試誤型史坦那樹演算法及電子設計自動化應用
PART II：3D 空間中試誤型可容錯十向史坦那樹演算法
Obstacle-Avoiding Heuristics for Steiner Tree Problem in EDA
PART II：Heuristics for Steiner Tree in 3D 10-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

本文延伸前一篇 Part I 2D 平面中試誤型可容錯八向史坦那樹演算法，將其擴展為 X 架構上 3D 空間之試誤型史坦那樹演算法演算法，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。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 $O(pN)$，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

The heuristics Steiner minimal tree algorithms on X architectures presented in part I can be extended to 3-dimensional 10-directional VLSI layers by adding extra parameters in the 2-dimensional data structures. The 3D SMT algorithms have the same time and space complexities as the 2D algorithms obviously.

Keywords: X architectures, Steiner minimal tree, 3D SMT algorithm.

一、緒論

現今在大型積體電路中(VLSI)或是印刷電路板(PCB)、網狀架構的容錯路徑上面所需要的需求已經朝向多層(multiple layers)的設計 [9]。所以我們將原本在 2D 平面上的史坦那樹演算法利用 Jan 三維空間與 Luo 三維空間中的最短路徑演算法 [1]，[5]，本文將其拓展成 3 維空間中 10 向史坦那樹演算法，符合未來可能應用到的需求。目前於朝 3D 空間發展 VLSI 與 PCB 應用多層(multiple layers)設計也已經有學者朝此方向研究與應用 [7], [8]。

Lou 於近年提出一個不同於先前 Areibi 的觀念 [2]，[5]，並且將其延伸到 3D 空間中應用，使用 Areibi 演算法的方法尋找所有的 Hanan grid [3]，再嘗試將其中每一個 Hanan vertices(HV)當作是整個 Steiner trees(ST)節點的潛在可能的成員之一，使用 Prim 的觀念去計算新節點集合的 Minimal spanning tree(MST)樹長 [6]，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 Steiner vertices(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 的 3 維空間演算法是在 X 架構上並且可存在障礙物的。

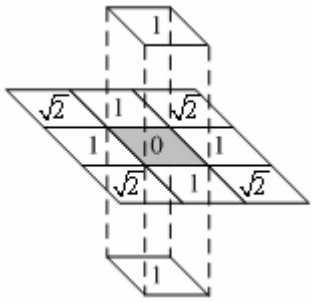
本演算法利用 HGMR 演算法可以向 3D 空間擴充的概念 [4]，將原本的 8 向 2D 平面最短路徑演算法，延伸到 10 向 3D 空間最短路徑演算法。將其應用到米字型延伸改良至 3D 空間觀念，如圖 1(a)、(b)所示，其十個不同的方向以及其所代表不同的距離，距起點的距離從 1、$\sqrt{2}$ 到上下兩向距離 1 為最基礎的單位。

本演算法使用 Hanan 觀念將其更改為空間十向米字型延伸，應用於空間下地圖無障礙情形下。此外，本演算法於有障礙物時，應用了 HGMR 演算法中的洪氾偵測障礙物觀念，能夠於空間中有效的避開障礙物，具備實際應用的價值。

二、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。

(a) 空間中 10 向示意圖形



(b) 空間中 10 向距離示意圖

圖 1 空間 10 向示意圖形

三、無障礙物史坦那樹演算法

(一)無障礙物十向史坦那樹演算法介紹

本演算法利用 HGMR 演算法可以向 3D 空間擴充的概念，將原本的 8 向 2D 平面最短路徑演算法，延伸到 10 向 3D 空間最短路徑演算法。本演算法使用 Hanan 觀念將其更改為空間十向米字型延伸，應用於空間下地圖無障礙情形下，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，並且利用 Prim 演算法的觀念求出各 Z 節點之間的最短距離，針對兩點間最先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 Improvement Table，將所產生的 *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：

$(V^{SM}_{(imin,jmin,kmin)}, V^{SM}_{(imax,jmin,kmin)},$
$V^{SM}_{(imin,jmax,kmin)}, V^{SM}_{(imax,jmax,kmin)})$
$(V^{SM}_{(imin,jmin,kmax)}, V^{SM}_{(imax,jmin,kmax)},$
$V^{SM}_{(imin,jmax,kmax)}, V^{SM}_{(imax,jmax,kmax)})$

此八點所形成的 3D 矩陣之內是我們座標軸所設定的範圍。

Step 1.2 For ($k=V^{SM}_{kmin}$;$k<=V^{SM}_{kmax}$;k++)
For ($j=V^{SM}_{jmin}$; $j<=V^{SM}_{jmax}$;j++)
For ($i=V^{SM}_{imin}$;$i<=V^{SM}_{imax}$;i++)
{
$V^{SM}_{i,j,k}$ = True
}
End {For}
End {For}
End {For}

/* 假設 $V^{SM}_{i,j,k}$ 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 $k=V^{SM}_{kmin}$ to V^{SM}_{kmax}
For $j=V^{SM}_{jmin}$ to V^{SM}_{jmax}
For $i=V^{SM}_{imin}$ to V^{SM}_{imax}
當 $v_{i,j,k}$ 屬於 Z 時，先產生一個相對應的 $AD(\eta,\iota,\kappa)$，然後將此 $v_{i,j,k}$ 依序存入 LL^Z，在此 $AD(\eta,\iota,\kappa)$ 中，以 $v_{i,j,k}$ 為起點執行米字型延伸。
End {For}
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 $k=V^{SM}_{kmin}$ to V^{SM}_{kmax}
For $j=V^{SM}_{jmin}$ to V^{SM}_{jmax}
For $i=V^{SM}_{imin}$ to V^{SM}_{imax}

Step 4.1

若 $V^{Steiner}_{i,j,k}$ 為指定範圍內重疊節點，則暫時將 $V^{Steiner}_{i,j,k}$ 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 $LMST_{i,j,k}$。

Step 4.2

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

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 $\|V\|$ 個節點。。
E	邊集合(set of edges)，其中包含有 $\|E\|$ 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。
$v_{i,j,k}$	vertices，儲存 $I \times J \times K$ 網狀網路內的節點狀態，其中 $(i, j\ i, k)$ 為三維陣列之索引值，$0 \le i \le I$，$0 \le j \le J$，$0 \le k \le K$。
$v_{i,j,k}^{Steiner}$ $v_{i,j,k}^{Steiner}(l)$	Steiner vertices，經由米字型延伸所得之交叉點，儲存 $I \times J \times K$ 網狀網路內的節點狀態，其中 (i, j, k) 為三維陣列之索引值，當以 $v_{i,j,k}^{Steiner}(l)$ 表示時，表其為存取 $v^{Steiner}(l)$ 上索引值 (i, j, k) 位置上之值，$0 \le i \le I$，$0 \le j \le J$，$0 \le k \le K$。
Atb $Atb_{i,j,k}$	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, k) 為三維陣列之索引值，$0 \le i \le I$，$0 \le j \le J$，$0 \le k \le K$。
	vertices Subset Mesh，描述終端點延伸範圍內的屬性，預設為 False
$v_{i,j,k}^{SM}$	，其值設為 True,則表示在指定範圍之內，其中 (i, j, k) 為三維陣列之索引值，$0 \le i_{Min}^{SM} \le i \le i_{Max}^{SM} \le I$，$0 \le j_{Min}^{SM} \le j \le j_{Max}^{SM} \le J$ $0 \le k_{Min}^{SM} \le k \le k_{Max}^{SM} \le K$
$AD_{i,j,k}$ $AD_{i,j,k}(l)$	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 $I \times J \times K$。當以 $AD_{i,j,k}(l)$ 表示時，表其為存取 $AD(l)$ 上索引值 $(i,\ j,\ k)$ 位置上之值，$0 \le i \le I$，$0 \le j \le J$，$0 \le k \le K$。
$AD(\eta,\iota,\kappa)$ $AD_{i,j,k}(\eta,\iota,\kappa)$	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延伸後所得之等距離圖，$0 \le l < p$。$AD(\eta,\iota,\kappa)$ 陣列之大小為 $I \times J \times K$。當以 $AD_{i,j,k}(\eta,\iota,\kappa)$ 表示時，表其為存取 $AD(\eta,\iota,\kappa)$ 上索引值 (i, j, k) 位置上之值，$0 \le i \le I$，$0 \le j \le J$，$0 \le k \le K$。
LL^Z $LL^Z(l)$	Linked List of Z-vertices，儲存記錄 $v_{i,j,k}$ 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 $v_{i,j,k}$。
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 $e_{v_{i,j,k},v_{i',j',k'}}$ 所組成。當以 $LL^{RST}(l)$ 表示時，係指取出 LL^{RST} 中第 l 個邊 $e_{v_{i,j,k},v_{i',j',k'}}$。
$v_{(imin,jmin,kmin)}^{SM}$ $v_{(imax,jmin,kmin)}^{SM}$ $v_{(imin,jmax,kmin)}^{SM}$ $v_{(imax,jmax,kmin)}^{SM}$ $v_{(imin,jmin,kmax)}^{SM}$ $v_{(imax,jmin,kmax)}^{SM}$ $v_{(imin,jmax,kmax)}^{SM}$ $v_{(imax,jmax,kmax)}^{SM}$	為設定終端點延伸範圍的座標位置
$(i_{Steiner}(l),$ $j_{Steiner}(l),$ $k_{Steiner}(l))$	$Steiner$ 節點的三維陣列之索引值，當以 $(i_{Steiner}(l), j_{Steiner}(l), k_{Steiner}(l))$ 表

	示時，係指取出第(l)個($i_{Steiner}$,$j_{Steiner}$ $k_{Steiner}$)，$0<i_{steiner}<I$，$0<j_{steiner}<J$，$0<k_{steiner}<K$。
$*v^{temp}$	儲存計算時所用的暫存節點 $v_{i,j,k}$ 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，$0<index<2$。
$Atb'_{i,j,k}$	vertices，係為輔助計算用時的暫存資料結構，其內儲存 $I\times J\times K$ 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, k)為二維陣列之索引值，$0\le i\le I$，$0\le j\le J$，$0\le k\le K$。
$Num^{obstacle}$	Number of obstacles，儲存 $v_{i,j,k}$ 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。
$LMST_{i,j,k}$	temporal Length of Minimal Spanning Tree，將點(i,j,k))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。
$DLMST_{i,j,k}$	Difference of Lengths of Minimal Spanning Trees，其值為 $LMST_{initial}-LMST_{i,j,k}$。。
IT $IT_{i,j,k}$	Improvement Table，為儲存 $DLMST_{i,j,k}$ 值所用，當以 $IT_{i,j,k}$ 表示時，係表存取 IT 上索引值為(i, j, k)位置上之值。其中(i, j, k)為三維陣列之索引值，$0\le i\le I$，$0\le j\le J$，$0\le k\le K$。

End {For}
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($i_{Steiner}(l)$，$j_{Steiner}(l)$,$k_{Steiner}(l)$)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 $IT_{i,j,k}$，結果小於 0，直接刪除 $IT_{i,j,k}$，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{無障礙物十向史坦那樹演算法 }

(三) 效能分析

1.空間複雜度分析

關於無障礙物十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 $I\times J\times K$ 三維矩陣 $v_{i,j,k}$。在初始化的過程中，我們使用了 p 個 $I\times J\times K$ 三維矩陣 $v_{i,j,k}$ 來儲存各個節點所洪氾的狀態，另外還需要一個 $I\times J\times K$ 的三維矩陣儲存 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

關於無障礙物十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 為設定座標軸求每個米字型延伸($v^{SM}_{(imin,jmin,kmin)}$, $v^{SM}_{(imax,jmin,kmin)}$, $v^{SM}_{(imin,jmax,kmin)}$, $v^{SM}_{(imax,jmax,kmin)}$)($v^{SM}_{(imin,jmin,kmax)}$, $v^{SM}_{(imax,jmin,kmax)}$, $v^{SM}_{(imin,jmax,kmax)}$, $v^{SM}_{(imax,jmax,kmax)}$)

伸的範圍此八點所形成的 3D 矩陣之內是我們座標軸所設定的範圍，由於八個座標僅作八次設定，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 10×O(p)次，因此 Step 4.1 的時間複雜度 $TC_{Step4.1}$ 為：

$$TC_{Step4.1} = 10p \times (p+1)^2 = O(p^3)$$

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

$$TC_{Step4.2} = 10 \times p = O(p)$$

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 10(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：

$$\begin{aligned} TC_{Step5} &= (TC_{Step3} + TC_{Step4}) + (10p \times p^2) \\ &= O(p^2) + O(p^3) + O(p^3) \\ &= O(p^3) \end{aligned}$$

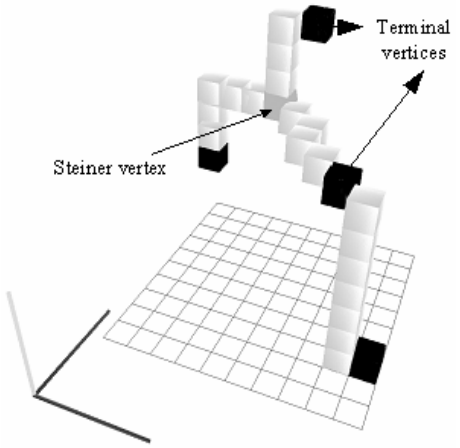
綜合 Steps 1-5 的時間複雜度可知道本演算法的時間複雜度應該為：

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

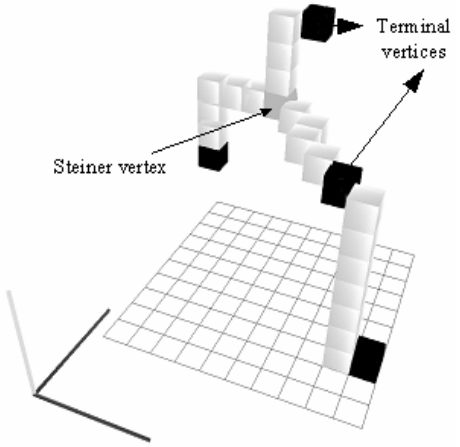
(四)演算法執行範例與比較

1.演算法範例

如圖 2 中所示，在 10×10×10 的空間中，深灰色方塊代表欲連結的終端節點，淺 p 為 Z 集合總數，N 自由節點總數灰色在圖 2 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較佳的路徑長



(a) 3D Lou SMT (4 terminals)無障礙物演算法的結果



(b) 本文 3D SMT (4 terminals)無障礙物演算法的結果

圖 2 執行 3D 空間中有障礙物史坦那樹演算法的結果

度，不過它所付出的時間及空間成本卻相對的比本文演算法來的大，本文所提出可避碰障礙物的史坦那樹演算法，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 的 SMT 演算法，如表 2 所示。

表 2 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 Without Obstacles	$O(p^3)$	$O(pN)$	In-between

四、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

在 3D 空間中的最短路徑演算法加上障礙物，拓展成 3D 空間中 10 向有障礙物史坦那樹演算法，我們都知道目前大型積體電路中(VLSI)或是印刷電路板(PCB)上多層設計幾乎都是在有障礙物(IC、電容、電阻..等等)的情況下，而拉線成垂直角橫跨障礙物，又必須花費高額的成本，因此本演算法因應現實社會的需求，在空間下能有效的避碰障礙物，並且找到最佳路徑。

根據HGMR最短路徑演算法可以向3D空間有障礙物延伸的概念，可以將原本的 2D 有障礙物八向式可容錯史坦那樹演算法，延伸成 3D 十向有障礙物可容錯史坦那樹演算法。而其演算法的基本架構以及精神皆不變，所以在 3D 有障礙物上面演算法所執行的結果和 2D 有障礙物上面所執行的結果並不會有太大的差異性。而在 Part I 2D 平面中試誤型可容錯八向史坦那樹演算法所提到的 2D 平面上史坦那樹演算法所執行的結果幾乎趨近於 Lou，由於 3D 史坦那樹演算法的精神不變，所以理論上在 3D 空間中所執行史坦那樹演算法的結果必定也是趨近於 Lou。
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,*0*)~(*I*,*J*,*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 1.2
針對地圖最南邊的列座標[(0,*J*,*0*)~(*I*,*J*,*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 1.3
針對地圖最西邊的行座標[(0,0,*0*)~(0,*J*,*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 1.4
針對地圖最北邊的列座標[(0,0,*0*)~(*I*,*0*,*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 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,*0*)~(*I*,*J*,*0*)]與最南邊的列座標[(0,*J*,*0*)~ (*I*,*J*,*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 2.2
針對地圖最西邊的行座標[(0,0,*0*)~(0,*J*,*0*)]與最南邊的列座標[(0,*J*,*0*)~ (*I*,*J*,*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 2.3
針對地圖最西邊的行座標[(0,0,*0*)~(0,*J*,*0*)]與最北邊的列座標[(0,0,*0*)~(*I*,0,*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,*0*)~(*I*,*J*,*0*)]與最北邊的列座標[(0,0,*0*)~(*I*,0,*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)有障礙物十向史坦那樹演算法

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 *k*=1 to *K*
For *j*=1 to *J*
For *i*=1 to *I*
當 $v_{i,j,k}$ 屬於 Z 時，先產生一個相對應的 $AD(\eta,\iota,\kappa)$，然後將此 $v_{i,j,k}$ 依序存入 LL^Z，在此 $AD(\eta,\iota,\kappa)$ 中，以 $v_{i,j,k}$ 為起點執行米字型延伸。
End {For}
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 *k*=1 to *K*
For *j*=1 to *J*
For *i*=1 to *I*
Step 4.1
若 $V_{i,j,k}^{Steiner}$ 為地圖中重疊節點，則暫時將 $V_{i,j,k}^{Steiner}$ 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 $LMST_{i,j,k}$。
Step 4.2
將 $DLMST_{i,j,k}=(LMST_{initial}-LMST_{i,j,k})$的值存

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

End {For}

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 $(i_{Steiner}(l), j_{Steiner}(l), k_{Steiner}(l))$ 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 $IT_{i,j,k}$，結果小於 0，直接刪除 $IT_{i,j,k}$，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{有障礙物十向史坦那樹演算法}

(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 $I \times J \times K$ 三維矩陣 $v_{i,j,k}$。在初始化的過程中，我們使用了 p 個 $I \times J \times K$ 三維矩陣 $v_{i,j,k}$ 來儲存各個節點所洪氾的狀態，另外還需要一個 $I \times J \times K$ 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 complex- ity) 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 $O(N)$，但地圖有十個方向因此需計算 10 次，因此時間複雜度為 $10 \times O(N)$。因此 Step 1 的時間複雜度 TC_{Step1}：

$$TC_{Step1} = 10 \times N = O(N)$$

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 $O((p)\ p)$。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 $10 \times O(p)$次，因此 Step 4.1 的時間複雜度 $TC_{Step4.1}$ 為：

$$TC_{Step4.1} = 10p \times (p+1)^2 = O(p^3)$$

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：

$$TC_{Step4.2} = 10 \times p = O(p)$$

因此 Step4 的總時間複雜度 TC_{Step4} 為：

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

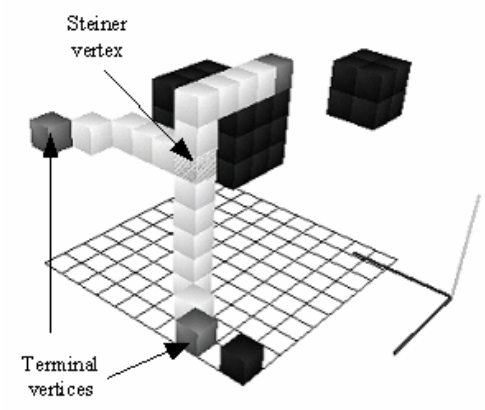
第 Step 5 中，必須將 Steps 3-4 的所計算得到的結果，將所產生的 Improvement Table 中的數值當成史坦那測試節點，在最多的情況下會有 $10(p)$個史坦那測試節點，這些節點使用 MST 遞回產生出史坦那節點，因此 Step 5 的時間複雜度 TC_{Step5} 為：

$$\begin{aligned} TC_{Step5} &= (TC_{Step3} + TC_{Step4}) + (10p \times p^2) \\ &= O(p^2) + O(p^3) + (10 \times p^3) \\ &= O(p^3) \end{aligned}$$

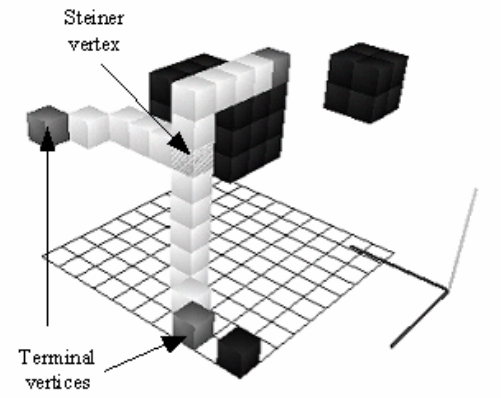
綜合 Steps 1-5 的時間複雜度可知道本演算法的時間複雜度應該為：

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

(四)演算法執行範例與比較



(a) 3D Lou SMT (3 terminals)
有障礙物演算法的結果



(b) 本文 3D SMT (3 terminals)
有障礙物演算法的結果

圖 3 執行 3D 空間中有障礙物史坦那樹演算法的結果

1.演算法範例

如圖 3 中所示，在 10×10×10 的空間中，黑色的部份是不可通行之障礙節點，深灰色方塊代表欲連結的終端節點，淺灰色方塊代表演算法所產生的路徑。我們可以在圖 3 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較佳的路徑長度，不過它所付出的時間以及空間的成本卻相對的比本文演算法來的大，本文所提出可避碰障礙物的史坦那樹演算法，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 的 SMT 演算法，如表 3 所示。

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

五、結論與未來展望

本文提供在 3D 空間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能夠控制在 $O(p^3)$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 26 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空間 10 向最短路徑演算法來做連結，延伸到 26 向 3D 空間最短路徑演算法。其二十六個不同的方向以及其所代表不同的距離，所以一共是 26 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] 羅仲志，"試誤型史坦那樹及單層多組線路連結演算法"，國立台灣海洋大學資訊工程學系碩士論文，中華民國九十三年七月.

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

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

[4] 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.

[5] 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.

[6] R. C. Prim, "Shortest connection networks and some generalizations, *Bell System Tech. J.*, vol.

36, pp. 1389-1401, 1957.
[7] W. D. Smith and J. M. Smith, “The Steiner Ratio in 3D Space”, Journ. Comb. Theory A69 (1995), pp.301-332.
[8] T. Y. Wang, Y. M. Lee, and C. P. Chen, "3D Thermal-ADI: An Efficient Chip-Level Transient Thermal Simulator," International Symposium on Physical Design (ISPD), 2003.
[9] 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.