國立交通大學

資訊科學與工程研究所

碩士論文

串擾效應導向和RLC串擾上限的軌道分配演算法

CROSSTALK-DRIVEN AND RLC-BOUNDED TRACK ASSIGNMENT ALGORITHMS

研究生：駱盈樹

指導教授：李毅郎　教授

中華民國九十四年十一月

串擾效應導向和RLC串擾上限的軌道分配演算法

CROSSTALK-DRIVEN AND RLC-BOUNDED TRACK ASSIGNMENT ALGORITHMS

研 究 生：駱盈樹　　Student：Ying-Shu Lou

指導教授：李毅郎　　Advisor：Dr. Yih-Lang Li

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

Nov 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年十一月

串擾效應導向和 RLC 串擾上限的軌道分配演算法

研究生 ： 駱盈樹　　指導教授 ： 李毅郎 博士

國立交通大學 資訊科學與工程研究所

摘要

隨著超大型積體電路製程的長足進步，現今的積體電路設計已經發展到超微米(VDSM)製程的時代。現在所設計出來的晶片尺寸越來越小，也使得電路繞線的間距變得越來越近，並且電路接線的寬度也相對應地變小。再者，因為積體電路設計的時脈持續在增加當中，目前晶片設計的時脈已經發展到了十億赫茲(Gigahertz)以上。因此，基於以上的理由，使得我們在設計高效能晶片的時候，必須考慮到串擾效應(Crosstalk effect)為電路設計所帶來的有關於電路完整性和正確性的問題。

之前做軌道分配的相關方法，是以區域為基礎的方法。這種以區域為基礎的方法，可能會產生較差軌道利用率的軌道分配結果。在這篇文章中，我們提出了兩個串擾導向的軌道分配演算法。首先提到的，是考量到電容串擾效應的以列為基礎的軌道分配演算法。RBTA 首先利用最左邊端點演算法來增加軌道的利用率，接著，我們提出了一個軌道重疊圖的新概念。藉著軌道重疊圖，我們就能將電容串擾導向的軌道分配問題，轉化為尋找最小成本漢米爾頓路徑(MWHP)的問題，也就是實現了針對串擾導向軌道分配問題，所產生的最佳軌道排序。在實驗數據中顯示出，相較於以前的方法，RBTA 可以減少平均 32.33%的電容串擾效應。與之前的方法相比較，RBTA 也能夠得到較佳軌道利用率的軌道分配結果。

第二個串擾導向的軌道分配演算法，就是針對串擾上限問題，分別考量到電容和電感串擾效應的串擾上限導向演算法(RLC-bounded TA)。我們選擇了以區域為基礎的軌道分配方式來實作這個演算法。我們的演算法由兩個步驟所組成。首先是第一個步驟，利用了我們的啟發式找結黨的演算法。我們將 ***IRoute*** 重疊圖中所找到的結黨，由大到小依序來做處理。我們將結黨相對應的 ***IRoute*** 做軌道分配的動作，產生了一個軌道分配的初步解。接著是第二個步驟，如果，這個結黨中已經分配好的 ***IRoute*** 有產生電感串擾效應的話，我們就用限制搜尋的方法，來進一步地減少電感串擾效應值。在我們的實驗中顯示，當訊號線影響率為 25%，33%，和 50%時，在和第一步驟完成後的串擾成本值比較，電感串擾值分別平均減少了 66.9%，62.80%，和 55.44%。因此，我們知道限制搜尋方法對電感串擾值有明顯的減少作用。而且，訊號線影響率越高的，電感串擾值減少率越低。

CROSSTALK-DRIVEN AND RLC-BOUNDED TRACK ASSIGNMENT ALGORITHMS

Student : Ying-Shu Lou Advisor : Dr. Yih-Lang Li

Institute of Computer Science and Engineering
National Chiao Tung University

ABSTRACT

In this work, we have proposed crosstalk-driven and RLC-bounded track assignment algorithms. First, RBTA first applies LEA to produces a utilization-driven TA, and then transforms the crosstalk minimization problem into finding a MWHP by the proposed new track overlap graph. Experimental results show that RBTA algorithm reduced crosstalk 32.33% and produced fewer failed nets than previous works. In second, we proposed an enhanced finding clique heuristic. For each clique in ***IRoute*** OLG in the decreasing order of clique size, RLC-bounded TA first apply initial assignment to produce a capacitance-free TA result, and then if inductance coupling occurred, we use Tabu search to reduce inductance coupling more. Experimental results show that Tabu search phase reduced inductance coupling 66.9%, 62.80% and 55.44% when the sensitive rate is 25%, 33% and 50%, respectively. In the future, we hope to develop a row-based RLC-bounded track assignment algorithm.

誌　　謝

我非常感謝我的指導教授李毅郎博士，感謝他，對這個研究題目從不間斷地導引，支持和熱心的討論。他很有價值的建議幫助我完成了這個論文。而且，我也要對實驗室同學和學弟們的熱切幫助和鼓勵表達我最誠摯的謝意。

僅將此論文獻給我的父母，家人，和所有關心我的人，謝謝他們的耐心，愛心，和長期以來的鼓勵與期望。

目　　錄

圖 目 錄

表 目 錄

一. 簡介

1.1 電路繞線串擾(Crosstalk)效應和軌道分配(Track assignment)

以下是超大型積體電路(VLSI)繞線的串擾效應和繞線流程中軌道分配步驟的相關介紹。

1.1.1 串擾效應

隨著超大型積體電路製程的長足進步，現今的積體電路設計已經發展到超微米(VDSM)製程的時代[1]。現在所設計出來的晶片尺寸越來越小，也使得電路繞線的間距變得越來越近，並且電路接線的寬度也相對應地變小。再者，因為積體電路設計的時脈持續在增加當中，目前晶片設計的時脈已經發展到了十億赫茲(Gigahertz)以上。因此，基於以上的理由，使得我們在設計高效能晶片的時候，必須考慮到串擾效應(Crosstalk effect)為電路設計所帶來的有關於電路完整性和正確性的問題。

所謂的串擾效應是指當兩條電路繞線相鄰的時候，電路通電時會產生磁場互相干擾影響，以前積體設計的電路由於繞線間距較大，因此效應較不明顯，但是，因為現在的製程進步，電路繞線的間距很小，所以這樣的效應會嚴重地影響到電路訊號傳輸的速度，也會因此造成電路訊號傳輸的不正確性。關於串擾效應的模組已經被徹底的討論[2][3]。根據以前的文獻資料研究指出，串擾效應大致上可區分為兩種較重要的造成因素，一個是電容(Capacitance)所形成的效應，另外一個則是電感(Inductance)造成的影響。電容所形成的效應，影響相鄰繞線的範圍是較小的;而電感造成的效應，所影響相鄰繞線的範圍則是較為廣泛的。

1.1.2 軌道分配

在傳統上，一般的繞線演算法都是將積體電路繞線區分為全域繞線(Global Routing)和細部繞線(Detailed Routing)兩個步驟。會將繞線問題分成兩個階段的主要原因是因為繞線問題太過於複雜，因此，如果直接使用細部繞線的方法去找繞線的解答，則會浪費太多的時間在搜尋不必要的部分，所以，先利用了一個全域繞線的步驟。全域繞線的作法通常是將整個積體電路晶片的繞線區域(Routing region)，切割成相等大小的小區域，也就是所謂的全域細胞(Global cell)，每個全域細胞所能通過的繞線線段，有其上限。而全域繞線所要做的便是要找出每一個訊號線(net)要連接其所屬接腳(pin)會經過哪些的小區域;做全域繞線時，同時也可以在較早的時候考慮到繞線密度的問題，如果有某個區域有太多訊號線經過，可以將其盡量避免。因此，可以大幅的加快細部繞線的速度[4][5]。

但是，由於積體電路的設計越來越趨於複雜化，所以，如果只是區分這兩個

程序來作電路繞線的話，仍會有一些不足的地方。第一，雖然有全域繞線的幫助，但是，由於設計的複雜化，細部繞線還是需要花費相當多的時間來完成整個繞線的流程;第二，在全域繞線和細部繞線這兩個步驟中，並沒有提供適當的機會來處理一些由於訊號延遲，線路串擾效應，或製程限制...等等，所引發的一些問題。因此，一個介於全域繞線和細部繞線之間的步驟，就被發表出來，藉以改善這些問題。也就是將繞線的步驟規劃區分為三個步驟，分別是全域繞線，軌道分配，和細部繞線。我們藉由利用在全域繞線所獲得的一些資訊，就可以在軌道分配時，有效率地處理上述的問題。我們在做繞線線段的軌道分配時，為了減少問題的複雜度，只處理穿透過一整個全域細胞的繞線線段，所以，可以預見的是，大部分較長的繞線線段都已經做好了軌道分配。因此，也就能夠有效率地幫助細部繞線完成整個積體電路的繞線過程，使得我們能夠在整個繞線的流程中，節省許多細部繞線所花費的成本，和改善繞線的品質[6]。

1.2 相關研究和我們的方法

目前已經有許多關於在全域繞線和細部繞線兩個積體電路繞線步驟中，考慮到串擾效應的研究[7][8][9]。但是，在全域繞線中，因為，整個過程並沒有關於軌道的資料，所以，在全域繞線中處理串擾效應的難度，主要是在於缺乏較為詳細的資訊;而在細部繞線時，則是由於在考量到串擾效應時，會需要處理到太多資料，進而會影響到繞線的速度和繞線流程完成的時間。

在軌道分配的步驟時，由於我們處理的都是穿透過一整個全域細胞的平行繞線線段，剛好符合造成串擾效應的線段特性，而且，都是較長的繞線線段。所以，有助於我們改善影響積體電路設計的串擾效應，和大幅減少了細部繞線的複雜度和所需花費的時間。

1.2.1 利用率和串擾效應導向的軌道分配

就我所知，第一個討論到關於串擾效應導向軌道分配的研究文獻[10]，是針對繞線線段的軌道和繞線層的分配(Layer assignment)，推導出了一個有效率的公式化整數線性程序(Integer linear programming)。在這個方法中，關鍵的技巧就是在將關於繞線的幾何(Geometric)和串擾這兩個限制，模組化(model)成一個適當的衝突圖(Conflict graph)中的一些結黨(clique)，並且利用整數線性程序的方法來解決這個問題。這個方法能處理的連接線數目較少，約介於 100~1000 條之間，而且，通常這類的方法都會花費較多的執行時間。

另一個相關的研究[11]，則是在一個多層次的繞線系統(Multilevel routing system)上，做了一個串擾導向的軌道分配。他們利用了水平限制圖(Horizontal constraint graph)來表示平行繞線線段的重疊情形，並且整合了雙分分配圖(Bipartite assignment graph)來紀錄繞線線段分配到軌道上的狀況。他們的方法是以區域分配為基礎的，就是先找到水平限制圖中的最大結黨，也就是相對於實際上，所有要做軌道分配的繞線線段上中重疊的最大子集先做軌道分配，然後，將

串擾導向的軌道分配問題，模組化成一個尋找最小成本的漢米爾頓路徑的問題(minimum weighted Hamiltonian path problem)。當完成了一個水平限制圖中的最大結黨後，便更新水平限制圖，將已經做過軌道分配的繞線線段相對應的點移除掉，再找下一個最大的結黨，依序做完軌道分配。這個方法的缺點就是軌道整體的利用率並不好，而且，針對串擾問題還是有改善的空間。

我們知道軌道分配的問題很類似管道繞線(Channel routing)的問題，兩個問題之間的區別就是要做軌道分配的繞線線段，並沒有垂直限制(Vertical constraint)方面的考量。因為，利用最左邊端點演算法(LEA)[12]處理這類問題時，在沒有垂直限制的情況下，我們可以得到最好的軌道利用率，因此，在我們的以列為基礎的軌道分配演算法(Row-based Track assignment)中，利用了最左邊端點演算法。所以，相對於之前的方法，我們得到了較好的軌道利用率和減少了整個電路設計中更多的串擾效應。

1.2.2 串擾上限(Noise bound)導向的軌道分配

在[13]這篇文章中，使用了長度向量的串擾模組(LSK model)來估算電感串擾值，不同以往的是，這個模組能夠同時考量到相鄰的和不相鄰的，且訊號有互相影響(sensitive)的訊號線。

根據[14]，串擾上限問題是時間複雜度為 np-hard 的問題，所謂的串擾上限問題就是晶片設計中的每個訊號線的連接終點(sink)，都必須滿足晶片設計者所給定的串擾上限值，而且彼此有影響的訊號線段不會被置放在相鄰的位置上，如此，就可以大幅減少電容所帶來的串擾效應。在這篇研究中，在繞線流程的全域繞線步驟中，以針對在同一個全域細胞中的所有線段同時做保護線段的加入(Shield insertion)和繞線線段定序(Net ordering)為基礎，提出了兩個啟發式(heuristic)的演算法，來達到電容和電感串擾效應的最小化。分別是：1.貪婪保護線段加入的串擾上限演算法(Greedy Shield Insertion SINO/NB Algorithm)，2.模擬鍛鍊基礎的串擾上限演算法(Simulated Annealing Based SINO/NB Algorithm)。

另外的一篇研究文獻[15]，一樣是在全域繞線的步驟中，針對同一個全域細胞中的所有線段，利用了限制搜尋(Tabu search)來做同時地保護線段加入和繞線線段定序(SINO)的演算法，這個方法在一些特定的問題上，運作的方式和效能與之前提到的模擬鍛鍊演算法(SA)類似，但是，執行速度較為快速。

我們的方法和之前的方法不同的地方是在於我們並不是完全地在全域繞線的步驟中，處理滿足串擾上限的問題，而是整合了全域繞線和軌道分配兩個繞線步驟共同來處理串擾上限的問題。首先，當全域繞線完成後，便會進到軌道分配的步驟，而軌道分配所要達到的目標便是要滿足串擾上限問題的要求，並且盡量使得整個設計的串擾值能夠最小化，其中，串擾值包含了電容和電感影響產生的串擾值。我們的方法改變了之前所提到的繞線三步驟的流程，原本，軌道分配完成後，就要進到下一個繞線階段，也就是細部繞線。但是，在這裡，我們完成了軌道分配之後，會再回到全域繞線的部分，去估算每個連接線的連接終點(sink)是不是有滿足設計者給定的串擾上限，若不滿足，則做繞線線段的拔除與重繞(rip-up reroute)，和加入保護線段，來達到滿足串擾上限的目的。圖一便是我們關於串擾上限問題提出的繞線流程。

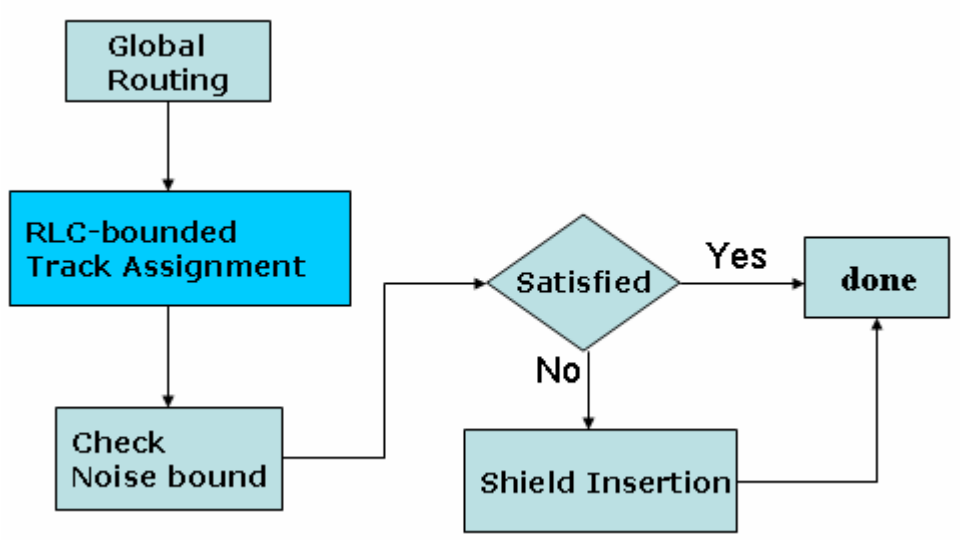


Figure 1. Our Routing Flow

二. 緒論

2.1 繞線和串擾的模組(model)

2.1.1 繞線模組

在我們的方法裡，繞線依循著限制性的繞線層模組，也就是說一個繞線層只限定一個方向的線段連接，水平或是垂直方向。若是在水平繞線層繞線的話，當訊號線段要走垂直方向的話，則必須付出昂貴的代價。相鄰的繞線層，其繞線方向是不同且正交的，而且，所有的訊號線皆是由曼哈頓(Manhattan)形狀所組成。在同一個繞線層中，所有要繞的訊號線都有一致的線寬規則(width rule)和線距規則(spacing rule)。但是，我們允許在不同的繞線層有不一樣的線寬規則和線距規則。

全域繞線在為了達到總繞線長度最小化的目標下，並且在沒有違反每個全域細胞(GCell)所擁有的繞線資源的情況下，分佈所有的訊號線。基於每個繞線層的積體電路設計規則的線寬規則和線距規則，以及考量到每個全域細胞的面積，一個全域細胞的繞線資源被當作分開，且等距離的格線(grid line)，這裡稱為***軌道***(Tracks)。所有我們要繞的訊號線，都必須被分配到這些軌道上。

軌道分配的其中一個目標就是要減輕細部繞線的負擔，並且改善繞線的品質，也就是要產生較多筆直且長度較長的繞線結果，因此，長度長的訊號線段要比長度短的訊號線段來得重要的多。而且，只處理較長的訊號線段，也可以降低軌道分配問題的複雜度。所以，在軌道分配的方法中，我們只處理穿透過一整個的全域細胞的繞線線段，也就是長度較長的線段，這裡稱之為 ***IRoute***。

這裡我們還要定義一個被稱為***嵌板***(Panel)的物件，一個嵌板是由一個繞線區域中一整列或是一整行的全域細胞所組成，而且水平嵌板裡的軌道都是走水平方向的，相對的，垂直嵌板裡的軌道都是走垂直方向的。

圖二是繞線模組和上述名詞定義的一個例子。

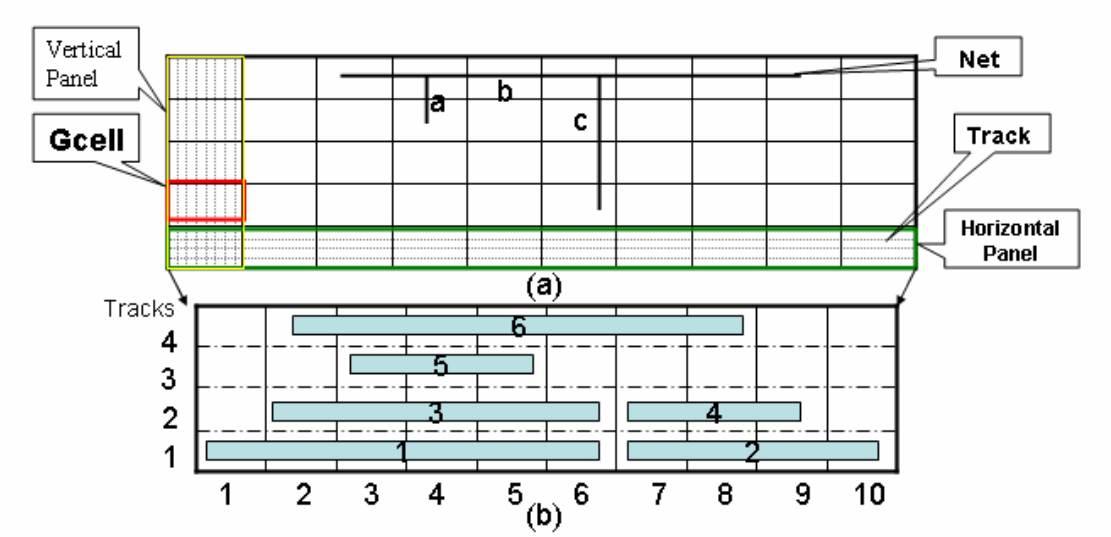


Figure 2(a) is a routing region consisting of 5 x 10 GCells, and shows some definitions in the example. In the case, the net segment a isn't an ***IRoute***, but b and c are both ***IRoute***s.(b) shows a track assignment result with six net segments in a horizontal panel.

2.1.2 串擾模組

本文中，我們針對串擾導向的軌道分配，提出了兩個啟發式的演算法，其中使用到兩個不同的串擾模組，分別是[11]中所討論到的關於電容產生的串擾效應的模組，和[13]中所提出的估量電感產生的串擾效應的模組。

(1)電容串擾效應模組：電容的串擾效應(C_c)在當兩個相鄰的繞線線段間距較遠時，會迅速的衰減，為了簡單化電容串擾效應的模組，所以，我們只考量到相鄰的繞線線段所產生的影響[8]。而且，在這個模組中，我們將處在不同繞線層和不同繞線方向的繞線線段，彼此之間是視為沒有串擾效應的產生。另外，基於我們的繞線模組，因為，每個繞線線段的間距都是一樣的，所以，我們的電容串擾模組，只需要利用相鄰的繞線線段重疊的長度來做估算。

(2)電感串擾效應模組：我們引用在[13]這篇文章中所提到的電感串擾效應估算模組，下圖三是電感串擾係數估算的說明。

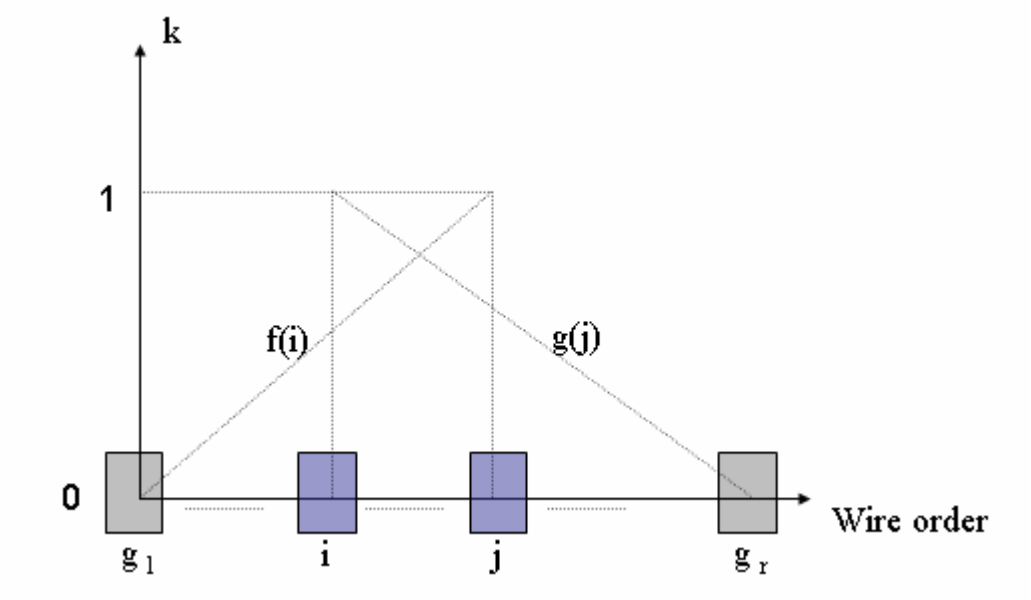


Figure 3.Illustration of K_{ij} computation. N_i and N_j are two signal wires in the same block sandwiched by ground wires g_l and g_r. f(i) and g(j) are two linear interpolation functions as shown by the sloping dotted line. The mutual inductive coupling is given by the mean of f(i) and g(j) .

在一般的情況下，兩個互相有串擾影響的訊號線N_i, N_j，在同一個集塊(block)中的的電感串擾係數，K_{ij} 和 K_{ji} 是相等的，而且這個數值很接近(1)式中的值。

$$K_{ij} = \frac{f(i) + g(j)}{2} \tag{1}$$

其中，f(i)和g(j)是兩個線性插補(interpolation)函式，$f(i) = (i - g_l) / (j - g_l)$，而$g(j) = (g_r - j) / (g_r - i)$。

同時，在一個繞線區域t中，繞線線段N_i所受到電感串擾效應的值為(2)式中所示，其中，N_j 表示在區域t中，跟N_i 互相有影響的繞線線段。

$$K_{it} = \sum_{j \neq i} k_{it,\ jt} \tag{2}$$

所以，由LSK模組，我們得到(3)式中，繞線線段N_i 的LSK值，也就是電感串擾值。其中，l_t 是繞線區域t的長度。

$$LSK_i = \sum_t l_t \cdot K_{it} \tag{3}$$

2.2 限制搜尋(Tabu Search)

限制搜尋最早是在[16]中所提出的，這個方法現在已經被廣泛的使用來處理需要極多計算時間的問題。例如，時間複雜度為 np-hard 的一些難題。限制搜尋與模擬鍛鍊(SA)的方法類似，在一些特定問題上，兩者的執行效果差不多，但是，執行速度以限制搜尋較為快速。

限制搜尋主要的技巧在於將已經到達過的區域最小值(local minimum)紀錄下來，並且禁止再到訪達到這個區域最小值的情況，因此，我們可以避免被這些區域最小值的情況侷限住，所以，最後，可以求得全域最小值的結果。
下圖四是從[15]中，引用的限制搜尋演算法的大概描述。

1.Select an initial solution x^{now},
and set Tabu list H=empty;
2.While not meet the stop conditions do
Generate a candidate list Can_N(x^{now}) from the neighborhood N(x^{now}, H) of x^{now} that doesn't conflict with H;
Select the best solution from Can_N(x^{now}) : x^{new};
x^{now}=x^{new};
Update Tabu list H;
End While

Figure 4. Outline of the Tabu Search algorithm

2.3 問題公式化(Problem Formulation)

T 是一個嵌板裡軌道的集合，I 則是這個嵌板裡需要被分配到軌道裡的 ***IRoute***的集合。每個軌道 $t \in T$ 能夠被表示成一個連續區段(interval)組成的集合。我們把這些區段表示為 x_i，所以，$t \equiv \bigcup x_i$。每個 x_i 可以是：

(1)一個被封鎖的區段，沒有任何***IRoute***能被分配到這個區段。

(2)一個被佔據的區段，某個***IRoute***已經被分配到這個區段上，或者是

(3)一個空置的區段，還沒有任何***IRoute***被分配到這個軌道上。

IRoute $ir \in I$，如果 $X_i \cap ir \neq \phi$ 就表示 x_i 是一個空置區段或者是這個區段已經被同一個訊號線的某個線段佔據住了，也就是說*ir*能夠被分配到軌道t上，$t \equiv \bigcup x_i$。

在這裡我們要定義兩個軌道分配的問題：

(1)串擾導向的軌道分配問題(Crosstalk-driven track assignment problem)：

首先，給定一個軌道的集合 T，一個 ***IRoute*** 的集合 I，還有一個成本函式 $F: I \times T \rightarrow N$ 來表示一個 *IRoute* 被分配到軌道上，所產生的串擾效應的大小，我們的目標就是要找到一個軌道分配的結果，使得整個設計中的串擾效應值最小化。

(2)串擾上限導向的軌道分配問題(Noise-bound driven track assignment problem)：

一樣，先給定一個軌道的集合 T，一個 ***IRoute*** 的集合 I，和一個紀錄訊號線互相有影響關係的集合，最後，還要有一個成本函式 $F: I \times T \rightarrow N$ 來表示一個 *IRoute* 被分配到軌道上，所產生的串擾效應的代價，我們的目標就是要找到一個軌道分配的結果，來達到以下兩個目的：

1.彼此有影響關係且所在區段位置重疊的 *ir* 不會被分配在相鄰的軌道上。

2.使整個設計中的電感串擾總值(Total LSK value)最小化。

三.串擾效應導向的演算法

3.1 以列為基礎的軌道分配演算法

我們的軌道分配是針對每個嵌板來做的，每個嵌板裡的所有 ***IRoute*** 可以表示成一個 ***IRoute*** 重疊圖(***IRoute*** OLG)，這個重疊圖用來紀錄嵌板裡所有 ***IRoute*** 所處位置重疊的情況，其中，圖上的點代表的是 ***IRoute***，而兩個點之間有邊的話，表示兩個 ***IRoute*** 的繞線位置有重疊到，如圖 5(a)中有 11 個 ***IRoute*** 分別是標示為 1~11，而這個嵌板裡共有 5 個軌道，圖 5(b)是圖 5(a)的 ***IRoute*** 重疊圖。

而且，我們也利用了雙分分配圖，來紀錄嵌板中的 ***IRoute*** 可被分配到哪個軌道上的資訊，雙分圖中左邊的點代表了嵌板裡的 ***IRoute***，而右邊的點則是表示嵌板裡的軌道。雙分圖中的左右集合若有邊相連，則表示相對應的 ***IRoute*** 可以被分配到這個軌道上，圖 5(c)是圖 5(a)的一個雙分分配圖的例子。

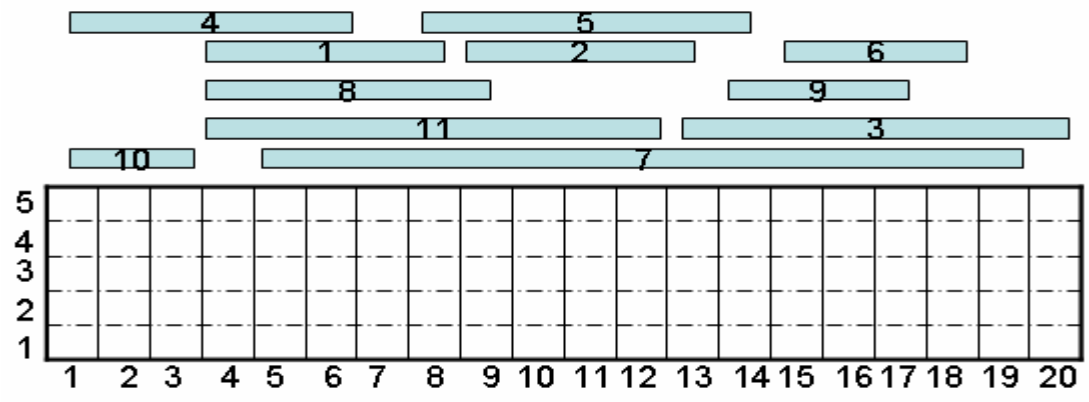


Figure 5(a)

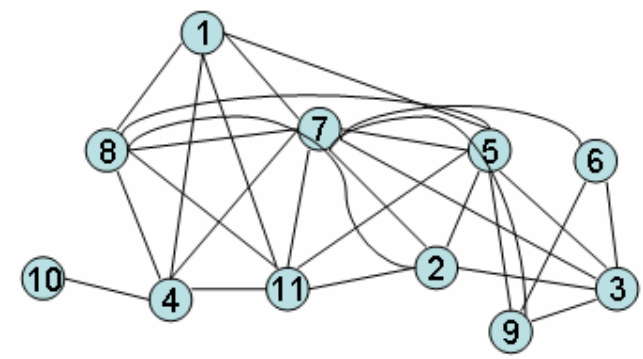
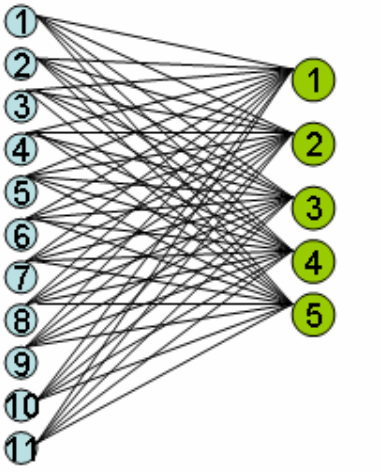


Figure 5(b). ***IRoute*** OLG



5(c).bipartite assignment graph

3.1.1 以區域為基礎的串擾導向軌道分配演算法

[11]中的串擾效應導向的軌道分配方法，是以 ***IRoute*** 重疊圖為基礎，從圖中最大的結黨(clique)，如圖 5(b)中的{1,4,7,8,11}，開始做軌道分配的動作。

他們將串擾效應導向的軌道分配問題，模組化成找最小成本的漢米爾頓路徑的問題（MWHP）。***IRoute*** 重疊圖上的邊的成本是相對應兩個 ***IRoute*** 重疊的長度。他們解 MWHP 的方法，首先是將這個結黨中，相對應的 ***IRoute*** 長度最長的，先分配到這個 ***IRoute*** 能配置的第一個軌道上，也就是視為找 MWHP 的第一個起始點，然後，選擇與第一個被分配好的 ***IRoute*** 重疊長度最小的 ***IRoute***，設定為第二個拜訪的點，並且將這個 ***IRoute*** 分配到第一個能被配置的軌道上。利用這個啟發式的方法，依序將這個結黨中所有相對應到的 ***IRoute*** 分配到不同的軌道上，以求達到串擾最小化的目標。圖 6(a)是利用圖 5(a)的例子來說明[11]中所提到的軌道分配方法。首先找到 ***IRoute*** 重疊圖中的最大結黨{1,4,7,8,11}，然後，將其中長度最長的 ***IRoute***，也就是 ***IRoute*** 7 分配到第一個能配置的軌道 1 上，接著，選擇第二個拜訪的點，也就是與 7 相連的邊上成本最小的點，在這個例子中，點 4 便是第二個要被選擇的點，將 ***IRoute*** 4 分配到其第一個能配置的軌道 2 上，依照著這個方法，依序拜訪結黨中還未被拜訪的點。也就是，將 ***IRoute*** 1,8,11 分別分配到軌道 3,4,5 上。

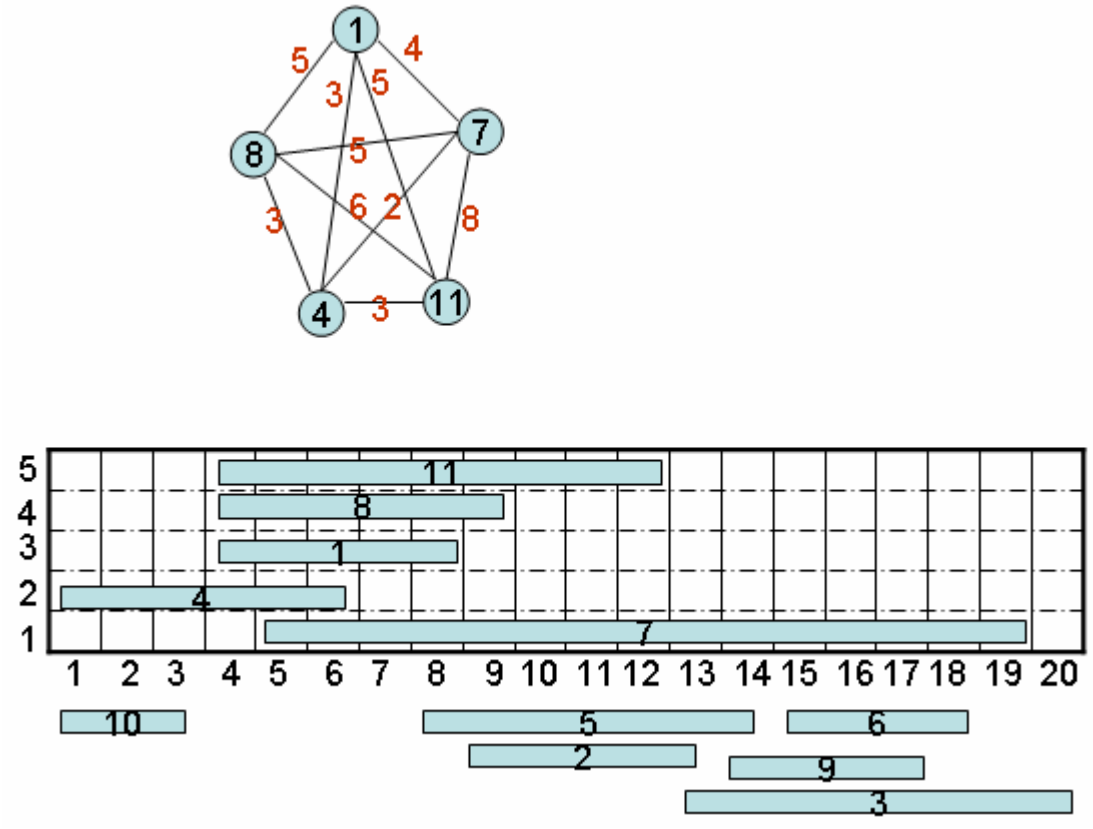


Figure 6(a) Process the maximum clique{1,4,7,8,11} of the ***IRoute*** OLG.

接著，把圖 5(b)中的 ***IRoute*** 重疊圖做更新的動作，也就是把剛剛上面找出來的結黨{1,4,7,8,11}，所對應的點和其伴隨的邊移除。然後，再從這個圖中，找最大的結黨，利用剛剛上述找 MWHP 的方法，繼續做軌道分配的動作。直到，***IRoute*** 重疊圖中的點都被移除掉了為止。

圖 6(b)~(d)是整個軌道分配的流程。

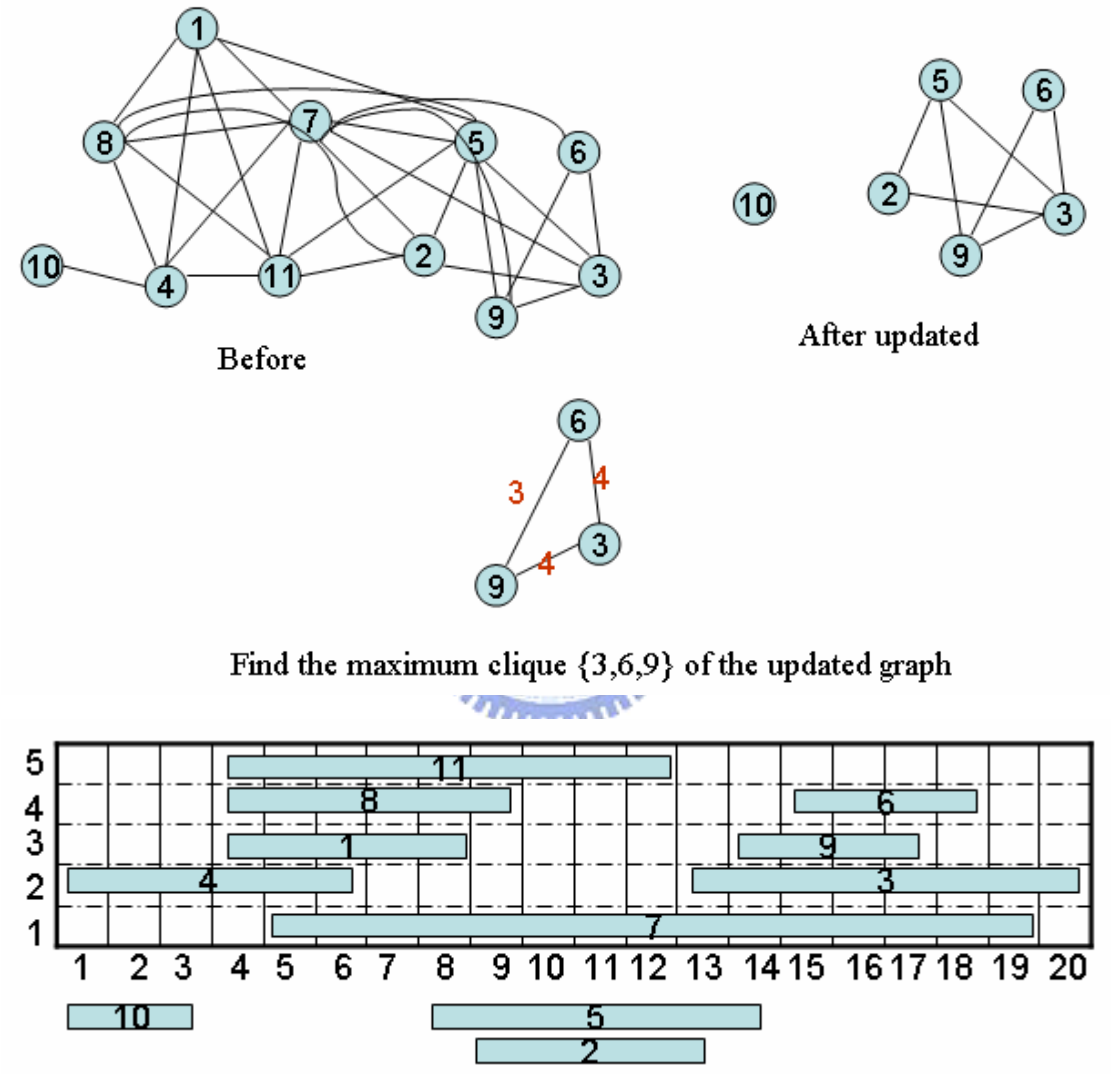
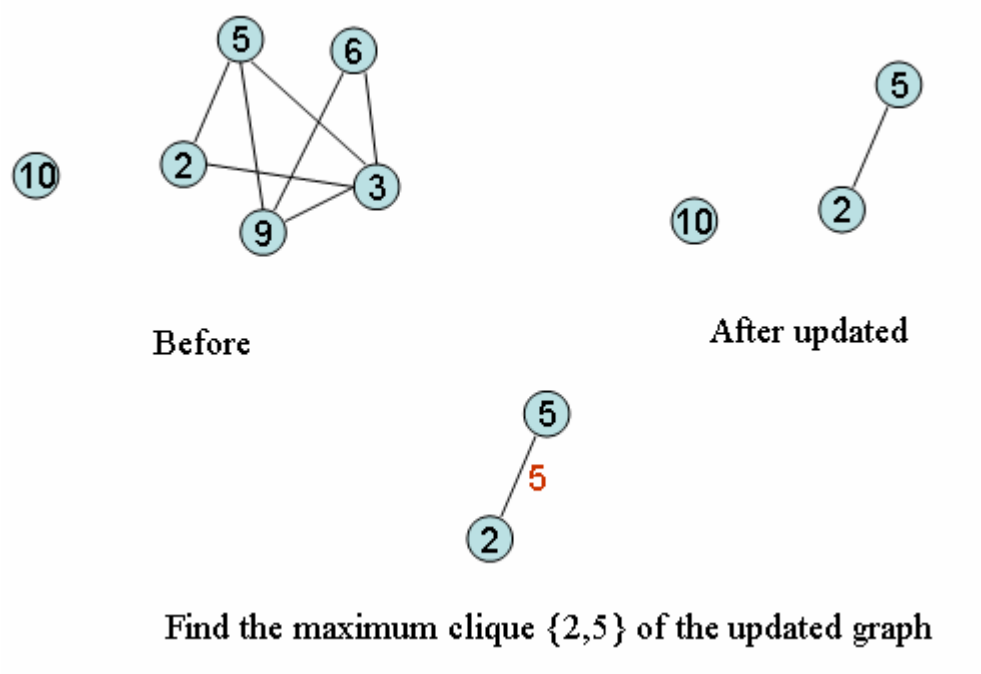


Figure 6(b) Process the clique {3,6,9}.



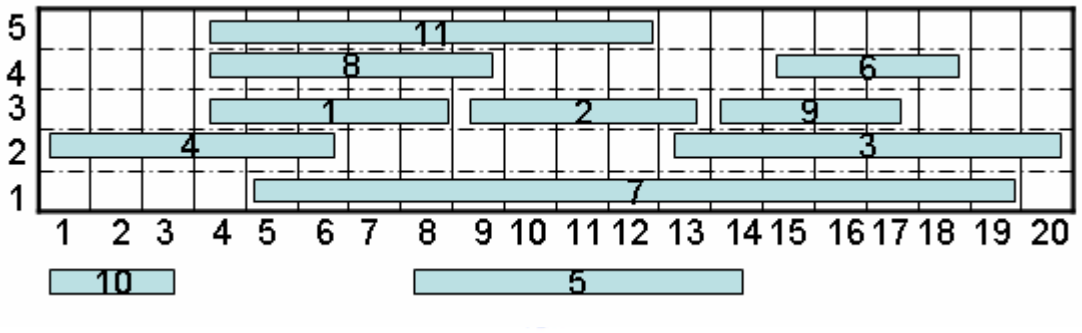


Figure 6(c) ***IRoute*** 5 can't be assigned.

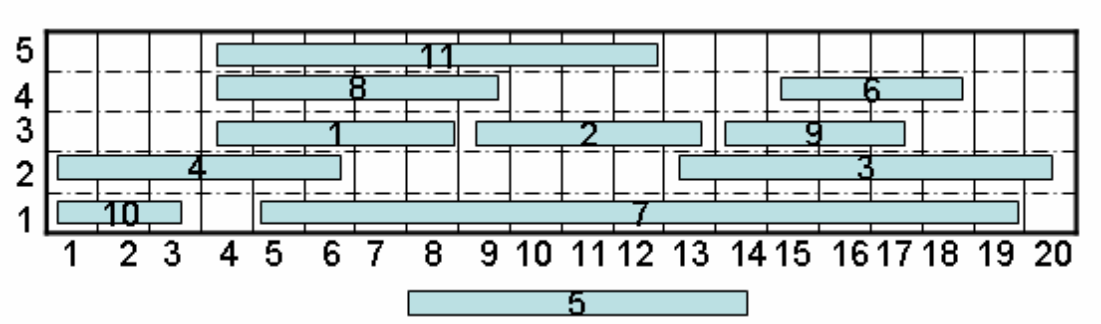


Figure 6(d) The final track assignment result which the ***IRoute*** 5 can't be assigned.

然而，這種以區域為基礎的軌道分配方式，可能會產生不好的軌道利用率的軌道分配結果，而使得固定大小的嵌板裡的 ***IRoute*** 沒有辦法完全地被分配到軌道上，如圖 6(d)的軌道分配結果。

而且，這種以區域為基礎的的軌道分配方式，在串擾效應值最小化的方面，由於，沒有考慮到區域與區域之間的關係，因此，還有可以改善的空間。

3.1.2 以列為基礎的軌道分配演算法

為了改善以區域為基礎的軌道分配演算法的缺點，我們提出了以列為基礎的軌道分配演算法。

我們的以列為基礎的軌道分配演算法(RBTA)，提出了一個新的概念，就是建構了軌道重疊圖，來改善軌道的利用率和串擾的最小化。

以列為基礎的演算法是由兩個步驟所組成的，第一個步驟是利用最左邊端點演算法，來得到一個具有良好軌道利用率的軌道分配初步解，其中要注意的是，若是兩個 ***IRoute*** 的左邊端點位置一樣的話，為了配置更多的 ***IRoute***，因此，我們選擇先分配長度較短的 ***IRoute***。

圖 7(a)是一個利用最左邊端點演算法運作後的軌道分配的初步解的例子。

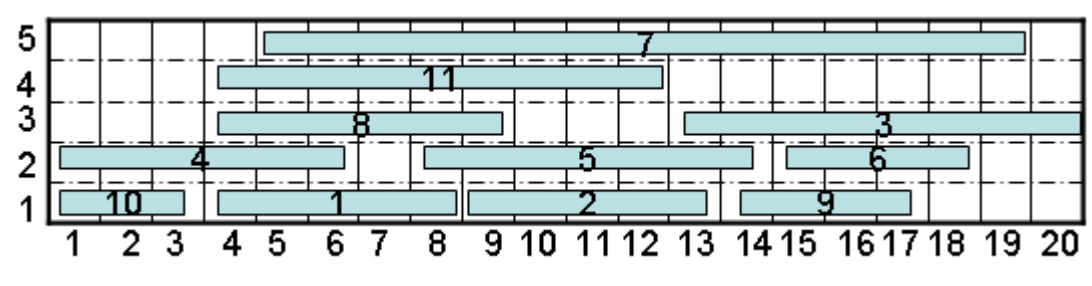


Figure 7(a). the TA initial solution after applying LEA.

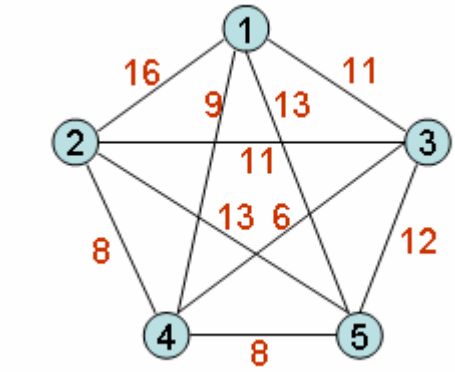


Figure 7(b). Fig7(a)’s related track OLG.

演算法的第二個步驟則是將第一步驟中得到的軌道分配初步解，做電容效應最小化的動作。由於，軌道分配問題中的 ***IRoute***，並沒有像管道繞線中的繞線線段一樣有垂直限制的要求。因此，我們可以藉由軌道的排列來做電容串擾效應的最小化。

IRoute 重疊圖的概念，可以被擴展到整個軌道。當被分配到兩個軌道上的任何 ***IRoute*** 的繞線位置有重疊到的話，我們就稱這兩個軌道，彼此是互相重疊的。根據最左邊端點演算法的特性，嵌板裡的每個軌道跟其他軌道都是彼此重疊的。除此之外，兩個沒有重疊的軌道，可以被合併，並且被視為是一個軌道。

因此，軌道重疊圖能夠被定義為之前所觀察到的特性。圖上的每個點代表嵌板裡的軌道，圖上的邊表示兩個軌道有重疊的關係。一個軌道重疊圖必須是完全圖。邊上的成本代表所有的電容串擾效應，例如：一個邊所連接的兩個軌道上所容納的所有 ***IRoute*** 重疊的總長度。圖 7(b)展示了圖 7(a)的軌道重疊圖。

串擾最小化的問題可以被模組成為在軌道重疊圖上找最小成本的漢米爾頓路徑的問題（MWHP）。這個問題可以用[11]中所提到的啟發式演算法來解決。

MWHP 搜尋程序的起始點是所有邊上成本總和最大的點。起始點相對應的軌道就是有最好的軌道利用率的軌道。與起始點相連且成本最小的邊會被包含進漢米爾頓路徑裡，但是，這個邊所連接的點必須是漢米爾頓路徑之前沒有拜訪過的點。這個新拜訪到的點，就變成下一個迭代(iteration)的起始點。這個步驟一直持續到所有的點都被拜訪過為止，如圖 8(a)~(e)的流程。沿著最小成本漢米爾頓路徑的點，組成了嵌板裡軌道的排列順序。舉例來說，利用這個演算法在圖 7(b)中找到的最小成本漢米爾頓路徑便是{1,4,2,3,5}，也就是執行這個演算法後，軌道的排列順序。圖 7(c)展示了基於 MWHP 方法的軌道排列後的最終結果。圖 7(a)中，分配在相鄰軌道上的 ***IRoute***，總共的重疊長度是 82，而圖 7(c)中，總共的重疊長度是 80。因此，我們的以列為基礎演算法在針對電容串擾效應方面有所改善。

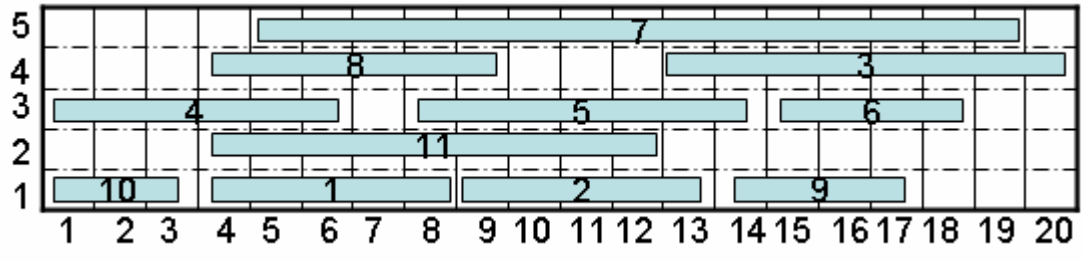


Figure 7(c). shows the final TA after track permutation based on the MWHP.

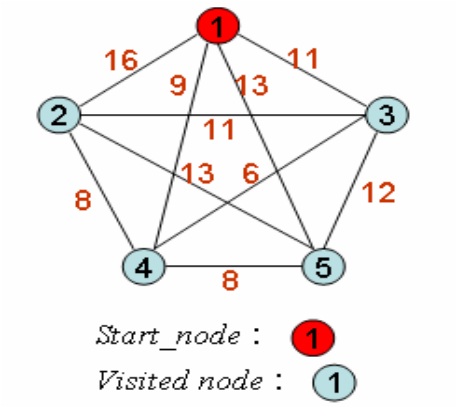


Figure 8(a). At first, we select node 1 to be the *Start_node* with the maximum total weight on its incident edges.

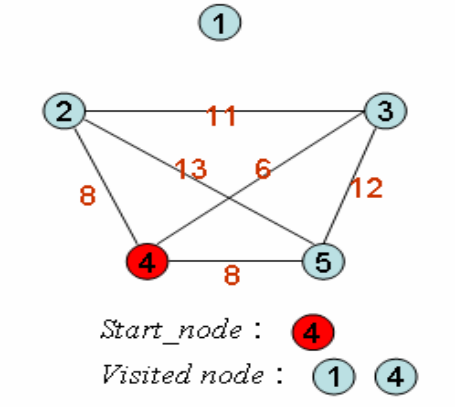


Figure 8(b). Select node 4 to be the next *Start_node* with the minimum weight on the edge connected node 1 and 4. And remove the edges connecting the visited nodes from the track OLG.

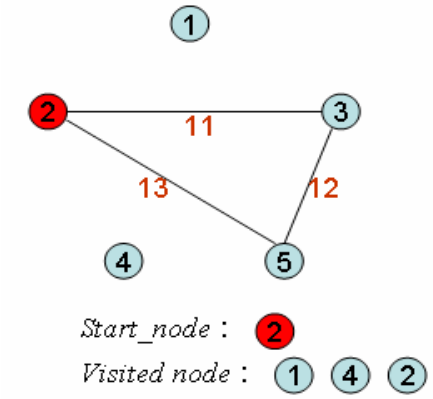


Figure 8(c). Select node 2 to be the next *Start_node* with the minimum weight on the edge connected node 4 and 2.
And remove the incident edges.

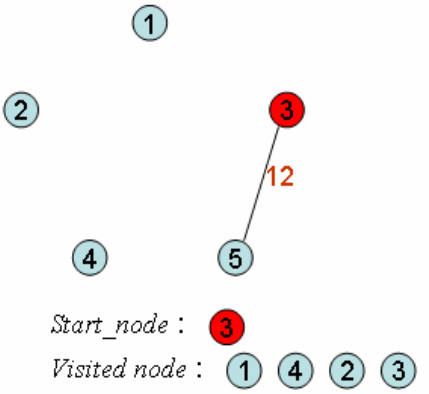


Figure 8(d). Select node 3 to be the next *Start_node* with the minimum weight on the edge connected node 2 and 3.
And remove the incident edges.

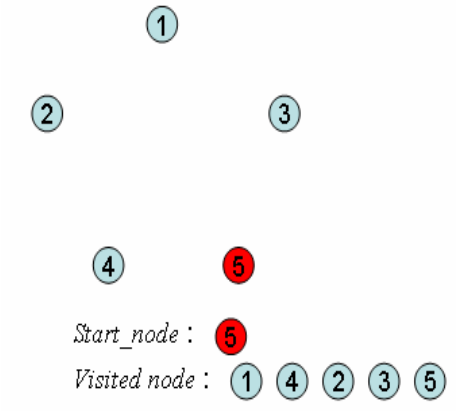


Figure 8(e). Finally, we add the last visiting node 5 to the path.
So, we find the minimum weighted Hamiltonian path {1,4,2,3,5}.

下圖 9 描述了以列為基礎的軌道分配演算法。

```
1.  Algorithm: RBTA
2.  Input: A global routing result for a panel
3.  Output: a track assigned panel
4.  begin
5.  Apply LEA to produce a utilization-driven TA;
6.  Construct the track OLG of the initial TA;
7.  start_node = the node of the totally maximum weight
        of its incident edges;
8.  Set all nodes unvisited; set start_node visited;
9.  while (there is unvisited nodes)
10.     Include the least-weighted incident edge of the
        start_node in the Hamiltonian path;
11.     Set the other connected node of the newly selected edge
        visited;
12      Remove the edges connecting the visited nodes from
        the track OLG;
13. end
```

Figure 9. The RBTA algorithm.

3.2 串擾上限導向的軌道分配演算法

我們的這個軌道分配演算法主要是要達到以下這兩個目標，首先是，彼此會造成串擾效應影響的兩個 *IRoute*，在位置有重疊的情況下，必須確保不會被分配到相鄰的軌道上，這是為了減少電容串擾效應對線路的影響。第二個目標則是，希望能使得每個嵌板裡，所有的 *IRoute* 所產生的電感串擾效應總值，能夠達到最小化。

在串擾上限導向的軌道分配演算法中，我們仍舊是針對設計中的每個嵌板來做處理。我們知道軌道分配演算法，大致上，可以分為兩個思考方向，一個是以列為基礎的軌道分配，另外一個則是，以區域為基礎的軌道分配方式。

若是以區域為基礎的軌道分配方式，來實作這個軌道分配演算法，我們能夠保證在完成軌道分配演算法之後，所有彼此有影響關係的 *IRoute* 不會被分配到相鄰的軌道上。至於另外一種選擇，也就是以列為基礎的軌道分配方式，依照之前的方式，選擇最左邊端點演算法來做 *IRoute* 的初始分配之後，由於，我們沒考慮到訊號線彼此影響的關係，因此，可能會產生彼此有影響的 *IRoute* 會被分配到相鄰軌道上的結果。此時，我們依舊相同地建立這個軌道分配初步解的軌道重疊圖。並且，如果是彼此有影響關係的 *IRoute* 被分配到相鄰的軌道上的話，我們就將軌道重疊圖中，這兩個軌道間相對應的邊移除掉，藉以產生一個滿足要求的軌道排序。在移除這些邊後，我們可能沒辦法找到一個路徑，能夠去拜訪所有的點，也就是由於訊號線互相影響的限制，並沒有適合的軌道分配結果存在。這就是利用以列為基礎的軌道分配方式潛在的問題。

3.2.1 以區域為基礎的軌道分配

透過以上兩種軌道分配方式的討論。因此，我們目前的軌道分配演算法，是使用了以區域為基礎的軌道分配方式。以區域為基礎的主要想法，是希望從整個嵌板中，*IRoute* 密度最高的地方，也就是整個嵌板最關鍵的地方，開始做軌道分配的處理。

但是，我們的以區域為基礎軌道分配方法，與之前[11]中所提到的方式，並不太一樣。由圖 6 的例子，我們可以看到，以區域為基礎的軌道分配方式，可能會造成不良的軌道利用率，導致有些 ***IRoute*** 無法被分配到軌道上。

透過觀察我們發現，每當找到 ***IRoute*** 重疊圖中的最大結黨，並且做完相對應的繞線線段的軌道分配後，就去更新 ***IRoute*** 重疊圖，將之前找到的結黨的點和其相伴隨的邊移除掉。再找更新後的 ***IRoute*** 重疊圖中的最大結黨，這樣的方式，有時候，無法找到嵌板中真正關鍵的區域。就如同圖 6 的例子，當找完{1,4,7,8,11}這個結黨之後，真正關鍵的區域應該是{2,5,9}這個結黨的區域，可是，之前的方法，卻可能會找到{3,6,9}這個區域，這樣的結果，就會導致 ***IRoute*** 5 無法被分配到軌道上。

因此，我們將上述的以區域為基礎的軌道分配方法，做一些修正，以便於增

加軌道的利用率。我們希望在不移除所有 ***IRoute*** 重疊圖所找出來的結黨中的點和其伴隨的邊的情況之下，將圖上的結黨由大到小依序的找出來(find all clique of graph)，這是 np 複雜度的問題。所以，我們提出了一個啟發式的演算法，下圖 10(a)~(h)的例子，便是我們所提出的方法之說明。

首先如圖 10(a)，我們先將 ***IRoute*** 重疊圖中的最大結黨找出來，並且將相對應的 ***IRoute*** 利用[11]中提到的，針對解 MWHP 的啟發式演算法，來做軌道分配的動作。

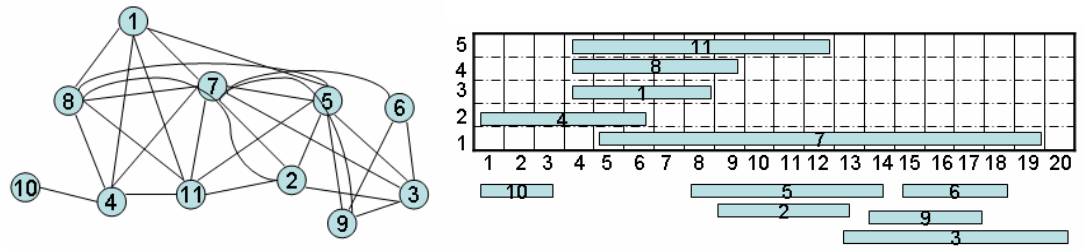


Figure 10(a) the left figure is the OLG of Fig(6). And the right figure is the TA result with the maximum clique {1,4,7,8,11} of OLG and we assign ***IRoutes*** by applying MWHP heuristic.

在找下一個結黨之前，我們要對水平重疊圖做一些處理，但是，並不是和之前的方法一樣，將所有找到最大結黨中的點跟相伴隨的邊都移除掉。取而代之的是，檢驗之前找到的結黨中的點，若是點的度(degree)等於結黨中除了本身之外的點數的話，我們就將這個點標記起來，並且，將這個點伴隨的邊都移除掉。

這個方法有個例外，就是當結黨中的點的度都不等於結黨中除了本身之外的點數，我們就無法標記任何的點，如圖 10(a)中的結黨{1,4,7,8,11}，因為結黨的大小是 5，因此，結黨中的每個點，除了本身之外的點數都是 4，但是，點 1 的度為 5，點 4 的度為 5，點 7 的度是 9，點 8 的度是 6，點 11 的度則是 7。所以，這個結黨中，便沒有任何的點可以被標記起來。

所以，我們就增加了一個規則來處理這個例外。當結黨中沒有任何的點可以被標記的時候，我們就依序假標記結黨中的每個點，並找出當這個點被標記，而且，這個點相伴隨的邊被移除的時候，***IRoute*** 重疊圖所能找出的最大結黨大小為何，我們比較每個找出來的結黨大小之後，就將標記之後，能找到最大結黨的那個點，真的標記起來。如圖 10(a)的例子，當標記點 1 的時候，我們找到的最大結黨是{2,5,7,8,11}，標記點 4 時，找到的最大結黨是{1,5,7,8,11}，標記點 7 時，最大結黨是{1,5,8,11}，標記點 8 時，所找到的最大結黨是{2,5,7,9,11}，最後，標記點 11 時，我們找到的最大結黨是{3,6,7,9}。我們將點 1 標記起來，並移除 ***IRoute*** 重疊圖中點 1 相伴隨的邊。結果如圖 10(b)。

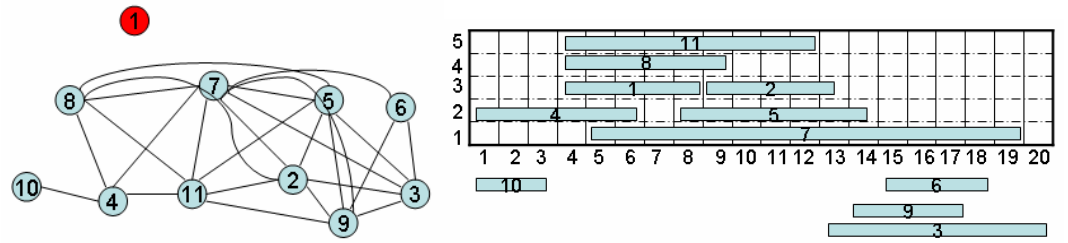


Figure 10(b) the left figure shows that we marked vertex 1 and remove its incident edges. And we find the maximum clique of ***IRoute*** OLG {2,5,7,8,11}. The right figure is the TA result after we assign ***IRoute*** 2, 5.

在將 ***IRoute*** 2,5 分配到軌道上後，我們發現又無法標記 ***IRoute*** 重疊圖中的任何點，因此，我們又依序假標記{2,5,7,8,11}中的點，我們發現當標記點 8 時，可以找到最大結黨{2,5,7,11}。因此結果如圖 10(c)。由於，***IRoute*** 2,5,7,11 我們都已做過軌道分配，因此，我們再繼續做標記點的動作。

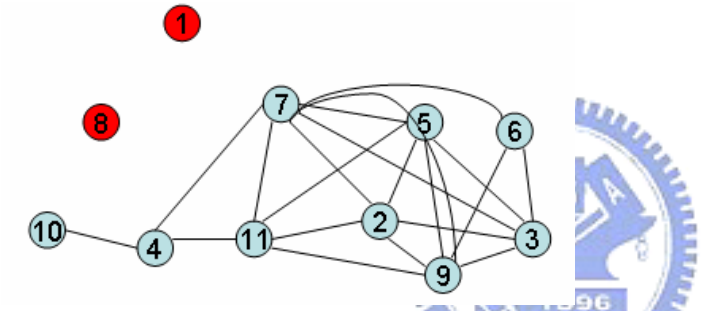


Figure 10(c) the left figure shows that we marked vertex 8 and remove its incident edges. And we find the maximum clique of ***IRoute*** OLG {2,5,7,11}.

當標記點 11 時，我們可以找到最大結黨{2,3,5,9}。結果如圖 10(d)。

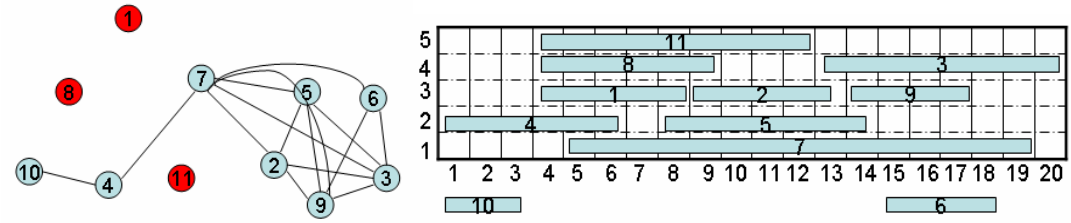


Figure 10(d) we marked vertex 11. And then we find the maximum clique is {2,3,5,9}。The right figure is the TA result after we assign ***IRoute*** 3, 9.

當標記點 2 時，可以找到最大結黨{3,6,7,9}。結果如圖 10(e)。

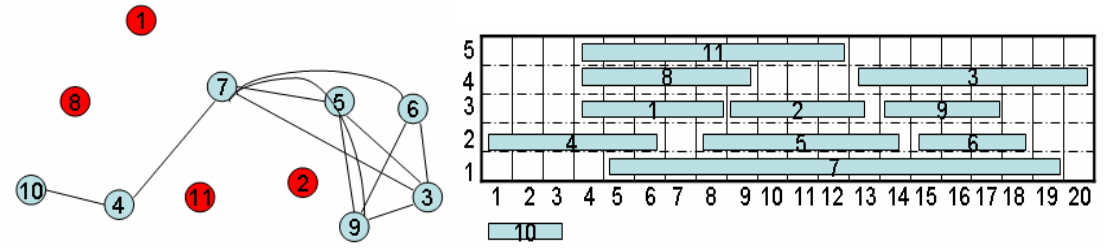


Figure 10(e) the left figure shows that we marked vertex 2 and remove its incident edges. And we find the maximum clique of ***IRoute*** OLG {3,6,7,9}. The right figure is the TA result after we assign ***IRoute*** 6 .

由於這時點 6 的度等於結黨大小減一。因此，我們將點 6 標記起來。結果如圖 10(f)。

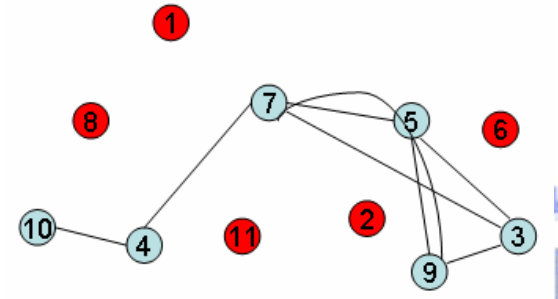


Figure 10(f) we marked vertex 6. And we find the maximum clique of ***IRoute*** OLG {3,5,7,9}

由於 ***IRoute*** 3,5,7,9 都已做過軌道分配。因此，我們繼續標記點 3,5,9。結果如下圖 10(g)。

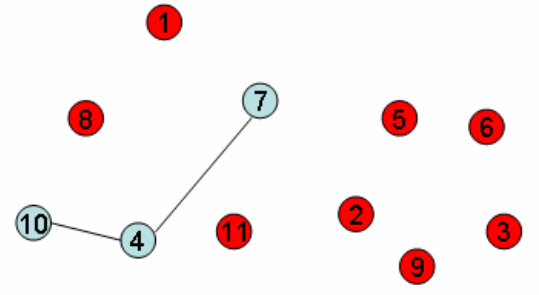


Figure 10(g) we marked vertex 3,5,9. And then we find the maximum clique is {4,7}。

因為 ***IRoute*** 4,7 已經做過軌道分配，所以，我們標記點 7，並且，繼續下一個步驟。如圖 10(h)，最後，我們發現所有的 ***IRoute*** 都已經被分配到軌道上。

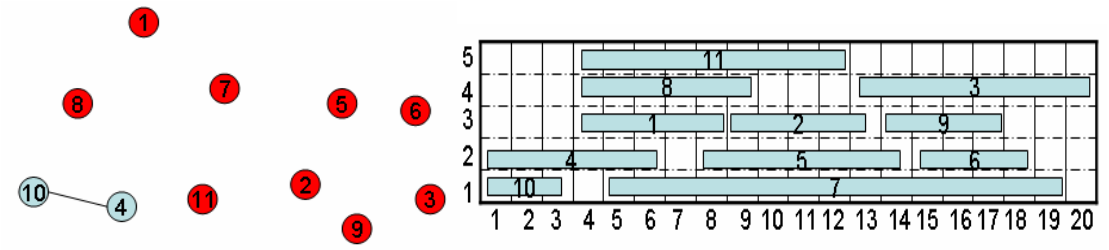


Figure 10(h) The final TA result.

我們這個找結黨的啟發式方法，和之前找結黨的方法不太一樣，我們找到的結黨順序，主要是連續密集的結黨排序，因為，我們並沒有將所找到的結黨中，所有相對應的點和邊都移除掉。

透過這個找結黨的方法，相較於之前的區域軌道分配方式，因為，我們是處理真正密度較高的區域，所以，我們能夠產生比較好軌道利用率的軌道分配結果。

3.2.2 串擾上限軌道分配演算法(RLC-bounded TA)

既然，我們選擇以區域為基礎的軌道分配方式，那麼接下來，就開始描述我們關於處理串擾上限軌道分配問題的啟發式演算法。

我們的演算法分為兩個步驟。第一個步驟，利用了我們在 3.2.1 節中提到的啟發式找結黨的演算法。我們將 ***IRoute*** 重疊圖中所找到的結黨，由大到小依序來做處理。也就是從嵌板裡，***IRoute*** 密度最高的區域，開始做軌道分配的動作。我們將結黨相對應的 ***IRoute*** 做軌道分配的動作，產生了一個軌道分配的初步解。第二個步驟則是，當我們針對每個圖中的結黨相對應的 ***IRoute*** 產生軌道分配的初步解後，我們會去判斷。如果，這個結黨中已經分配好的 ***IRoute*** 有產生串擾效應的話，我們就利用類似模擬鍛鍊(SA)的一種方法，也就是限制搜尋的方式，來進一步減少電感串擾效應的值。下圖(11)是我們的串擾上限軌道分配演算法。

1.**Algorithm**:RLC-bounded TA
2.**Input**: A global routing result for a panel
and nets' sensitive information.
3.**Output**: a track assigned panel.
4.**begin**
5.build the graph model
6. build the ***IRoute*** OLG;
7. build the bipartite assignment graph;
8.For each clique in ***IRoute*** OLG in the decreasing order of clique size
9. if there is ***IRoute*** unassigned in the clique
10. Initial assignment( );
11. if inductance coupling occurred
12. apply Tabu search to the initial assignment;
13. if unassigned ***IRoute*** exists, try to check if there is any suitable track to assign it;
14.**end**

Figure 11. The RLC-bounded TA algorithm

(1)軌道分配的初步解

首先，我們要介紹串擾上限軌道演算法的第一個步驟，也就是將結黨從大到小依序做軌道分配的處理。我們從 ***IRoute*** 重疊圖中找結黨的方法，是採用我們的啟發式演算法，也就是，不將找到的結黨，所相對應到的點和其伴隨的邊，從 ***IRoute*** 重疊圖中全部移除。利用這個啟發式的找結黨的方法，跟之前的方式不同。除了可以改善軌道利用率之外，我們還可以考量到不同區域間的串擾效應。

當我們找到一個結黨後，若結黨中的 ***IRoute*** 還有沒被分配到軌道上的，我們就進行演算法中的第一個步驟，也就是產生軌道分配的初步解。

我們處理一個結黨中的 ***IRoute*** 的方式，大致上，還是依循著從最關鍵的區域開始做軌道分配的動作。由於，串擾上限問題的特性，只有彼此有影響且繞線位置重疊的訊號線，才會產生串擾效應。所以，當我們找到一個結黨之後，就利用之前所輸入的訊號線互相影響關係，來建構影響圖。在影響圖中的點，一樣代表的是 ***IRoute***，而點與點之間有邊相連的話，則表示兩個 ***IRoute*** 的訊號，彼此互相有影響的關係。我們建構的影響圖中，可能像 ***IRoute*** 重疊圖一樣，有許多的結黨存在，我們一樣是從影響圖中的結黨由大到小依序做軌道分配的動作。

當找到影響圖中的一個結黨時，由於，結黨可能包含很多的 ***IRoute***，因此，我們要決定 ***IRoute*** 分配的優先序。我們以 ***IRoute*** 在 ***IRoute*** 重疊圖中的度由大到小依序做軌道分配的處理[17]，若是 ***IRoute*** 在 ***IRoute*** 重疊圖中的度一樣大的時候，我們就以長度較長為優先。會這麼決定優先序的原因，是因為我們想從最關鍵的 ***IRoute*** 開始處理起。

最後，在影響圖中，可能有一些獨立的點，也就是在這個 ***IRoute*** 重疊圖的結黨中，它並沒有跟其他的 ***IRoute*** 有訊號影響的關係，針對這些 ***IRoute*** 我們就利用最左邊端點演算法，來做軌道分配的運作。

當我們決定了 ***IRoute*** 做軌道分配的優先序後，我們就將 ***IRoute*** 依序地分配到第一個可以配置，並且不會造成衝突的軌道上，也就是達到彼此有影響關係的 ***IRoute***，不會被分配到相鄰的軌道上。

因此，透過這個軌道分配演算法第一步驟的時候，我們可以保證彼此有互相影響的 ***IRoute***，不會被配置到相鄰的軌道上，這樣可以幫助我們減少電容串擾效應的影響。

在圖 12(a)~(e)中，我們舉例來說明這個演算法的第一個步驟，也就是產生每個結黨的軌道分配初步解。

圖 12(a)中，標示了 ***IRoute***(1,6)，(2,6)，(3,5)分別有互相影響關係。

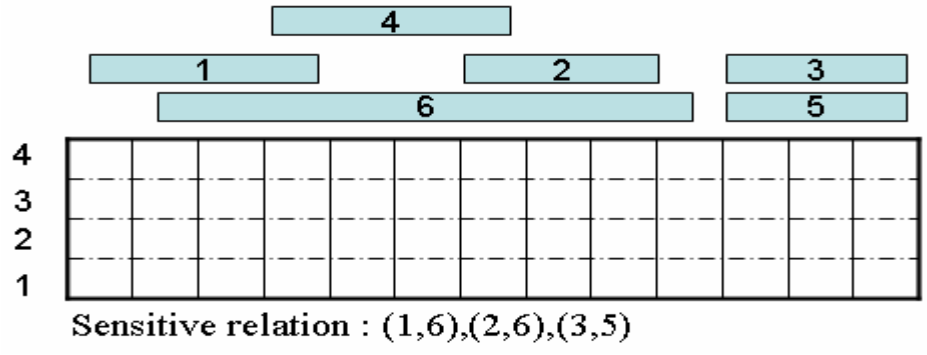


Figure 12(a) A RLC-bounded TA example.

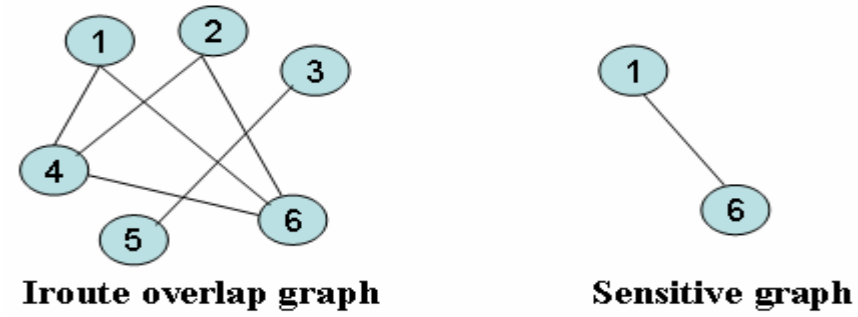


Figure 12(b) ***IRoute*** OLG and Sensitive graph.

圖 12(b)的左圖是根據圖 12(a)所建構的 ***IRoute*** 重疊圖。而右圖則是我們根據 ***IRoute*** 重疊圖中，找到的最大結黨{1,4,6}，對照輸入的訊號線影響關係後，所建構出來的影響圖。

我們得到軌道分配初步解的方法，是從 ***IRoute*** 重疊圖的最大結黨{1,4,6}開始做起，然後，再從其建構的影響圖中，也是將結黨由大到小，依序做軌道分配。影響圖中的結黨{1,6}是我們第一個要做軌道分配的結黨。而我們決定同一個結黨中，***IRoute*** 做軌道分配的優先序是依據其在 ***IRoute*** 重疊圖中的度來做決定，度越大的越先做軌道分配。其中，***IRoute***6 的度是 3，而 ***IRoute***1 的度是 2。因此，我們第一個處理的結黨，所包含的 ***IRoute*** 軌道分配優先序便是 6,1,4。軌道分配的結果如圖 12(c)所示。

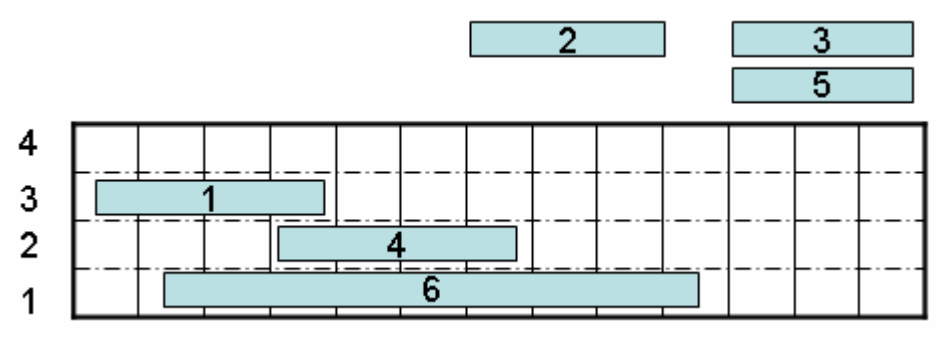


Figure 12(c) TA result after assigned the clique {1,4,6}.

圖 12(d)是我們做軌道分配的第二個結黨{2,4,6}的例子。由於，***IRoute***1 在 ***IRoute*** 重疊圖中的度等於結黨大小減一，因此，我們將點 1 標記起來，並且將其相伴隨的邊移除。接著，我們在更新後的圖中，找到了最大的結黨{2,4,6}。因為，***IRoute***4,6 已經做過軌道分配，因此，我們只要處理 ***IRoute***2 即可。將 ***IRoute***2 分配到第一個可以配置，並且不會造成衝突的軌道上，也就是軌道 3。

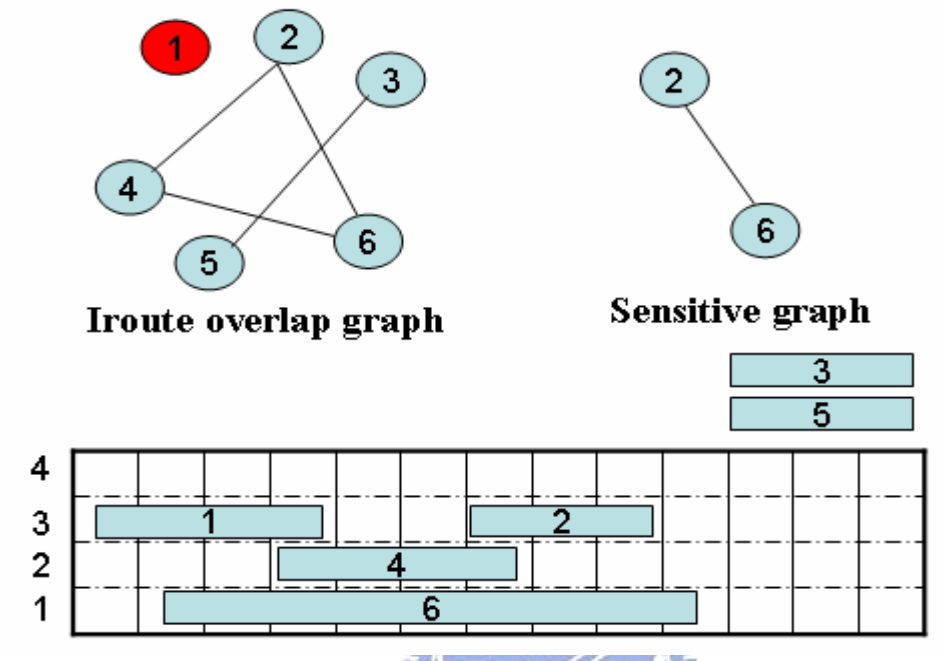


Figure 12(d) TA result after assigned the ***IRoute*** 2.

最後，我們將點 2,4,6 標記起來。從更新後的 ***IRoute*** 重疊圖中，找到最大結黨{3,5}。建構其相對應的影響圖後，依序做軌道分配的動作。結果如圖 12(e)。我們可以發現在第一步驟後，彼此互相有影響關係的 ***IRoute*** 都不會被分配到相鄰的軌道上。

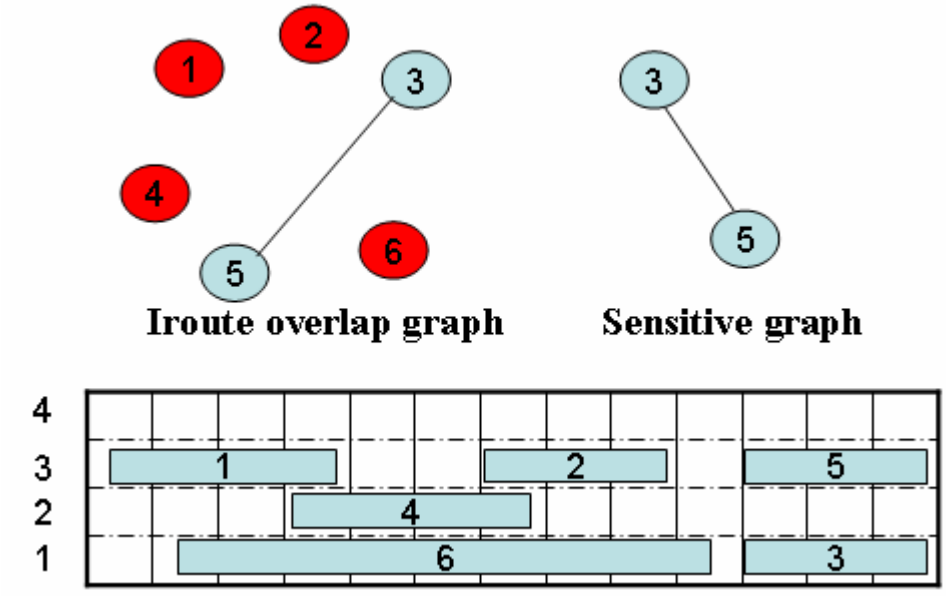


Figure 12(e) Final TA result after the initial assignment.

圖 13 是 RLC-bounded TA 中的第一步驟，找軌道分配初步解的演算法。

1.Initial assignment
2. build the sensitive graph
3. For each clique in sensitive graph in the decreasing order of clique size
4. ***IRoute***s are processed first in the decreasing order of their degree in ***IRoute*** OLG, and then in the decreasing order of their ***IRoute*** length if there is a degree tie.
5. each ***IRoute*** is assigned to the first assignable track without introducing capacitance coupling.
6.For each insensitive ***IRoute***
7. apply the LEA algorithm to the ***IRoute*** to the first assignable track without capacitance coupling.

Figure 13. Initial assignment in RLC-bounded TA.

(2)限制搜尋

限制搜尋這個方法現在已經被廣泛的使用來處理需要極多計算時間的問題。限制搜尋與模擬鍛鍊(SA)的方法類似，兩者執行效果差不多，但是，執行速度以限制搜尋較為快速。

限制搜尋主要的技巧在於將已經到達過的區域最小值(local minimum)紀錄下來，並且禁止再到訪達到這個區域最小值的情況。如此一來，就可以避免被這些區域最小值的情形限制住，所以，我們可以得到逼近全域最小值的結果。

我們在 RLC-bounded TA 的第二步驟，使用限制搜尋來減少電感串擾效應的影響。就如同前面所提到的，我們是以區域為基礎，將 ***IRoute*** 重疊圖中的結黨由大到小依序做軌道分配的動作。當這個結黨中的 ***IRoute*** 做完軌道分配後，若有電感串擾效應的產生。也就是說，這個結黨軌道分配後的結果中，有配置了相互影響的 ***IRoute***，我們就利用限制搜尋的方法，來進一步減少串擾效應的值。

我們在產生一個結黨的軌道分配初步解後，就利用隨機移動的方式，來改變軌道分配的結果，然後，透過成本函式的計算來逼近電感串擾效應最小值的結果。

其中隨機移動的方法，我們主要是利用結黨中 ***IRoute*** 位置的移動，或者是 ***IRoute*** 的位置交換，來產生一個新的軌道分配結果。然後，透過(4)式，這個成本函式的計算，來得到電感串擾最小化的軌道分配結果。

我們的成本函式中，其中S_{ij}表示了兩個***IRoute***是否彼此有互相影響的關係，若S_{ij}=1，則表示這兩個***IRoute***彼此有影響的關係。而N_{ij}和$NonN_{ij}$則是表示兩個IRoute是否有被分配到相鄰的軌道上，若是兩個IRoute有被分配到相鄰的軌道上，則N_{ij}=1，且$NonN_{ij}$=0。這個成本函式裡，還包含了兩個成本計算。第一個是電感串擾效應的估算方式，我們利用了[3]中，所提到的LSK模組來估算。第二個成本則是為了滿足串擾問題的要求，也就是彼此互相有影響的***IRoute***不能被分配到相鄰的軌道上。這個成本我們稱之為影響成本(Sensitive_cost)，因為，我們不希望彼此有影響的***IRoute***在隨機移動後，被配置在相鄰的軌道上，所以，這個成本我們設定為最大的整數(maximum integer)。我們透過限制搜尋所要得到的軌道分配結果，就是要總成本達到最小的軌道分配結果。

$$Total\_cost = \sum_{i}\sum_{j\neq i} S_{ij}\cdot(N_{ij}\cdot Sensitive\_cost + NonN_{ij}\cdot LSK_{ij}) \qquad (4)$$

總而言之，我們利用限制搜尋中的 ***IRoute*** 隨機移動或交換，和成本函式的計算，來逼近電感串擾效應的最小值。我們利用圖 14 的例子來做這個步驟的說明。首先，圖 14(a)和圖 12(c)相同，是第一個找到的最大結黨{1,4,6}，透過演算法第一個步驟，所產生的軌道分配結果。

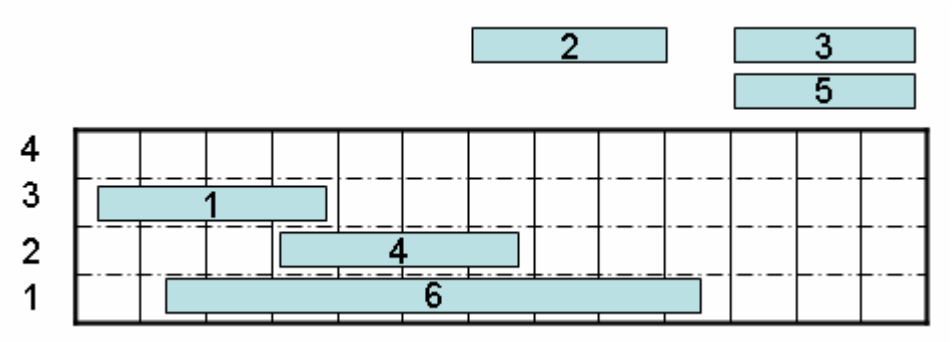


Figure 14(a) Initial assignment of the clique {1,4,6}.

在圖 14(a)中，由於 ***IRoute***1,6 彼此有影響的關係。我們利用限制搜尋的方法，來逼近電感串擾值的最小化。其中，電感串擾值的估算，是利用第二章中的(1),(2),(3)式來計算。

我們將整個嵌板視為一個區塊，在這個例子中，軌道數目有 4 個，分別是 1~4。因此，電線跟接地線(power ground wire)的位置我們分別設定為 0 和 5。當圖 14(a)中的初步解產生時，電感串擾值$K_{1,6}$為((1-0)/(3-0)+(5-3)/(5-1))*3=2.49。透過隨機移動，和限制搜尋運作後，我們發現當***IRoute***1 被分配到軌道 4 時，電感串擾值$K_{1,6}$為((1-0)/(4-0)+(5-4)/(5-1))*3=1.5。結果如圖 14(b)，就這個結黨{1,4,6}而言，電感串擾值有明顯的減少。

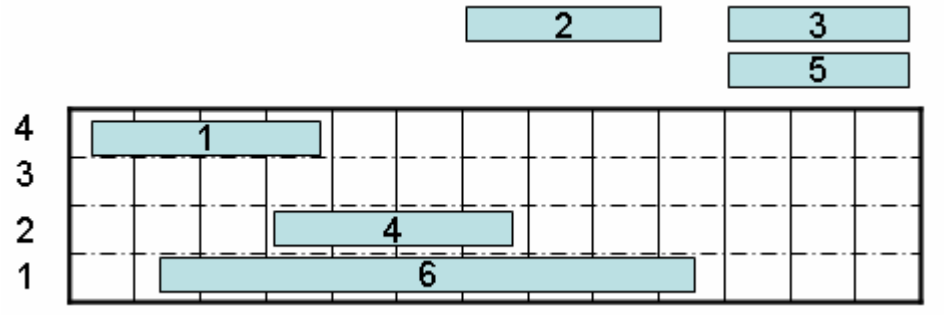


Figure 14(b). Move ***IRoute*** 1 from track 3 to 4 after Tabu search.

接著，我們依序做每個結黨的軌道分配動作。當每個結黨完成初步的軌道分配結果後，如果有產生電感串擾效應的話，我們就使用限制搜尋來減少產生的電感串擾效應值。如圖 14(c)~(f)所示。

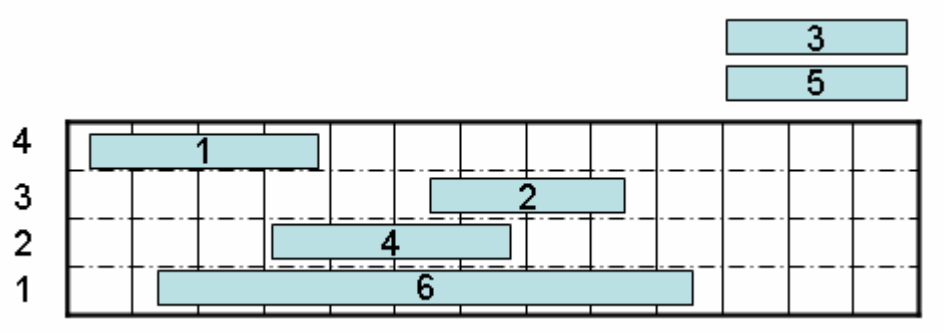


Figure 14(c) Initial assignment after assigned ***IRoute*** 2 of the clique {2,4,6}.

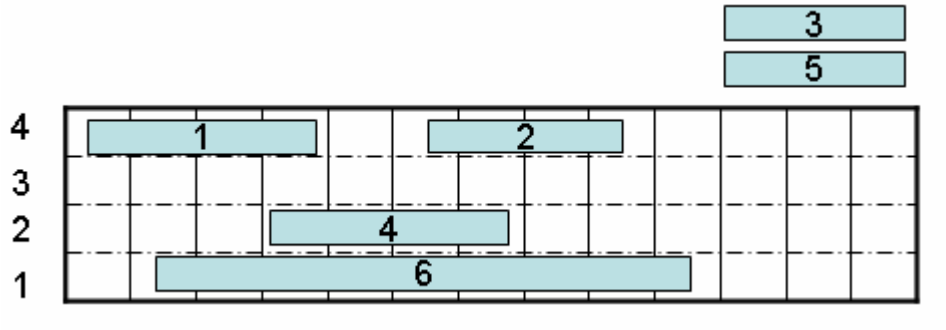


Figure 14(d). Move ***IRoute*** 2 from track 3 to 4 after Tabu search.

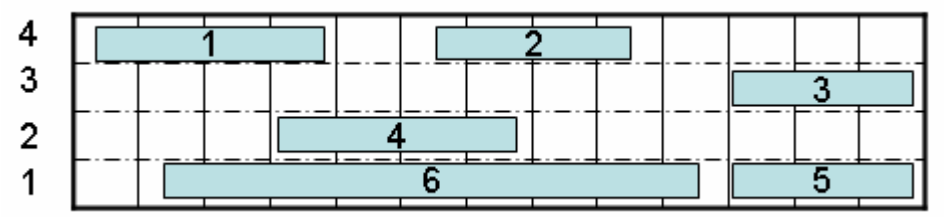


Figure 14(e) Initial assignment of the clique {3,5}.

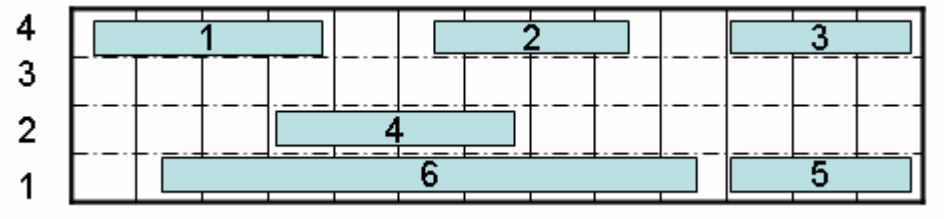


Figure 14(f). Move ***IRoute*** 3 from track 3 to 4 after Tabu search.

當完成 RLC-bounded TA 的兩個步驟之後，我們可以預期，每個嵌板中的電感串擾值，都會趨近最小化，而且，不會有彼此互相影響的 ***IRoute*** 被分配到相鄰的軌道上。

下圖 15 是引用[15]中的限制搜尋演算法。

```
1.Tabu search
2.Set the initial solution of a clique as initial solution TA^cur;
3.Set Tabu list H=empty; a=0; c=0;
4.while(a < Na){
5.      tmpcost = TA^cur_cost;
6.      b=0;
7.      while(b < Nb){
8.          TA^new = TA^cur;
9.          TA^new_cost = TA^cur_cost;
10.         Random_move (TA^new);
11.          if TA^new_cost is in H
12.                  c++;
13.                  if c < N_c, then continue;
14.                  else c = 0;
15.         if TA^new_cost < tmpcost, TA^tmp = TA^new;
16.            tmpcost = TA^new_cost;
17.            b++;}
18.     Insert TA^cur_cost into H;
19.     TA^cur = TA^tmp;
20.     if TA^cur_cost < TA^min_cost, then TA^min = TA^cur;
21.        a = 0;
22.     else a++;}
23.     Update H;
```

Figure 15. Tabu search in RLC-bounded TA.

其中，Na 是限制搜尋中，如果無法找到一個新的最佳解，所迭代的總次數。

Nb 是限制搜尋方法中，在一個迭代所找到的合法鄰解的總數。

Nc 是嘗試找到一個合法解的次數。

Na 和 Nb 是決定 Tabu search 的執行結果，和執行時間的重要參數，我們透過實驗，在執行效果和執行時間之間取得一個平衡點。

四. 實驗數據

4.1 以列為基礎軌道分配演算法的實驗數據

Case	No. of nets	Track size	Panel length (No. of GCells)
test1	12	5	20
test2	14	6	18
test3	9	5	13
test4	9	5	12
test5	12	5	14
test6	15	5	15
test7	12	5	23
test8	10	5	12
mcc1	1694	20	47
mcc2	7118	20	169

Table 1. THE STATISTICS OF TEST CASES.

我們的 RBTA 是以 C++程式語言實作的，而且這些測試資料都是在配備 Intel 2.4GHz cpu 和 768Mb 記憶體的電腦上執行。表一列出了我們從管道繞線相關的文章中取得的，8 個較小的測試資料，和兩個標準檢查電路 mcc1 及 mcc2。表 2 則列出了比較[11]中方法的測試結果。為了比較[11]中的方法，我們在相同的機器上，實作並執行他們的演算法。

表 2 的每個方法中，第一行的成本表示了軌道分配後的串擾效應總值，也就是相鄰 ***IRoute*** 的總重疊長度。[11]中的方法，並沒有辦法將 test1 中的 ***IRoute*** 完全分配到軌道上。但是，RBTA 能將 test1 中的 ***IRoute*** 完全分配到軌道上，所以，我們能得到較好的軌道利用率。而且，我們的 RBTA 相較於[11]中的方法能得到較少串擾效應的結果。概括來講，RBTA 能夠完全地將 ***IRoute*** 分配到軌道上，並且能夠達到和[11]中方法比較，平均減少 46.81%較佳的串擾效應結果。在和[11]中方法以及 HZTA 的比較之下，RBTA 能夠得到最佳的，減少串擾效應的結果。

case	Ho. [11]			HZTA[18]				RBTA			
name	Cost	I.C.	T (sec)	Cost	I.C.	T (sec)	R.R.	Cost	I.C.	T (sec)	R.R.
test1	94	1	< 0.01	94	0	< 0.01	******	82	0	< 0.01	******
test2	112	0	< 0.01	92	0	< 0.01	17.85%	82	0	< 0.01	26.78%
test3	60	0	< 0.01	40	0	< 0.01	33.33%	40	0	< 0.01	33.33%
test4	66	0	< 0.01	56	0	< 0.01	15.15%	50	0	< 0.01	24.24%
test5	74	0	< 0.01	66	0	< 0.01	10.81%	62	0	< 0.01	16.21%
test6	50	0	< 0.01	22	0	< 0.01	56.00%	22	0	< 0.01	56.00%
test7	106	0	< 0.01	80	0	< 0.01	24.52%	66	0	< 0.01	37.73%
test8	50	0	< 0.01	34	0	< 0.01	32.00%	34	0	< 0.01	32.00%
Average							**27.09%**				**32.33%**
mcc1	18088	0	1.382	834	0	9.953	95.38%	547	0	1.375	96.97%
mcc2	227842	0	21.453	5968	0	180.828	97.38%	4308	0	9.563	98.10%

I.C.: incomplete net; T(ms): runtime;

R.R.(reduction rate): (cost of [11] – cost of **HZTA[18]**/**RBTA**)/(cost of [11]).

Table 2. THE COMPARISONS FOR THREE CTA ALGORITHMS.

4.2 RLC 串擾上限軌道分配演算法的實驗數據

Circuits	Size(μ m)	#Layer	#Nets	#pins
S5378	4330x2370	3	1694	4734
S9234	4020x2230	3	1486	4185
S13207	6590x3640	3	3781	10562
S15850	7040x3880	3	4472	12566
S38417	11430x6180	3	11309	32210
S38584	12940x6710	3	14754	42589

Table 3. THE BENCHMARK CIRCUITS.

我們利用 C++程式語言實作了串擾上限導向軌道分配演算法，並且在配備 1GHz cpu 和 2GB 記憶體的 SUN Blade2000 工作站上執行我們的方法。表 3 列出了我們用來測試串擾上限導向的軌道分配演算法的標準測試資料。

Case name				Sensitivity rate : 25%						
	Panel	Net	Track	Initial assignment			Tabu search (10,5,5)			
				Cost	I.C.	T(s)	Cost	I.C.	T(s)	R.R.
S5378_h	10	1694	60	4486.74	0	0.19	1641.89	0	0.53	63.41%
S5378_v	19	1694	30	724.86	0	0.02	192.463	0	0.08	73.45%
S9234_h	10	1486	60	1929.74	0	0.05	525.489	0	0.15	72.77%
S9234_v	18	1486	30	652.705	0	0.01	180.224	0	0.05	72.39%
S13207_h	16	3781	60	9361.63	0	0.39	3334.99	0	1.23	64.38%
S13207_v	29	3781	30	2284.14	0	0.05	761.559	0	0.21	66.66%
S15850_h	17	4472	60	11859.9	0	0.48	4409.67	0	1.81	62.82%
S15850_v	31	4472	30	3097.2	0	0.05	1095.45	0	0.27	64.63%
S38417_h	27	11309	60	21817.6	0	0.75	7213.47	0	3.84	66.94%
S38417_v	51	11309	30	6082.66	0	0.11	1955.57	0	0.51	67.85%
S38584_h	22	14754	80	31732.8	0	1.97	12314	0	6.09	61.19%
S38584_v	43	14754	40	7873.24	0	0.16	2655.49	0	0.7	66.27%
Average										**66.9%**

I.C.: incomplete net; T(s): runtime;

R.R.: (Cost of Initial assignment – Cost of Tabu search) / (Cost of Initial assignment).

Table 4. THE COMPARISONS FOR Initial assignment AND Tabu search PHASES.

表 4~6 分別列出了在訊號線影響率(Sensitive rate)為 25%，33%，和 50%之下，當 RLC-bounded TA 完成 Initial assignment 步驟，和 Tabu search 步驟後的成本值，以及軌道分配的結果。

其中成本值(cost)是利用(4)式計算得來，而訊號線影響率我們則是利用隨機產生，25%的影響率，則是表示，我們針對每個訊號線隨機產生訊號線總數的 25%個訊號線，與其有互相影響的關係。例如:S5378 中，若是訊號線影響率為 25%，則我們對每個訊號線隨機產生 423 個訊號線，與之有影響關係。在 3.2.2 節中，我們有提到，限制搜尋方法的執行速度和執行效果，和參數 Na，Nb 有相當的關係，因此，我們透過實驗，找出 Na，Nb，Nc，分別為 10，5，5 時，有較佳的執行結果。

在我們的實驗結果中，所有的 ***IRoute*** 都有被分配到軌道上。而且，沒有任何彼此有影響關係的 ***IRoute*** 被分配到相鄰的軌道上。在訊號線影響率 25%，33%，和 50%之下，當我們執行 Tabu search 步驟後，對電感串擾效應減少率的平均分別為 66.9%，62.80%，和 55.44%，我們發現當訊號線影響率越小，電感串擾效應的減少率便越大。而且，Tabu search 對電感串擾的減少有相當明顯的幫助。

Case name				Sensitivity rate : 33%						
	Panel	Net	Track	Initial assignment			Tabu search (10,5,5)			
				Cost	I.C	T(s)	Cost	I.C	T(s)	R.R
S5378_h	10	1694	60	5934.69	0	0.19	2459.23	0	0.53	58.56%
S5378_v	19	1694	30	903.46	0	0.02	286.58	0	0.09	68.28%
S9234_h	10	1486	60	2538.58	0	0.04	792.086	0	0.17	68.80%
S9234_v	18	1486	30	828.633	0	0.01	260.543	0	0.04	68.56%
S13207_h	16	3781	60	12816.1	0	0.38	5086.83	0	1.25	60.31%
S13207_v	29	3781	30	3020.7	0	0.05	1100.88	0	0.23	63.56%
S15850_h	17	4472	60	15696	0	0.47	6355.73	0	1.88	59.51%
S15850_v	31	4472	30	3926.48	0	0.05	1528.57	0	0.28	61.07%
S38417_h	27	11309	60	28088	0	0.76	10340.5	0	3.8	63.19%
S38417_v	51	11309	30	8061.3	0	0.11	3032.16	0	0.56	62.39%
S38584_h	22	14754	80	41474.7	0	1.96	17582.7	0	6.22	57.61%
S38584_v	43	14754	40	10102.3	0	0.16	3860.14	0	0.76	61.79%
Average										**62.80%**

Table 5. THE COMPARISONS FOR Initial assignment AND Tabu search PHASES.

Case name				Sensitivity rate : 50%						
	Panel	Net	Track	Initial assignment			Tabu search (10,5,5)			
				Cost	I.C	T(s)	Cost	I.C	T(s)	R.R
S5378_h	10	1694	60	8888.06	0	0.19	4159.72	0	0.55	53.20%
S5378_v	19	1694	30	1350.01	0	0.02	547.061	0	0.08	59.48%
S9234_h	10	1486	60	3863.79	0	0.05	1511.11	0	0.17	60.89%
S9234_v	18	1486	30	1162.62	0	0.01	435.116	0	0.05	62.57%
S13207_h	16	3781	60	18850.5	0	0.39	8757.07	0	1.27	53.54%
S13207_v	29	3781	30	4429.4	0	0.05	1976.78	0	0.24	55.37%
S15850_h	17	4472	60	23246.1	0	0.48	10756.6	0	2.01	53.73%
S15850_v	31	4472	30	5548.93	0	0.05	2667.58	0	0.27	51.93%
S38417_h	27	11309	60	40828.8	0	0.77	17957.5	0	3.72	56.02%
S38417_v	51	11309	30	12013.9	0	0.11	5499.5	0	0.58	54.22%
S38584_h	22	14754	80	59155.4	0	1.97	28972.1	0	5.98	51.02%
S38584_v	43	14754	40	14540.5	0	0.16	6784.11	0	0.73	53.34%
Average										**55.44%**

Table 6. THE COMPARISONS FOR Initial assignment AND Tabu search PHASES.

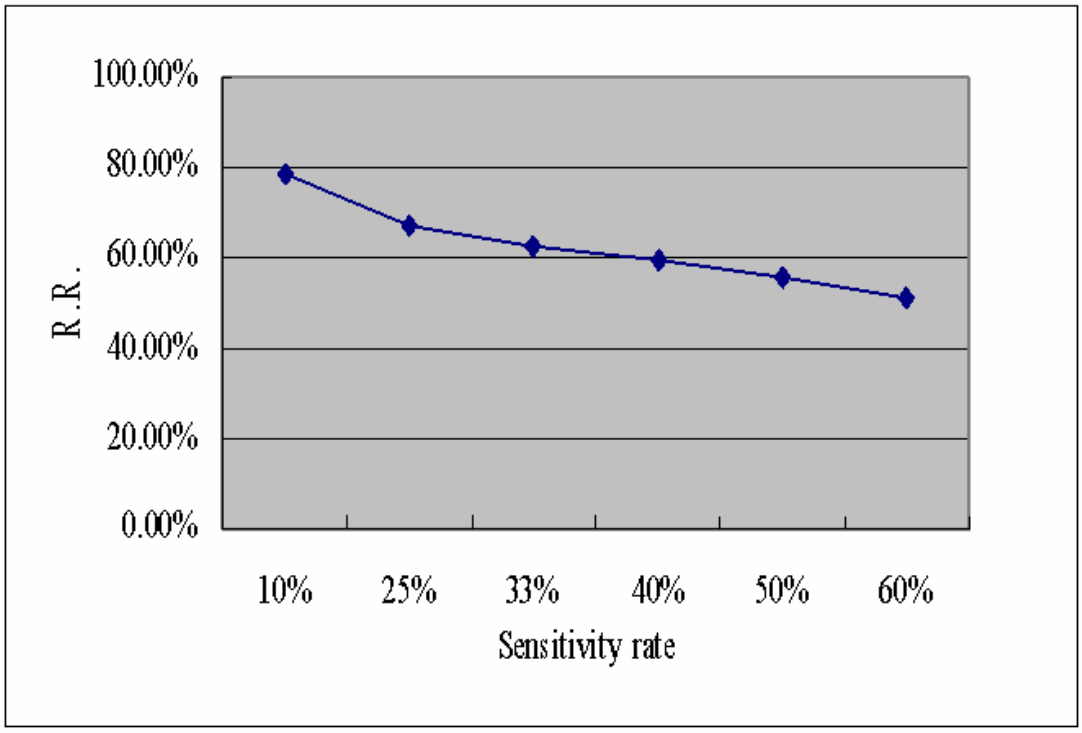


Figure 15. The relation of R.R. and Sensitivity rate.

五.結論

在這篇文章中，我們提出了兩個串擾導向的軌道分配演算法。首先提到的，是考量到電容串擾效應的以列為基礎的軌道分配演算法，這是一個軌道利用率和串擾導向的軌道分配演算法。RBTA 首先利用最左邊端點演算法來增加軌道的利用率，接著，我們提出了一個軌道重疊圖的新概念。藉著軌道重疊圖，我們就能將電容串擾導向的軌道分配問題，轉化為尋找最小成本漢米爾頓路徑(MWHP)的問題，也就是實現了針對串擾導向軌道分配問題，所產生的最佳軌道排序。在實驗數據中顯示出，相較於[11]中的方法，RBTA 可以減少平均 32.33%的電容串擾效應。而且，就如預期的，RBTA 也能夠得到較佳軌道利用率的軌道分配結果。

第二個串擾導向的軌道分配演算法，就是針對串擾上限問題，分別考量到電容和電感串擾效應的串擾上限導向演算法(RLC-bounded TA)。我們選擇了以區域為基礎的軌道分配方式來實作這個演算法。我們的演算法由兩個步驟所組成。首先是第一個步驟，利用了我們在 3.2.1 節中提到的啟發式找結黨的演算法。我們將 ***IRoute*** 重疊圖中所找到的結黨，由大到小依序來做處理。我們將結黨相對應的 ***IRoute*** 做軌道分配的動作，產生了一個軌道分配的初步解。當第一個步驟完成時，我們能夠確保彼此有互相影響關係的 ***IRoute*** 不會被分配到相鄰的軌道上。接著是第二個步驟，當我們針對每個圖中的結黨相對應的 ***IRoute*** 產生軌道分配的初步解後，我們就會去判斷。如果，這個結黨中已經分配好的 ***IRoute*** 有產生電感串擾效應的話，我們就用限制搜尋的方法，來進一步地減少電感串擾效應值。在我們的實驗中顯示，當訊號線影響率為 25%，33%，和 50%時，在和第一步驟完成後的串擾成本值比較，電感串擾值分別平均減少了 66.9%，62.80%，和 55.44%。因此，我們知道限制搜尋方法對電感串擾值有明顯的減少作用。而且，訊號線影響率越高的，電感串擾值減少率越低。

在未來，我們希望能夠針對串擾上限問題，研究以列為基礎的軌道分配方式。期望能夠得到一個更加提升軌道利用率，和降低電感串擾值的軌道分配結果。

參考文獻

[1] D. Sylvester et al, “Interconnect scaling : Signal integrity and performance in future high speed CMOS designs,” *Proc. VLSI Symposium on Technology*, 1998.

[2] A. B. Kahng and S. Muddu, “New Efficient Algorithm for Computing Effective Capacitance,” *Proceeding of International Symposium on Physical Design*, pp. 147-151, Apr. 1998.

[3] S.W. Tu, W. Z. Shen, Y. W. Chang and T. C. Chen, “On-Chip Inductance modeling for coplanar interconnect structure,” Proceeding of IEEE International Symposium on Circuit and System, Vol 3, pp. 787-790, 2002.

[4] Sherwani, N. A., Algorithms for VLSI physical design automation, 3rd ed, *Kluwer Academic Publishers*, 1999.

[5] Sadiq M Sait, Habib Youssef, VLSI PHYSICAL DESIGN AUTOMATION Theory and Practice, *World Scientific Publishing*, 1999.

[6] S. Batterywala, N. Shenoy, W. Nicholls and H. Zhou, “Track assignment : A Desirable Intermediate Step Between Global Routing and Detailed Routing,” *IEEE/ACM International Conference on Computer Aided Design*, pp. 59-66, Nov. 2002.

[7] H. Zhou and D. F. Wong, “Global Routing with Crosstalk Constraints,” *Design Automation Conference*, pp. 374-377, May 1998.

[8] Gao, T. and C. L. Liu, “Minimum Crosstalk Channel Routing,” *IEEE Trans Computer-Aided Design* **15**(5), 465-474, 1996.

[9] J. Xiong and L. He, “Full-Chip Routing Optimization With RLC Crosstalk Budgeting,” *IEEE Transactions on Computer Aided Design*, pp. 366-377, Mar. 2004.

[10] R. Kay and R. A. Rutenbar, “Wire Packing : A strong formulation of crosstalk-aware chip-level track/layer assignment with a efficient integer programming solution,” *International Symposium on Physical Design*, pp. 61-68, 2000.

[11] T. Y. Ho, Y. W. Chang, S. J. Chen and D. T. Lee, “A Fast Crosstalk- and Performance-Driven Multilevel Routing system,” *IEEE/ACM International Conference on Computer Aided Design*, pp. 382-387, Nov. 2003.

[12] Hashimoto, A. and J. Stevens, “Wire Routing by Optimizing Channel Assignment Within Large Aperture,” *Proceeding 8th Design Automation Workshop*, pp. 151-169, 1971.

[13] L. He and K. M. Lepak, “Simultaneous shield insertion and net ordering for capacitive and inductive coupling minimization,” *Proc. ACM ISPD*, pp. 56-61, San Diego, CA, USA, 2000.
[14] K. M. Lepak, M. Xu, J. Chen, and L. He, “Simultaneous shield insertion and net ordering for capacitive and inductive coupling minimization,” *ACM Trans. Design Automation of Electronic Syst.*, vol. 9, no. 3, pp. 290-309, 2004.
[15] L. Zhang, T. Jing, X. Long, J. Xu, J. Xiong, L. He, “Performance and RLC Crosstalk Driven Global Routing,” International Symposium on Circuits and Systems, Volume: 5, 23-26, pp. 65-68, May 2004.
[16] F. Glover, “Future paths for integer programming and links to artificial intelligence,” *Computers and Operations research*, **13**(5) : pp. 533-549, 1986.
[17] R. Yazdani and M. R. Zargham, “A Gridless Multilayer Channel Router,” *IEEE*,1990.
[18] Meng-Xin Jiang, Ying-Shu Luo, Yih-Lang Li, and Wen-Bin Chen, “Utilization-And Crosstalk-Driven Track Assignment”, in *The 16th VLSI Design/CAD Symposium*, Hua-Lien, Taiwan, Aug. 2005.