多層印刷板電路繞線演算法及其應用

An Efficient Router for Multi-Layer PCBs

詹景裕
國立臺北大學
電機工程研究所
gejan@mail.ntpu.edu.tw

李明哲
銘傳大學
資訊傳播工程研究所
leemc@mcu.edu.tw

郭芳誠
國立臺北大學
通訊工程研究所
s79469203@webmail.ntpu.edu.tw

劉萬榮
國立臺北大學
電機工程研究所
wrliou@mail.ntpu.edu.tw

摘要—印刷電路板在我們的日常生活中，佔有了很重要的地位，應用幾乎無所不在。為了降低其製作的成本及難易，在過去電子設計自動化的研究領域中，一直以減少層數與金屬線總長度為目標。本文的多層繞線演算法，從單組的 High Geometry Maze Router 架構發展而來，再利用了過去我們的所研究的單層多對連結演算法以及結合了新提出的單層 PCB 連續節點延伸技術，而衍生出了多層印刷電路板繞演算法。在 $X \times Y \times L$ 的三維空間中，提出了可以將 P 組欲相連的節點，以較少的層數與較短的金屬線總長度完成繞線，其時間複雜度與空間複雜度分別為 $O(PN)$及 $O(P^2N)$，其中 $N = X \times Y$ 為單層 PCB 的節點總數，L 為總層數。

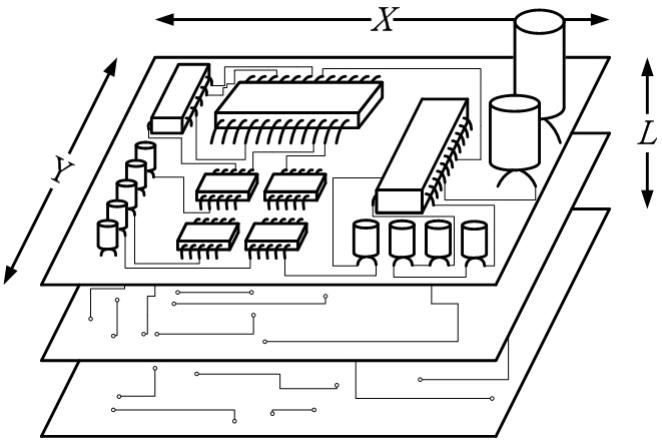
關鍵詞—單層 PCB 連續節點延伸、High Geometry Maze Router, PCB Routing Problem, Single-Layer Two Terminal Nets Algorithm

一、緒論

印刷電路版(PCB)技術在現代人的生活中，往往佔有著不可缺少的地位，不論是先進的 3C 產品或是一般的生活家電，內部的各個電子元件都必定有著 PCB 的連接。依照 PCB 的製作方式，主要分為三大類：單面板、雙面板、多層板。其製成成本的高低，便是以 PCB 的層數及使用的金屬線為主要的考量。如何將 PC 總層數降低，以及將所使用的金屬線總長度減少一直是各個研究想要追求的目標 [1]。最早期的 PCB Layout 過程中，以人工來配置電子元件及手繪轉印的方式來完成電路繞線。但隨著時代的進步，電子產品應用日驅精密，完全人工的方式已不能繪製出如此多層壓縮且複雜的電路，市面上因此出現了需多的電腦輔助設計的軟體，例如 Protel、Power PCB、Allegro PCB Design 等，都是以半人工的方式，利用的軟體協助配合人工拉線方式輔以來完成。但其使用上，在配置之前必需先決定好所要的 PCB 的層數才能進行繞線，但是層數多寡的決定，往往需要長期的人力培訓及豐富的實作經驗才能做出較適當的決策。

在過往的研究中，自動化的元件配置已被提出各種不同的方式[2, 3]，亦有平衡各層之間金屬線多寡的方法被提出[4]。而其中 Murata 的方法在配置完成後，仍是以 Protel Advance PCB 2.8 來輔助完成繞線，但是其仍是以人工方式先決定以四層板的配置來進行輔助繞線。所以本文以提出一個當各個電子元件的配置已決定後，在繞線時以電腦計算方式，迅速且自動化的決定 PCB 所需層數，並在各層之間取得金屬線總長度較短且平衡的演算法。

第二節的部份會先描述本多層印刷電路板問題的定義，再列出本演算所需要使用變數的定義。第三節會解釋本演算法的主要架構，並



圖一 多層印刷電路版示意圖

講解所使用到的四個副程式，前兩個為本研究室之前的發表的單層多對連結副程式 [5]以及 Higher Geometry Maze Router (HGMR) [6]，後面兩個其中一個是為了解決印刷電路板節點對連續之特性，而進行的前置處理，另一個為進行多層繞線時，降低層數與金屬線總長度的連結方法。第四節將針對時間複雜度以及空間複雜度兩方面，對於本演算法效能的效能進行分析。最後第五節有實驗的結果範例以及第六節將進行總結並提出未來研究方向。

二、多層印刷電路板繞線問題與變數定義

假設在 $X \times Y \times L$ 的多層自由平面空間中，其中 X 表單層電路板的長度，Y 表單層電路板的寬度，而 L 代表其總共的層數，如圖一所示。共有 P 組節點要連結，每一組只有兩個節點欲彼此相連。在每一組節點要各自相連時，連結的路徑必需在同一層之中，其他各組的節點不論是否位在同一層皆視為障礙物節點，而且每一組的路徑在同一層時不可以相互交叉或者重疊。

在處理這多層印刷電路版繞線問題時，由於需要利用到過去我們所提出的單層多對節點連結演法[5]，表一將整合介紹所需使用到的變數與其說明。

表一 變數表

變數	型態	說明
AP	二維矩陣	Array of Pairs，為一個輔助計算用的資料結構，為一個 $P \times 9$ 的矩陣，內容儲存了各組路線的 k、Lay、$Path(k)$、$OL(k)$、$T(k, 0)$、$T^{Ext}(k, 0)$、$T(k, 1)$、$T^{Ext}(k, 1)$、$Int(k, p)$
$Continuous_i$	集合	$Continuous_i = \{C_1, C_2, \ldots, C_m, \ldots, C_n\}$表示第 i 列上連續 3 個以上的節點列，其中 $C_m = \{\ldots T_{i,j-1}, T_{i,j}, T_{i,j+1} \ldots\}$
$Int(k, p)$	布林陣列	Intersection，其長度，用來記錄第 k 組其原始最短路線與第 p 組原始最短路線是否有相交，有則記為 true，無則記為 false
l	整數	索引值，表目前正在處理的第 l 層，$1 \leqq l \leqq L$
L	整數	所需電路板的總共層數，預設為 1
$Lay(k)$	整數	Layer，第 k 組線路所在的層數
$Len(k)$	整數	Length，第 k 組線路的路徑長度
N	整數	單層電路板的節點總數，$N = X \times Y$
$Obs_{i,j}$	二維布林矩陣	於 HGMR 時，各個節點的資訊，若為障礙節點則設為 true 若為自由節點則設為 false。
$OL(k)$	浮點數	Original Length，第 k 組線路在未被他組路線阻擋下，可直接連結的最短路徑長度
P	整數	欲連結的總對數
$Path(k)$	連結串列	第 k 組線路在 $Lay(k)$層所連結的路線

T^D	陣列	Destination，用於 HGMR 時紀錄終點座標(i, j)
T^S	陣列	Start，用於 HGMR 時紀錄起點座標(i, j)
$T_{i,j}(k, h)$	陣列	Terminal，記錄第 k 組對的第 h 點座標(i, j)
$T^{Ext}_{i',j'}(k,h)$	陣列	Terminal Prime，記錄第 k 組的第 h 點對移動過後座標(i', j')
$V_{i,j,l}$	三維整數矩陣	Vertices，$X \times Y$ 儲存電路佈局圖內節點在第 l 層目前的狀態，其值介於{0, 1, 2,…,P }之間，若為障礙節點則設為。其中值為 0 時表示該節點為自由節點，其中值不為 0 時表示該節點為欲連結的第 k 組對其中之一點。其中(i, j)為二維矩陣之索引值，$0 \leqq i \leqq X$-1, $0 \leqq j \leqq Y$-1
X	整數	單層電路板的長度
Y	整數	單層電路板的寬度

三、演算法

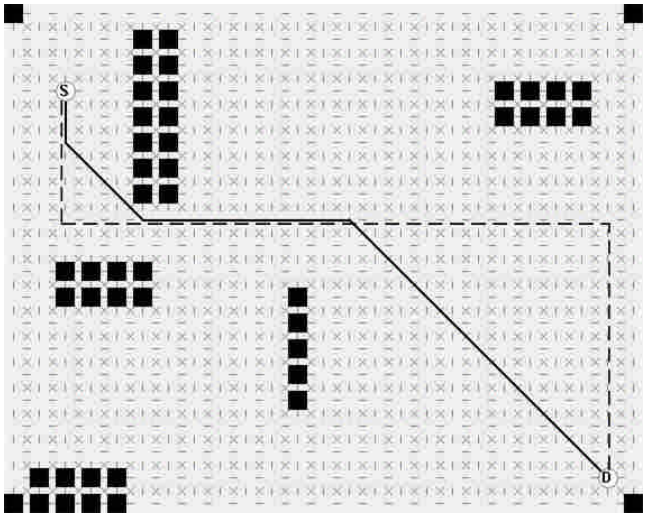
本演算法的主要架構是先記錄下各個組對以及障礙空間的必要資訊後，再利用四個副程式的運算結果加以整合評估，計算出可能所需的最少層數。最後配合著回饋調整的機制，進行多次的加權運算，將多餘的層數及金屬線長度調至最少。

演算法：多層多組對連結

BEGIN

STEP 1：初始化

將各組的起始點與終點、障礙物位置等參數輸入。而在各組尚未連結時，先計算各組的最小連結長度 $OL(k)$，且更新 $Int(k, p)$，預設各組的路徑所在之層數 $Lay(k) = 1$，$0 \leqq k \leqq P$。



圖二 八向與四向路經規劃比較。

STEP 2：第一次初步估計層數， $l = 1$

STEP 2.1：

Call **單層 PCB 連續節點延伸**副程式

Call **單層多組對連結**副程式

並更新有連結成功的各組 AP 資訊

STEP 2.2：

對於未能求得路徑的 k 組對，其 $Lay(k) = Lay(k) + 1$

STEP 2.3：若有任一組未能求得連結路線，則 $L = L + 1$，重覆 STEP 2.1 直到所有組對都先有一組解

STEP 3：縮短總長度與降低層數

STEP 3.1：以決定總層數 L 來放置

Call **多層 PCB 繞線**副程式

STEP 3.2：嘗試降低層數

$L = L-1$ 回 STEP 3.1 看是否能找到解，若有解則以新的 L 來決定最佳解法，若無則結束，$L = L + 1$

END{多層多組對連結}

多層印刷電路在做繞線時，需要使用四個的副程式加以輔助才得以計算，在我們研究室過去的研究中，已提出了單組的八向最短路徑規劃[6]，以及單層多對的路徑連結[5]。

副程式 1：HGMR (T^S, T^D, $Obs_{i,j}$)

單組的繞線部份，本文使用建立在八向連接網格的最短路徑演算法 Higher Geometry Maze Router，能夠有效的降低連結的金屬線長，並有效於時間複雜度 $O(N)$與空間複雜度亦 $O(N)$下，得以計算完成，其中 $N = X \times Y$。如圖二從起始點 S 到終結點 D，利用八向繞線的方式相較於四向繞線的方式更可以將佔據的面積降的更低，在實務應用上有正面的幫助。

在尚未確認所需要層數的多寡之前，單層多對連結 [5]是很重要的。利用單層多對連結之結果，來連結最多的組對，再將剩下無法連結的組對移至另一層，乃是本多層繞線演算法的估計最多所需層數的依據。

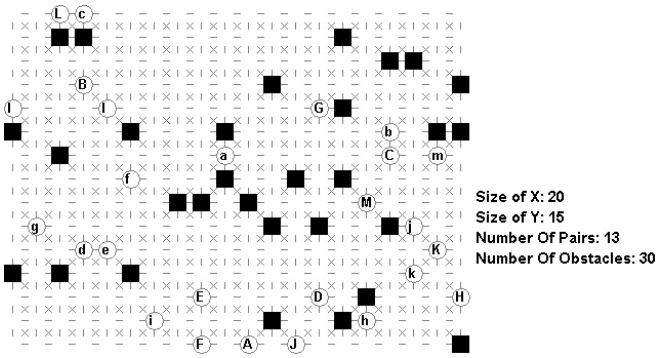
副程式 2：Single-layer TTN Routing(T_k^S, T_k^D, V)

在總節點數為 N 的二維的平面中，有 P 組的線路必需要相互連結，且各組與組之間的連結路徑不可以有交叉或重疊。在給定各組的起點 T_k^S 與終點 T_k^D 後，$1 < k < P$，並有效於時間複雜度 $O(N)$與空間複雜度亦 $O(PN)$下，得以計算完成。如圖三(a)中有 13 組節點對彼此欲相連，最後找到一組繞線結果，如圖三(b)金屬線總長度為 125.4 個單位長。

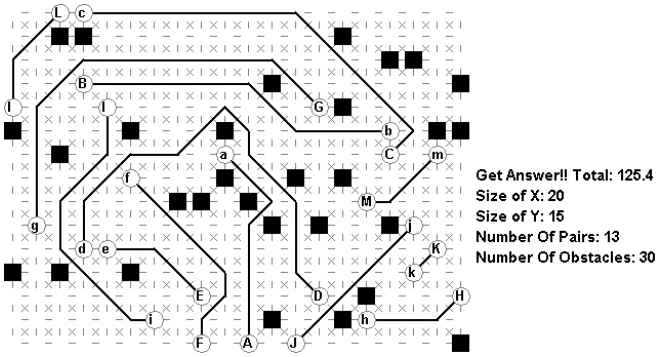
但是在處理印刷電路版的問題時，由於印刷電路版元件其節點為一整排的 Pin 腳，有連續且依附障礙物的特性。因此在直接使用單層多組對連結時，必需先配合著一個各組連續節點延伸的前置處理，才能達到單層連結對數最多之目的。

副程式 3：Terminals Extension for Single-layer PCB(V)

BEGIN



(a)起始狀態圖



(b)連結成功結果圖

圖三 單層多對連結副程式範例

STEP 1：初始化

將要處理的起始點與終點、障礙物位置等參數輸入。

STEP 2：針對 column 方向進行掃瞄

STEP 2.1：由 $i = 1$ 到 X

掃描出 $V_{i,j,l}$ 為 Free-Terminal-Obstacle 的情況則紀錄 $V_{i,j,l}$

STEP 2.2：由 $j = 1$ 到 Y

判斷是否有 3 個以上連續掃描出的節點被紀錄，若有則將其 insert 於 $Continuous_i$

STEP 2.3：

For $i = 1$ to $|Continuous_i|$

For $index = 1$ to $|C_m|$

記錄 C 的延伸區域，其延伸區域的

Length = $|C_m|$，Width = $\left\lfloor \frac{|C_m|-1}{2} \right\rfloor$

End for

STEP 3：將平面旋轉 90°，回 STEP 2 重覆執行直到上下左右四個方向的延伸區域均已記錄完成

STEP 4：檢查 STEPS 1~2 所記錄的延伸區域是否有交集

if 無交集

Call **山型延伸**副程式 (附錄 A)

else

Call **L 型延伸**副程式(附錄 B)

END{單層 PCB 連續節點延伸}

副程式 4：Multi-lay PCB Routing(*L, AP*)

BEGIN

STEP 1：初始化，Call **HGMR** 副程式，計算各組 *Int*(*k*)與 *Len*(*k*)，並將各組的 *Lay*(*k*)設為 0

STEP 2：排序 AP

利用 radix sort 的觀念，依序把 *Int*(*k*)、*Len*(*k*)當作十位、個位作排序，其值越小排在前面。求出其先後順序寫入 *Rank*(*k*)值。

STEP 3：依 *Rank*(*k*)將各組分散放入各層

STEP 3.1：從第一層開始依序放置

初始化 $l = 1$

STEP 3.2

For k =1 to P

$Lay(k) = l$

STEP 3.2.1：檢查同層與它組是否有相交

For k' =1 to P

若 $Lay(k') = Lay(k)$且 $Int(k, k')$ 為 True

則 $Lay(k) = Lay(k)+1$ 回 STEP 3.2.1

$l=(l+1) \bmod L$ 回 STEP 3.2

END{ Multi-lay PCB Routing }

四、效能分析

(一) 空間複雜度分析

多層印刷電路演算法的空間複雜度分析，本演算法使用了一個 $X \times Y \times L$ 的三維陣列 $V_{i,j,l}$ 來儲存多層印刷電路板中各個節點的狀態、P 個 AP 二維陣列做為輔助計算用的資料結構，其大小為 $O(P^2)$。其中單層電路板的空間需求為 $N = X \times Y$ 個。又因為總層數 L 為 STEP 2 計算出來的結果，其值必定會比欲連結的總對數 P 為少，所以本演算法的空間複雜度(space complexity) S_{total} 為：

$$S_{total} = O(LN + P^2)$$

其中，$L < P < N$。

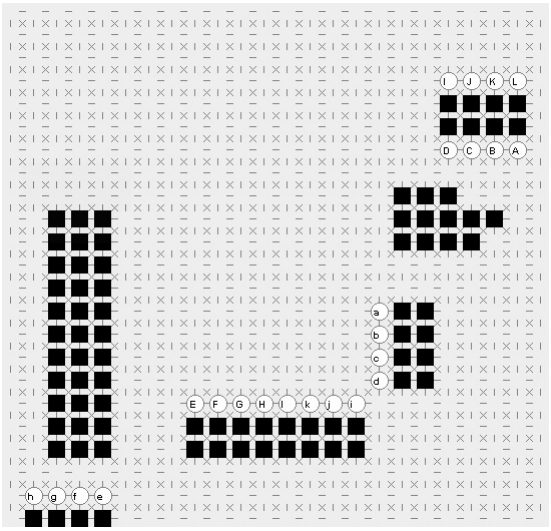
(二) 時間複雜度分析

多層印刷電路演算法主要分為三大步驟，在時間複雜度分析時，STEP 1 為初始化的動作每組要執行一次 HGMR 與計算 AP 中的參數總，共有 P 組，其複雜度為 $O(P(N+1)) = O(PN)$。STEP 2 為每增加一層要執行一次 Single-layer TTN Routing 與 Single-layer PCB Terminal Extension，其總複雜度為 $O(L(PN+N))$。STEP 3 每一組要重新分配到各層去，其總複雜度為 $O(PN)$。

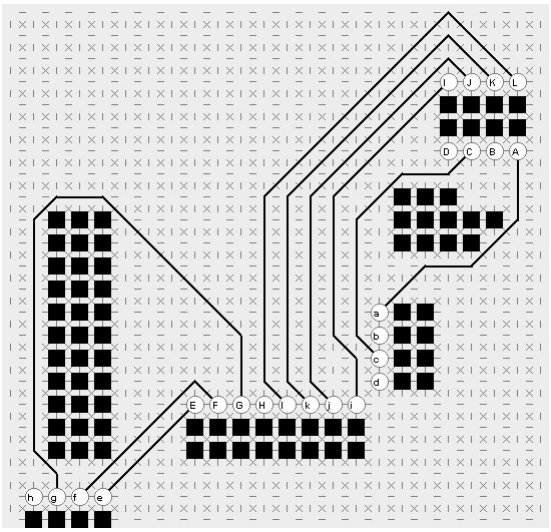
$$\begin{aligned} T_{total} &= T_{STEP\,1}+T_{STEP\,2}+T_{STEP\,3} \\ &= O(PN)+ O(L(PN+N))+ O(PN) \\ &= O(PN)+ O(LPN)+ O(PN) \\ &= O(LPN) \end{aligned}$$

五、程式範例

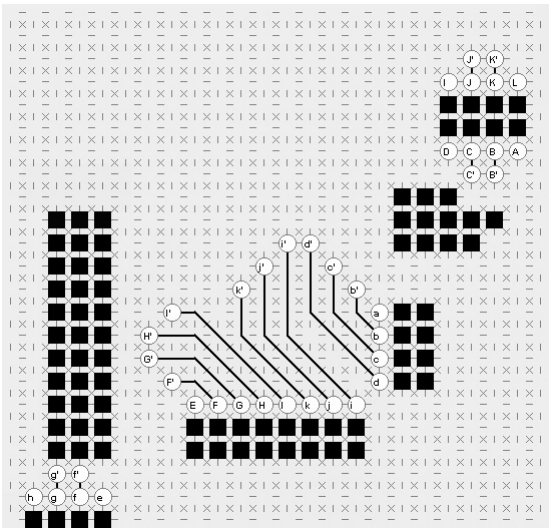
第一個範例為在 24×24 的單層電路板中，有著 12 對的節點要做繞線演算法，其初始狀態如圖四(a)。而直接使用單層多對連結演算法連結時，像圖四(b)中，B-b、D-d 與 H-h 三對並未能達成連結。而利用單層 PCB 連續節點延伸，先將各個鄰近的節點延伸出虛擬節點，如圖四(c)中的 b 節點與 a、c 節點相鄰而延伸到 b’ 節點後，可以在連結時避免掉繞線空間被其它組對佔據，d 節點和 H 節點的情況亦同，因而達到圖四(d)的繞線成果。



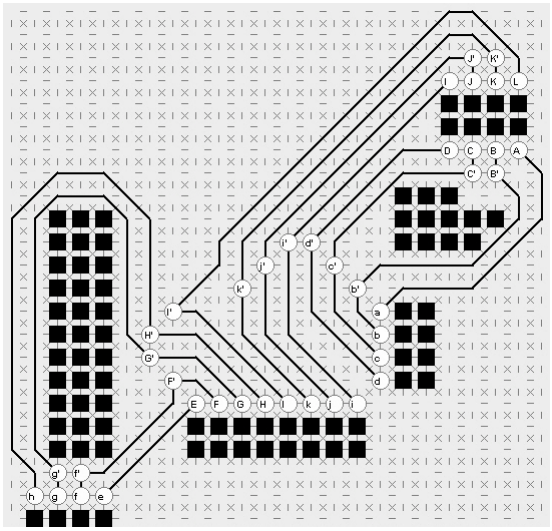
(a)起始狀態圖



(b) 未使用單層 PCB 連續節點延伸而連結失敗



(c)使用單層 PCB 連續節點延伸圖



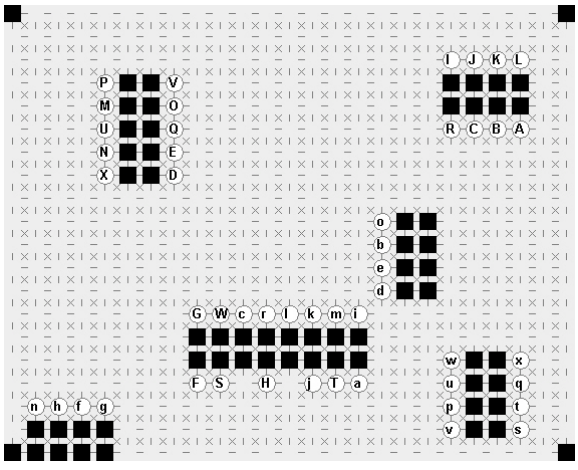
(d)繞線成功結果圖

圖四 單層多對連結比較範例

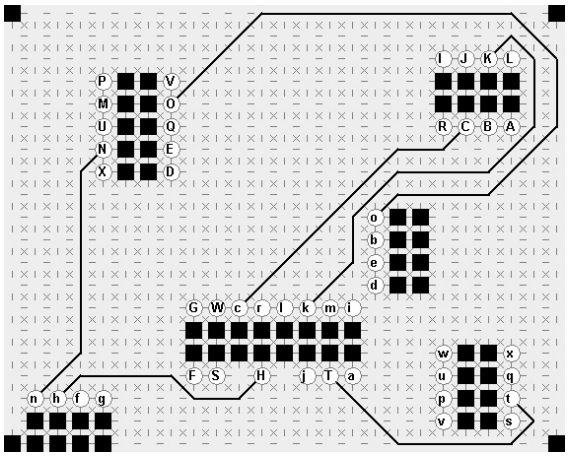
圖五範例為在單層為 25×20 的情形下，有 24 對節點對要相連的初始圖。使用原本單層多對的方法，會一直增加到第五層才繞線完成，圖六(a) ~ (e)，重新改善計算後可以降低至四層即可完成繞線，圖七(a) ~ (d)。

六、結論與未來研究

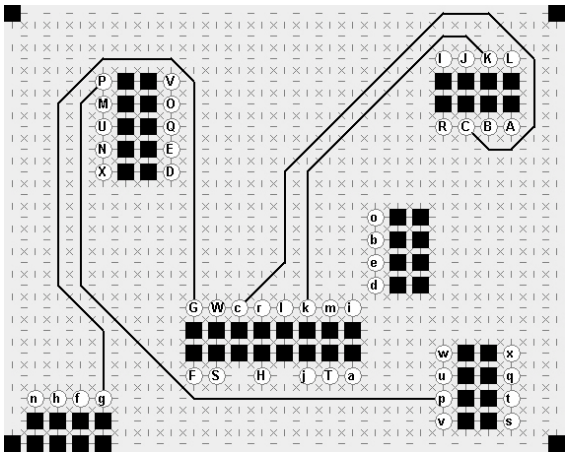
本文章以提出一個迅速判斷多層印刷電路板最多所需層數的方法，其空間複雜度為 O(PN)，時間複雜度為 O(P^2N)，並且在配合著重新加權計算的方式，試著降低印刷電路板多餘



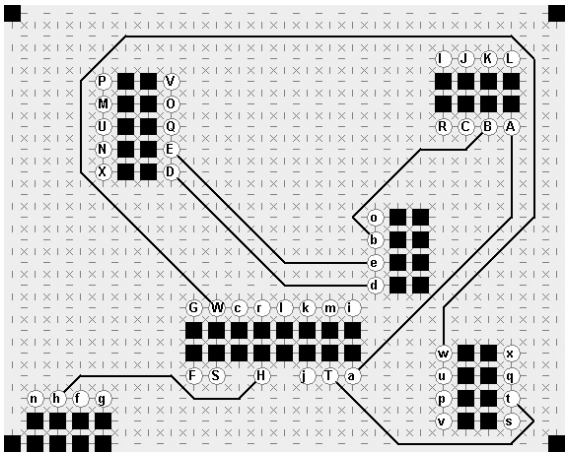
圖五 24 對節點初始圖



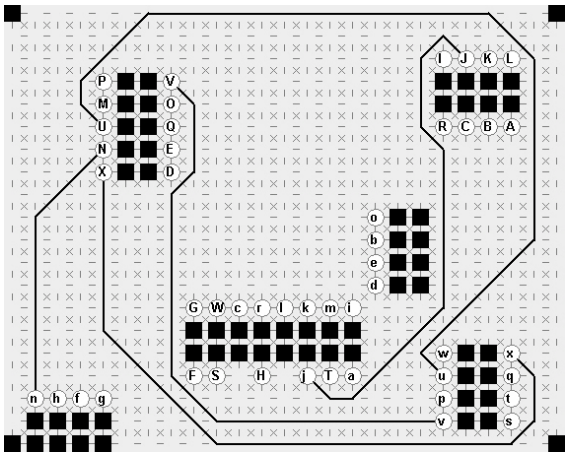
(a) 第一層



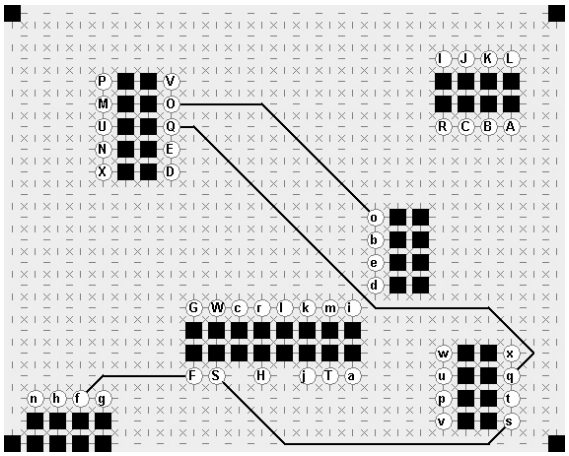
(b) 第二層



(c) 第三層

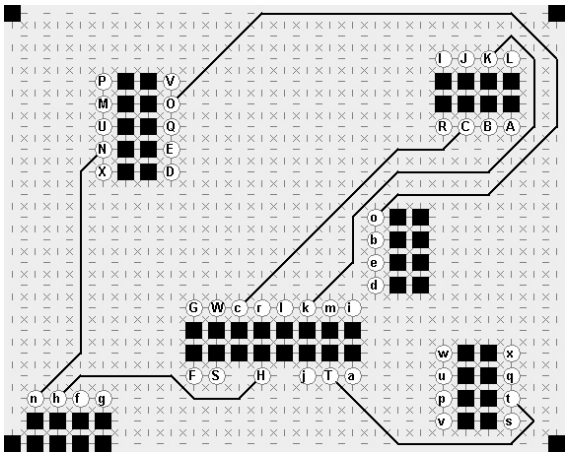


(d) 第四層

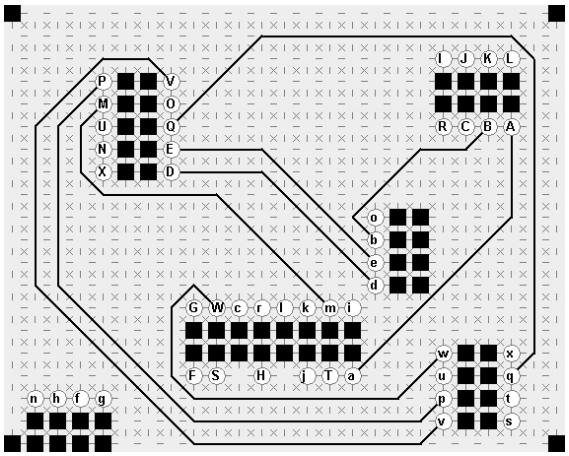


(e) 第五層

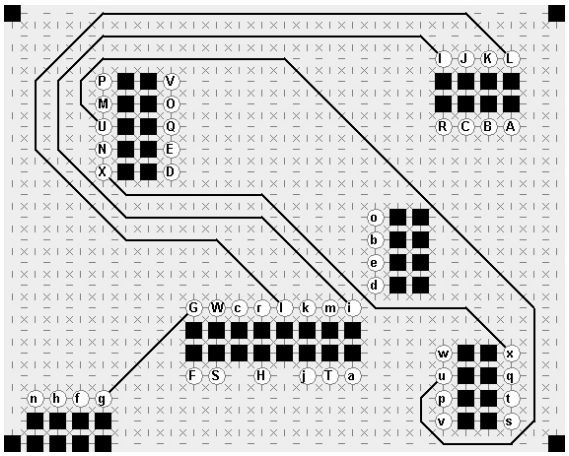
圖六 未改善前繞線結果



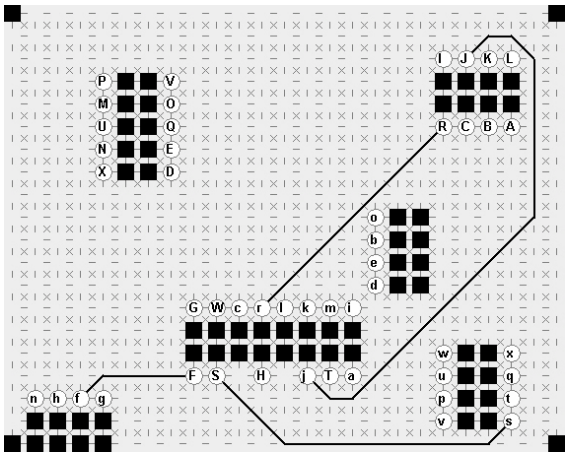
(a) 第一層



(b) 第二層



(c) 第三層



(d) 第四層

圖七 改善後繞線結果

的層數。而在未來的研究中，可以尋求得一個更全面的方法，來降低總金屬線的長度與總層數，並且在各層之間分佈取得平衡。

參考文獻

[1] C. E. Leiserson and F. M. Maley, "Algorithms for routing and testing routability of planar VLSI layouts," 1985, pp. 69-78.

[2] S. Nakatake, K. Fujiyoshi, H. Murata, and Y. Kajitani, "Module placement on BSG-structure and IC layout applications," 1997, pp. 484-491.

[3] H. Murata, K. Fujiyoshi, and M. Kaneko, "VLSI/PCB placement with obstacles based on sequence-pair," 1997, pp. 26-31.

[4] J. D. Carothers, T. Liu, and D. Li, "MCM multilayer routing with layer balancing," 1996, pp. 179-182.

[5] 詹景裕, 劉萬榮, 郭芳誠, 陳永源, "單層多對節點連結演算法," *2007 年全國計算機會議*, 亞洲大學, 2007.

[6] G. E. Jan, K.-Y. Chang, S. Gao, and I. Parberry, "A 4-Geometry Maze Router and Its Application on Multi-terminal Nets," *ACM Transactions on Design Automation of Electronic Systems*, vol. 10, pp. 116-135, 2005.

附錄

(A)山型節點延伸

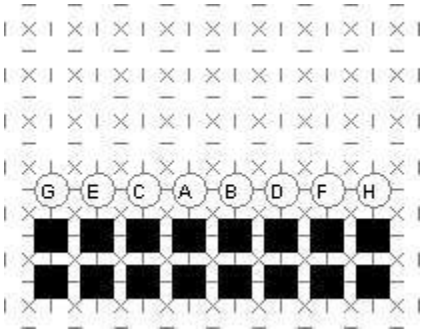
副程式：Mountain Extension (C_m)

Begin

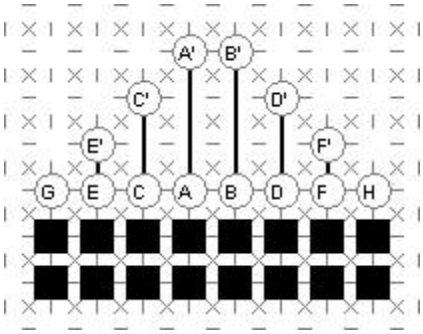
shift_up = 0

For $index'_{cont}$ = 1 to $\left\lfloor \frac{|C_m|-1}{2} \right\rfloor$

$(i_L', j_L') = (i, j + shift\_up)$



(a) 初始的節點位置。



(b) 處理後的節點位置

圖八 針對直線型節點的前處理

$T_{i,j}^{Ext} = T_{i_L', j_L'}$

/*其中 $T_{i_L', j_L'}$ 為 $T_{i,j}$ 山型左邊之延伸點*/

$(i_R', j_R') = (|C_m|\text{-}i, j + shift\_up)$

$T_{|c_m|-i,j}^{Ext} = T_{i_R', j_R'}$

/*其中 $T_{i_R', j_R'}$ 為 $T_{i,j}$ 山型右邊之延伸點*/

$shift\_up = shift\_up + 1$

end for

END{山型延伸}

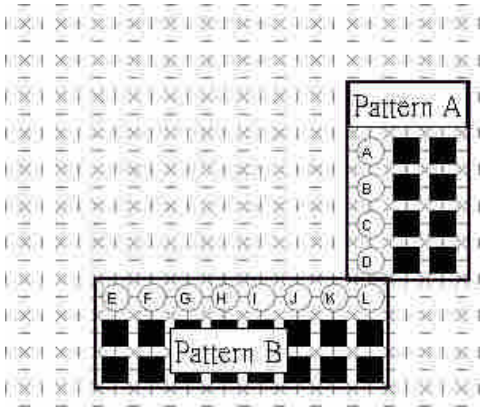
(B) L 型節點延伸

副程式：L-Type Extension (C_{mv} ,C_{mh})

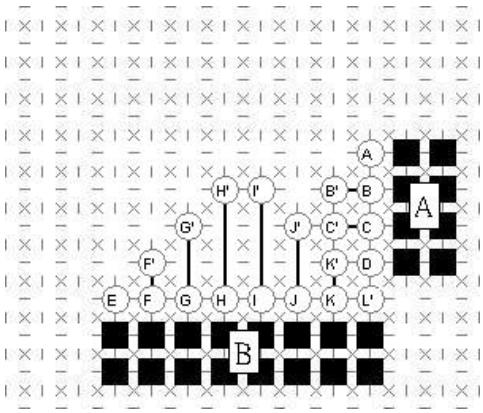
/*其中 C_{mv} , C_{mh} 分別為 L 型中垂直與水平的連續節點*/

Begin

for $index"_{cont}$ = 0 to $|C_{mv}|$



(a) 初始的節點位置。



(b) 節點 k'被卡死

圖九 直接採用直線樣式的前處理

$C'_{mh}.\text{add}(T_{i-index"_{cont}, j+index"_{cont}})$

end for

for $index"_{cont}$ = $|C_{mv}|$ to $|C_{mh}|$

$C'_{mh}.\text{add}(T_{i-|C_{mv}|, j+|C_{mv}|})$

end for

Call **Mountain Extension**(C'_{mh}) 副程式

for $index"_{cont}$ = 0 to $|C_{mv}|$

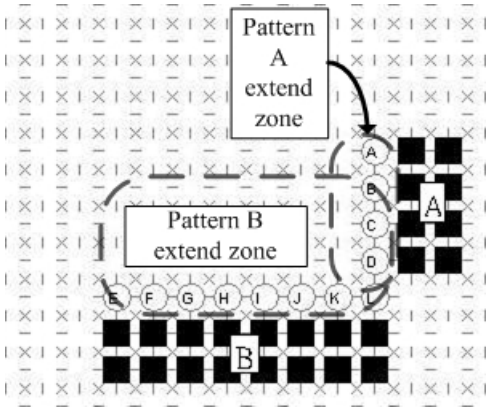
$C'_{mv}.\text{add}(T_{i-index"_{cont}, j+index"_{cont}})$

end for

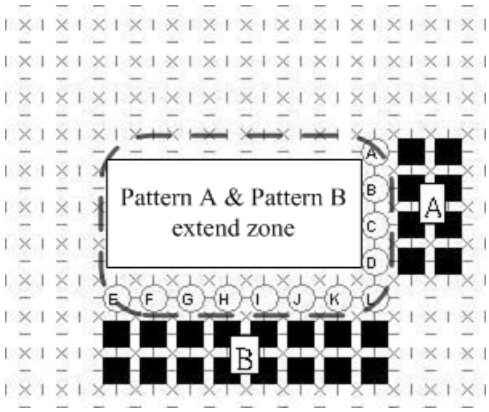
for $index"_{cont}$ = 0 to $|C'_{mv}|$

將 C'_{mV} 中各個節點旋轉 90°

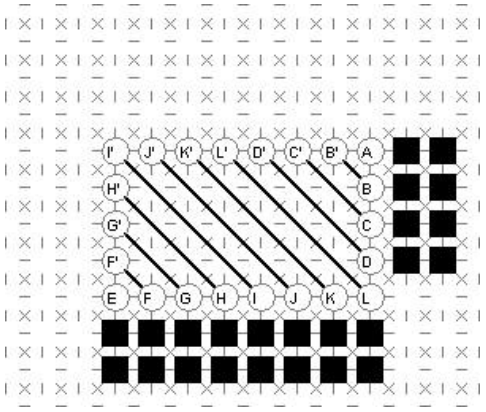
end for



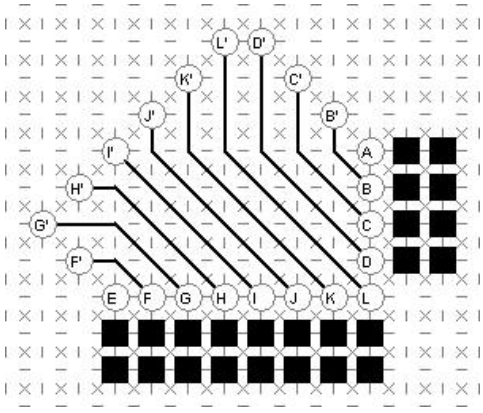
(a) 樣式 A 與 B 擴張區域重合。



(b) 將 A 與 B 的擴張區域進行合拼



(c) 針對 L 型先進行第一次的前處理



(d) 針對直線型再進行第二次的前處理

圖十 針對 L 型節點的前處理

Call **Mountain Extension**(C'_{mv}) 副程式

END{L 型延伸}

若兩個於連續的直線節點樣式方向垂直並且過於接近，如圖九(a)的樣式 A 與樣式 B，可能造成其延伸路徑有所重疊或是卡死，如圖九(b)節點 K'即被卡死在其中。因此要先針對如圖十(a)樣式 A 與樣式 B 的擴展區域計算，並且施行合拼如圖十(b)的結果。再將各節點延伸到樣式 A 與樣式 B 合併後的擴展區域邊緣，如圖十(c)變成為方形的排列。最後再回歸到直線節點樣式的進行第二次的前處理，將各節點依序延伸到形成如圖十 d)的結果。