3 種演算法的差異性，在圖 6(b)中使用 HGMR 平面可容錯八向式史坦那樹試誤型演算法得到的結果，從中可以發現在利用 HGMR 史坦那樹演算法所計算出的結果中，所選擇路徑的結果，所產生一些可以很輕易的發現的非最佳解，反之圖 6(d)中使用本演算法的執行結果中很難去發現這樣的例子，而且在圖 6 例子中，所執行樹長的結果剛好等於圖 6(c)Lou's 平面可容錯八向式史坦那樹試誤型演算法。

從樹長結果來看，Lou's SMT Algorithm 與本文 SMT Algorithm 優於 HGMR SMT Algorithm，明顯的 Lou 與本文所提出的史坦那樹演算法，能夠有效的找出降低樹長的史坦那節點，本文的史坦那樹演算法也充分利用了 X 架構的特性，相較於 Lou 來看，能夠更快速的找到所需的史坦那節點。這裡的例子中使用 The SMT Algo.的演算法的結果會比 HGMR 平面可容錯八向式史坦那樹演算法結果要好，而且接近等於 Lou's 平面可容錯八向式史坦那樹試誤型演算法的結果。

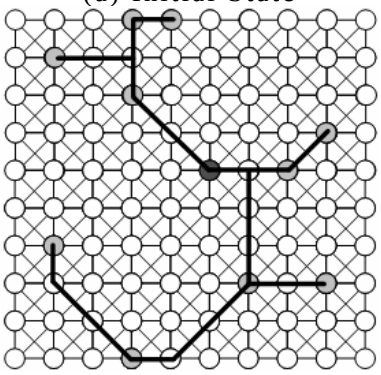
我們可以很清楚的在圖 6 的範例中所觀察到，使用 Lou 的演算法所得到的結果可以獲得比較短的路徑長度，不過它所付出的時間以及空間的成本卻相對的比其他兩種演算法來的大，本文所提出的無障礙物史坦那樹演算法，不僅近似於 Lou 的路徑長度，付出的時間最短且空間的成本也優於 Lou 與 HGMR 的 SMT 演算法，整理結果如表 2 所示。

表 2 無障礙物演算法分析比較表
p 為 *Z* 集合總數，*N* 自由節點總數

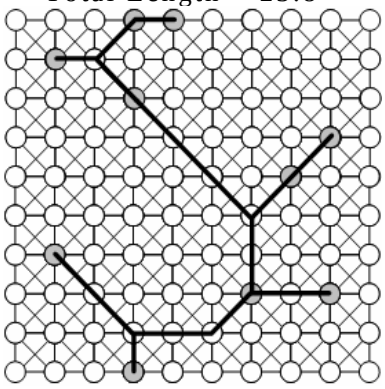
Items / Algo.	Time Complexity	Space Complexity	Total Length of SMT
HGMR SMT	$O(p^2N)$	$O(pN)$	Near-Optimal
Lou's SMT	$O(N^2+p^3N)$	$O(N^2)$	Shorter
The SMT Without Obstacles	$O(p^3)$	$O(pN)$	In-between



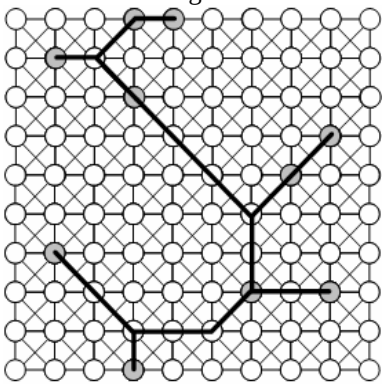
(a) Initial State



(b) Result of HGMR SMT Algorithm
Total Length = 23.8



(c) Result of Lou's SMT Algorithm
Total Length = 23.0



(d) Result of The SMT Algorithm
Total length = 23.0

圖 6 Difference between the results of HGMR, Lou's and The SMT Algo.

五、有障礙物史坦那樹演算法

(一)有障礙物八向史坦那樹演算法介紹

本小節介紹平面中有障礙物八向式史坦納樹演算法，本文提出在平面式有障礙物 X 架構上面所計算之史坦那樹演算法。這裡我們使用使用 HGMR Algorithm 中的洪氾特性將本文所提之米字型延伸能夠避碰障礙物，由於米字型延伸有八個方向，我們的目的是在有障礙物平面地圖上讓每個終端節點的米字型延伸都可以延伸到地圖的盡頭，因此我們用反相思考的方式，以地圖每個方向的盡頭(東、南、西、北、東北、東南、西南、西北)使用整排的列座標或行座標當作 HGMR Algorithm 中的起始點進行整張地圖的洪氾，如此一來在有障礙物空間下每個終端節點都能夠進行米字型延伸到地圖八個方向的盡頭，終端點可利用累加的方式進行米字型延伸，進而找取 MST 與可能的 SV，接著求得 IT，再利用貪進法的觀念求得史坦納節點，直到無法求出史坦納節點為止。

如圖 7 所示，圖 7 地圖西邊的盡頭進行整排的洪氾範例以地圖八個方向的盡頭，東、南、西、北在此使用西邊的洪氾為範例，其餘三個方向以此類推。如圖 8 所示，東北、東南、西南、西北則以西北為範例，其餘三個方向以此類推。使用整排的列座標或行座標當作 HGMR Algorithm 中的起始點進行整張地圖的洪氾。本演算法突破了能在有障礙物空間下能夠避碰的功能，並且以最快速的時間找出史坦那節點，建立史坦那樹。其相關副程式描述如下副程式

八個方向洪氾：

Step 1 以地圖上最東、南、西、北邊的整排或整列的座標軸當作起始點，引用 HGMR 演算法中的洪氾步驟。

Step 1.1

針對地圖最東邊的行座標[(*I*,0)~(*I*,*J*)]之中的所有節點，將上面的所有節點放入 HGMR 演算法中的 LL_0 進行洪氾。

Step 1.2

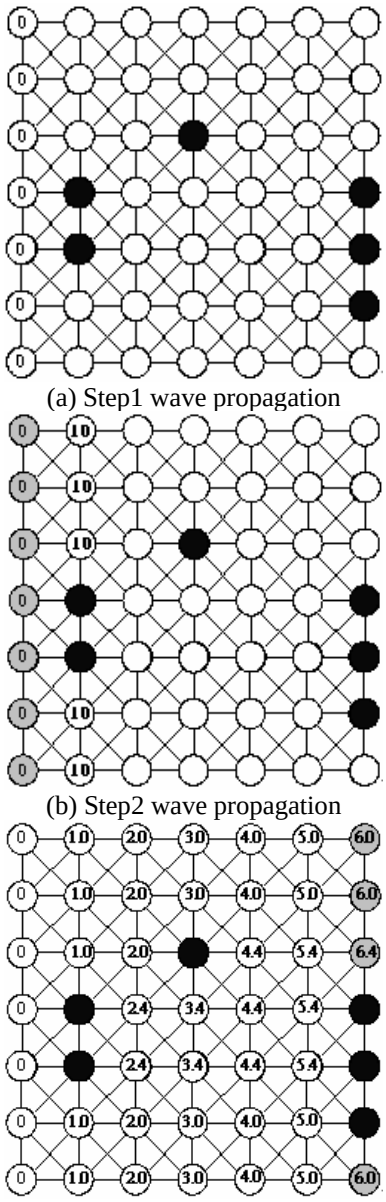
針對地圖最南邊的列座標[(0,*J*)~(*I*,*J*)]之中的所有節點，將上面的所有節點放入 HGMR 演算法中的 LL_0 進行洪氾。

Step 1.3

針對地圖最西邊的行座標[(0,0)~(0,*J*)]之中的所有節點，將上面的所有節點放入 HGMR 演算法中的 LL_0 進行洪氾。

Step 1.4

針對地圖最北邊的列座標[(0,0)~(*I*,0)]之中的所有節點，將上面的所有節點放入 HGMR 演算法中的 LL_0 進行洪氾。

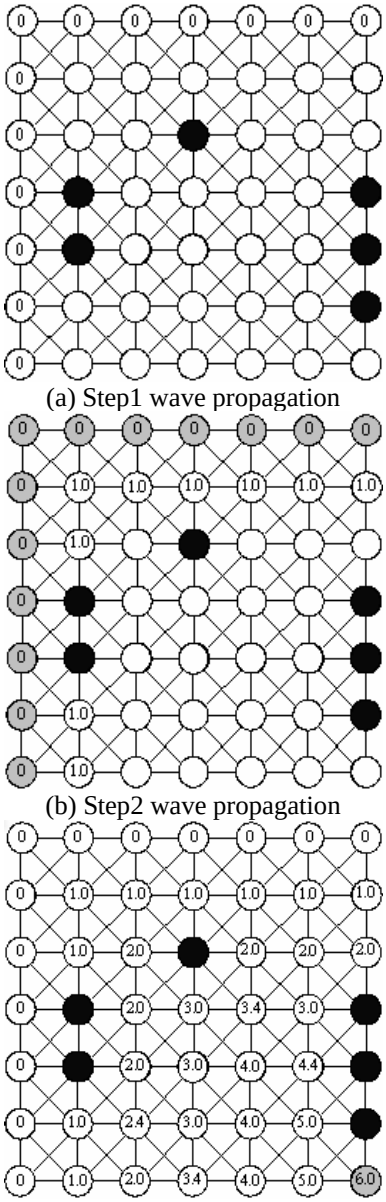


(a) Step1 wave propagation

(b) Step2 wave propagation

(c) wave propagation finish

圖 7 地圖西邊的盡頭進行整排的洪氾範例



(a) Step1 wave propagation

(b) Step2 wave propagation

(c) wave propagation finish

圖 8 地圖西邊與北邊的盡頭進行整排的洪氾範例

Step 2 以地圖上以地圖上最東南、西南、西北、東北邊的整排與整列座標軸當作起始點，引用 HGMR 演算法中的洪氾步驟。

Step 2.1

針對地圖最東邊的行座標[(*I*,*0*)~(*I*,*J*)]與最南邊的列座標[(0,*J*)~ (*I*,*J*)] 之中的所有節點，將上面的所有節點放入 HGMR 演算法中的 LL_0 進行洪氾。

Step 2.2

針對地圖最西邊的行座標[(0,0)~(0,*J*)]與最南邊的列座標[(0,*J*)~ (*I*,*J*)] 之中的所有節點，將上面的所有節點放入 HGMR 演算法中的 LL_0 進行洪氾。

Step 2.3

針對地圖最西邊的行座標[(0,0)~ (0,*J*)]與最北邊的列座標[(0,0)~(*I*,0)] 之中的所有節點，將上面的所有節點放入 HGMR 演算法中的 LL_0 進行洪氾。

Step 2.4

針對地圖最東邊的行座標[(*I*,0)~ (*I*,*J*)]與最北邊的列座標[(0,0)~(*I*,0)] 之中的所有節點，將上面的所有節點放入 HGMR 演算法中的 LL_0 進行洪氾。

END {八個方向洪氾}

(二)有障礙物八向史坦那樹演算法

BEGIN

Step 1 Call 地圖八個方向做 HGMR 演算法洪氾

Step 2 使用米字型延伸求所有終端節點間最短距離與可能的 steiner 節點

For *j*=1 to *J*

For i=1 to I

當 $v_{i,j}$ 屬於 Z 時，先產生一個相對應的 $AD(\eta,\iota)$，然後將此 $v_{i,j}$ 依序存入 LL^Z，在此 $AD(\eta,\iota)$ 中，以 $v_{i,j}$ 為起點執行米字型延伸。

End {For}

End {For}

Step 3 計算出 Z 集合的最小伸展樹長

從 Z 集合中 $AD(l)$中，可以得知每一個 Z_l 點到其他所有 $Z_{l'}$點的距離。所以可以根據 Prim 的最小伸展樹演算法求得最小伸展樹的樹長 $LMST_{initial}$。

Step 4 求出 Improvement Table

For j=1 to J

For i=1 to I

Step 4.1

若 $V_{i,j}^{Steiner}$ 為地圖中重疊節點，則暫時將 $V_{i,j}^{Steiner}$ 當作是 Z 集合中的一點，然後再利用 Prim 的最小伸展樹演算法計算求出加入新節點後的樹長 $LMST_{i,j}$。

Step 4.2

$DLMST_{i,j}$=($LMST_{initial}$ − $LMST_{i,j}$)的值存入相對應的 $IT_{i,j}$ 位置中，然後將 $V_{i,j}^{Steiner}$ 移除原本的 Z 集合。若此值為正，則代表此點 $V_{i,j}^{Steiner}$ 具有使原本的 Z 集合的總樹長降低的意義。

End {For}

End {For}

Step 5 產生 Steiner vertex

搜尋整個 IT 所有的值，取其最大的值，若 ($i_{Steiner}(l)$,$j_{Steiner}(l)$) 為其相對應的位置，即為 Steiner vertex 所在的座標位置，暫將所產生的 Steiner vertex 加入原本的 Z 集合中，使用原本 Z 集合的總樹長減去 Z 集合加入 Steiner vertex 的總樹長，結果大於 0，則修改 Z，然後刪除 $IT_{i,j}$，結果小於 0，直接刪除 $IT_{i,j}$，重複此步驟，遞迴直到 IT 所有的值都計算完為止。

Step6 產生 Steiner Minimal Tree

將 Step 5 的所得到的 Z 集合，執行一次 Prim 最小伸展樹演算法，所得到的結果利用米字型延伸的方法，將整個樹的所有的路徑追蹤完成，整個演算法即完成。

END{有障礙物八向史坦那樹演算法}

(三) 效能分析

1. 空間複雜度分析

關於有障礙物八向史坦那樹演算法的空間複雜度分析，最原始的地圖必須使用一個 $I\times J$ 二維矩陣 $v_{i,j}$。在初始化的過程中，我們使用了 p 個 $I\times J$ 二維矩陣 $v_{i,j}$ 來儲存各個節點所洪氾的狀態，另外還需要一個 $I\times J$ 的二維矩陣儲存 IT 的值。再者輔助計算用的資料結構中，空間中要求最大的 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：

$$\begin{aligned} SC_{total} &= SC_v + SC_{AD} + SC_{IT} + SC_{auxiliary} \\ &= N + pN + N + 2N \\ &= O(pN) \end{aligned}$$

2. 時間複雜度分析

關於有障礙物八向史坦那樹演算法的時間複雜度分析，可分為五個階段來探討，Step 1 為求每個進 Z 節點進行米字型延伸時，能夠避開障礙物，直到地圖八個方向盡頭的等距離圖，其時間複雜度於計算個別的等距離圖時，受限於 HGMR 演算法的洪氾程序，其時間複雜度為 $O(N)$，但地圖有八個方向因此需計算 8 次，因此時間複雜度為 8×$O(N)$。因此 Step 1 的時間複雜度 TC_{Step1}：

$$TC_{Step1} = 8\times N = O(N)$$

Step 2 為求每個米字型延伸的距離矩陣，計算各個 Z 節點的等距離圖，其時間複雜度於計算個別的等距離圖時，其時間複雜度為 $O(p)$，但因為需計算 p 次，因此時間複雜度為 $O(p^2)$。因此 Step 2 的時間複雜度 TC_{Step2}：

$$TC_{Step2} = p\times p = O(p^2)$$

在 Step 3 中計算出 Z 集合中的最小伸展樹，根據 Prim 的 MST 演算法所需要的時間複雜度為 $O(p^2)$，所以 Step 3 的時間複雜度 TC_{Step3}：

$$TC_{Step3} = O(p^2)$$

在 Step 4.1 求出 Improvement Table 的過程中，因為每一次執行 Prim 的最小伸展樹演算法必須花費 $O(p^2)$的時間，而其過程為假設每個進行米字型延伸的第一個重疊節點為 Z 集合中其中一點，所以最差的情形之下，總共必須執行 8×O(p)次，因此 Step 4.1 的時間複雜度 $TC_{Step4.1}$ 為：

$$TC_{Step4.1} = 8p\times(p+1)^2 = O(p^3)$$

在 Step 4.2 中，從 Improvement Table 中搜尋 Steiner vertex，由於每次搜尋必須從 $I\times J$ 的空間中搜尋，因此 Step 4.2 的時間複雜度 $TC_{Step4.2}$ 為：

$$TC_{Step4.2} = 8\times p = O(p)$$

因此 Step4 的總時間複雜度 TC_{Step4} 為：

$$\begin{aligned} TC_{Step4} &= TC_{Step4.1} + TC_{Step4.2} \\ &= O(p^3) + O(p) \\ &= O(p^3) \end{aligned}$$

第 Step 5 中，必須將 Steps 3-4 的所計算得到的

結果，將所產生的 Improvement Table 中的數值當成史坦那測試節點，在最多的情況下會有 8×O(p)個史坦那測試節點，這些節點使用 MST 遞回產生出史坦那節點，因此 Step 5 的時間複雜度 TC_{Step5} 為：

$$TC_{Step5} = (TC_{Step3} + TC_{Step4}) + (8p \times p^2)$$
$$= O(p^2) + O(p^3) + 8(p^3)$$
$$= O(p^3)$$

綜合 Steps 1-5 的時間複雜度可知道本演算法的時間複雜度應該為：

$$TC_{total}$$
$$= TC_{Step1} + TC_{Step2} + TC_{Step3} + TC_{Step4} + TC_{Step5}$$
$$= O(N) + O(p^2) + O(p^2) + O(p^3) + O(p^3)$$
$$= O(N + p^3)$$

(四)演算法執行範例與比較

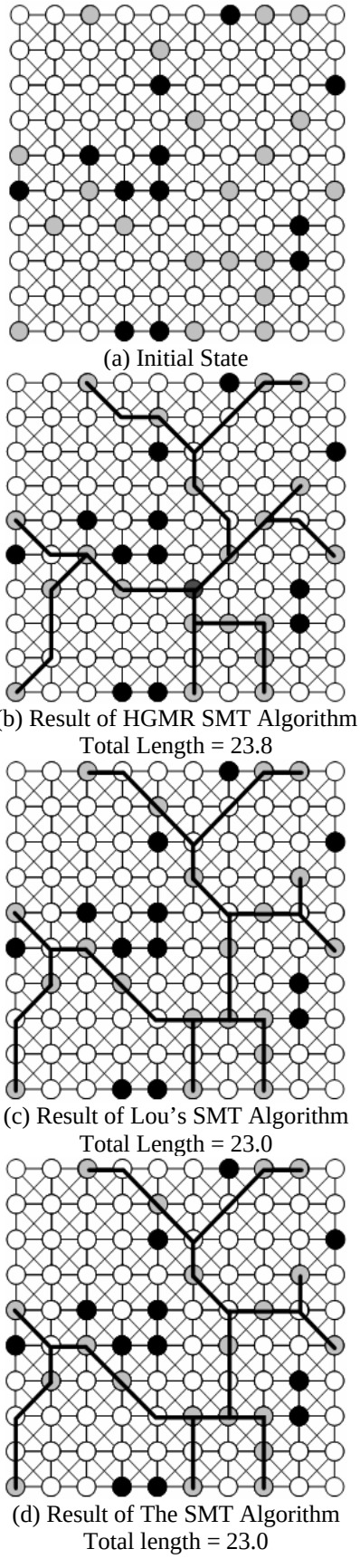
1.演算法範例

這邊所舉的例子比較分析三種演算法的差異性。在圖 9(a)中，黑色的節點代表是不可通行的障礙物，淺灰色節點所代表的是欲連結的終端節點，而白色的部分是可以圖 9(b)-(d)中黑色的通行的自由節點。線段代表根據演算法所計算的結果。從樹長結果來看，我們可以很清楚的發現，在利用 HGMR 史坦那樹演算法所計算出的結果中，所選擇路徑的結果，所產生一些可以很輕易的發現的非最佳解，Lou's SMT Algo. 與本文 SMT Algorithm 優於 HGMR SMT Algorithm，明顯的 Lou's 與本文所提出的史坦那樹演算法，在有障礙物空間下，能夠有效的找出降低樹長的史坦那節點，本文的史坦那樹演算法也充分利用了 X 架構的特性，相較於 Lou's 的演算法來看，能夠更快速的找到所需的史坦那節點，迅速有效的建立出史坦那樹。

我們可以很清楚的在上述的範例中所觀察到，使用 Lou 的演算法所得到的結果可以獲得比較短的路徑長度，不過它所付出的時間以及空間的成本卻相對

表 3 有障礙物演算法分析比較表
p 為 Z 集合總數，N 自由節點總數

Items / Algo.	Time Complexity	Space Complexity	Total Length of SMT
HGMR SMT	$O(p^2N)$	$O(pN)$	Near-Optimal
Lou's SMT	$O(N^2+p^3N)$	$O(N^2)$	Shorter
The SMT With Obstacles	$O(N+p^3)$	$O(pN)$	In-between



(a) Initial State

(b) Result of HGMR SMT Algorithm
Total Length = 23.8

(c) Result of Lou's SMT Algorithm
Total Length = 23.0

(d) Result of The SMT Algorithm
Total length = 23.0

圖 9 Difference between the results of HGMR, Lou's and The SMT Algo.

的比其他兩種演算法來的大，本文所提出的可避碰障礙物史坦那樹演算法，不僅近似於 Lou 的路徑長度，付出的時間最短且空間的成本也優於 Lou 與 HGMR 的 SMT 演算法，其整理結果如前頁表 3 所示。

六、結論與未來展望

本篇文提供在 2D 平面中計算史坦那樹的演算法，本文所提出無障礙物演算法其時間複雜度皆夠控制在 $O(p^3)$之內，有障礙物演算法時間複雜度則控制在 $O(N+p^3)$。本演算法樹長結果都相當近似於 Lou 的演算法。在先前所發現的許多例子中，我們不難發現，有許多的固定樣式(pattern)在本文的結果中並不會發生，換言之，HGMR 演算法其結果卻是跟本文所提演算法與 Lou 演算法的結果是有一段差距的。

日後可將本文所提出的史坦那樹演算法運用到 3D 空間中 10 向式最短路徑演算法，在 PCB 或是 VLSI 實際情況下，每一層之間可以如同本文演算法一樣的行走連接，當同一層之間的連結依照米字型延伸平面 8 向最短路徑演算法來做連結，每一層之間要做連結還多了上、下兩種方向，所以一共是 10 向式的空間試誤型史坦那樹演算法，具備實際應用的價值。

將來所研究的方向是利用"簡單的概念"可以將其無障礙空間下，時間複雜度降為 $O(p^2)$，有障礙物空間下則降低 $O(N+p^2)$為當前目標。可嘗試由重複執行 MST 的條件下，將其時間複雜度降 p 次方。

最後，還可以結合先前的史坦那樹演算法應用到多組線路上面，也就是平面上單層多組史坦那樹連結問題的研究，以期能夠更符合在 EDA 領域中的需求以及研究發展之應用層面。

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] S. Areibi, M. Xie and A. Vannelli, "An efficient rectilinear Steiner tree algorithm for VLSI global routing ,"CCE CE 1067-1072, 2001

[3] W. M. Boyce, "An improved program for the full Steiner tree problem," *ACM Trans. on Math.* Software 3, pp. 194-206, 1977.

[4] W. M. Boyce and J. B. Seery, "STEINER 72: An improved version of the minimal network problem," Tech. Rep., No. 35, Comp. Sci. Res. Ctr. Bell Laboratories, Murray Hill, N.J.

[5] S. K. Chang, "The generation of minimal trees with a Steiner topology," *J. ACM*, vol. 19, pp. 669-711, 1972.

[6] E. J. Cockayne and D. G. Schiller, "Computation of Steiner minimal trees in Welsh and Woddall (Eds.), *Combinatorics*, *Inst. Math. Appl*. pp. 52-71, 1972.

[7] L. R. Foulds and R. L. Graham, "The Steiner problem in phylogeny is NP- complete," *Adv. Appl. Math*., vol. 3, pp. 43-49, 1982.

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

[9] M. Hanan, "On Steiner's problem with rectilinear distance," *J. SIAM Appl. Math*., vol. 14, pp. 255-265, 1966.

[10] J. Hesser, R. Manner, and O. Stucky, "Optimization of Steiner trees using genetic algorithms," *Proc. 3rd Int. Conf. Genetic Algorithms*, pp. 231-236, 1989.

[11] J. M. Ho, C. Vijayan, and C. K. Wong, "New algorithms for the rectilinear Steiner tree problem," *IEEE Trans. Computer-aided Design*, vol. 9, pp. 185-193, 1990.

[12] F. K. Hwang, "On Steiner minimal trees with rectilinear distance," *SIAM J. Appl. Math*., vol. 30, pp. 104-114, 1978.

[13] 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.

[14] G. E. Jan, K. Y. Chang, S. Gao, and I. Parberry, "A 4-geometry maze router and its application on multiterminal nets," *ACM Transactions on Design Automation of Electronic System*, vol. 10, pp. 116-135, Jan., 2005.

[15] B. A. Julstrom, "A genetic algorithm for the rectilinear Steiner problem," *Proc. 5th Int. Conf. Genetic Algorithms*, pp. 474-480, 1993.

[16] 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.

[17] 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.

[18] 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.

[19] J. H. Lee, N. K. Bose, and F. K. Hwang, "Use of Steiner's problem in suboptimal routing in rectilinear metric," *IEEE Trans. Circuit and Systems*, CAS-23, pp. 470-476, 1976.

[20] C. C. Luo, Y. S. Hwang, and G. E. Jan, "Minimal Steiner Trees in X Architecture with Obstacles," to appear in 2005 International Conference on VLSI, pp. 198-203, Las Vegas, Nevada, U. S. A., June 2005.

[21] Z. A. Melzak, “On the problem of Steiner,” *Canad. Math. Bull.*, vol. 4, pp. 143-148, 1961.
[22] R. C. Prim, “Shortest connection networks and some generalizations, *Bell System Tech. J.*, vol. 36, pp. 1389-1401, 1957.
[23] 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.
[24] 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. Optimization*, vol. 4, pp.179-192, 1980.
[25] 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.
[26] P. Winter, “An algorithm for the Steiner problem in the Euclidean plane,” Networks, vol. 15, pp. 323-345, 1985.
[27] P. Winter, “Steiner problem in networks: a survey,”Net works, vol. 17, pp. 129-167, 1987.
[28] J. Wu, “A fault-tolerant and deadlock- free routing protocol in 2D meshes based on odd-even turn model,” IEEE Trans. on Computers, vol. 52, Issue: 9, Pages 1154-1169, Sept. 2003.