目錄

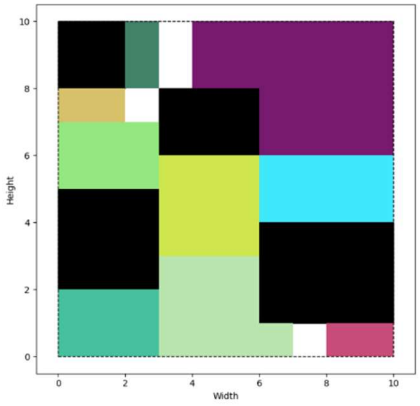
摘要

隨著技術的不斷進步，晶片設計的複雜性與電晶體的數目都在不斷增加。 為了因應晶片設計的複雜性與電晶體的數目的增加，階層式（hierarchical）設計和矽智產（IP）模組已經被廣泛使用。平面規劃（floorplanning）在這些設計方法中扮演著非常關鍵的角色。它不僅能在早期階段提供反饋，幫助評估系統架構，還能估計晶片面積以及預測由佈線引起的延遲和擁塞。因此，平面規劃一直是超大型積體電路設計中不可或缺的環節，甚至變得更加重要。

除了極小化晶片面積之外，積體電路平面規劃也常處理一些其他重要問題，例如處理固定輪廓（fixed-outline）限制及可變形模組（soft module）等。解決這些問題需要採用最佳化方法與技術，以最大程度地提高設計效率與設計質量。

由於晶片尺寸在設計初期已被確定，產生了固定輪廓限制，因此僅考慮最小化面積的平面規劃方法在現代的積體電路設計中並不完全適用。透過考慮固定輪廓的平面規劃方法，可以在晶片輪廓內適當地擺置模組，才能實現最大化晶片輪廓內的空間利用率及最小化連線長度等目標。因此，在現代的積體電路設計中，固定輪廓的平面規劃方法是不可或缺的。

此專題以隅腳編織法(Corner-stitching)作為儲存模組的資料結構，增加可幫助優化執行速度與輸出成果的演算法。並以此為基礎階段性地使用模擬退火法來設計固定輪廓的平面配置器，在面對各式的固定輪廓與模組形狀限制下，均可在允許的運算時間內將模組間導線長最佳化。



僅能以多邊形模組進行平面配置的例子，若將所有模組固定為矩形，將無法置入所有模組（黑色矩形為固定輪廓，白色為未使用區域，其它顏色多邊形為可調整模組）

隅腳編織法

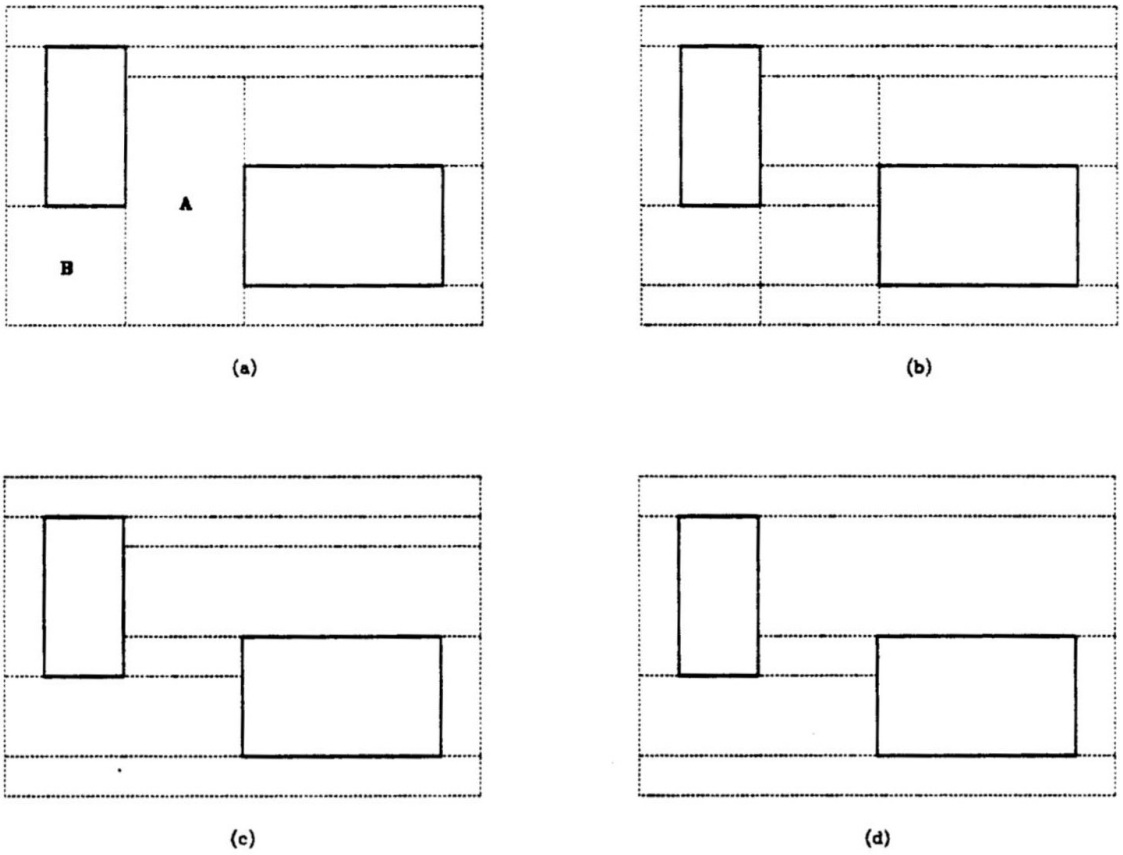
隅角編織法（Corner stitching）是一種處理即時超大型積體佈線的資料結構。此資料結構具有兩個特點，(1). 將晶片以多個矩形以稱作「針腳(corner stitch)」的指標連接。(2). 除了記錄矩形模組(Solid Tile)，也記錄了「空矩形」(Space Tile)。

空矩形(Space Tile)被表示為一個最大橫條(marital horizontal strips)，指其左右側必為一個實際矩形模組。而其最高/低垂直座標必然對應到一個矩形模組的最高或最低垂直座標(圖一)。在確定空矩形為最大橫條性質後，當有任兩個空矩形垂直相鄰，且左右側所連接的矩形模組相同，則可合併為較大空矩形。

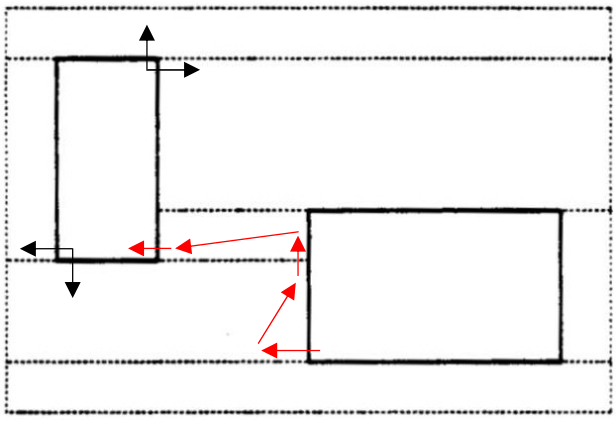
隅角編織法的特點在於其相關演算法的局部性 : 即，演算法的平均運行時間與受更動面 積中的矩形數量相關($O(Tiles\ in\ affected\ Area)$)。而基本操作函數具有下列幾項:

- Tile(coord, height, width) · 建構子，定義其左下座標為 coord，長寬為 height, width
- Point finding · 回傳給定座標坐落在的矩形
- Area search · 確認給定面積是否皆為空矩形，是則回傳真
- Neighbor finding · 利用針腳沿著給定矩形的邊界尋找相鄰矩形
- Create ·Area search 確認當前面積占用情況，若回傳真則在該範圍建立新的實心模組
- Delete · 將矩形變更為空矩形，並將此空矩形藉由分 割與合併以維持最大橫條之特性
- Split · 作用於空矩形，在置入前將空矩形做適當切割，確保空矩形在置入後仍維持其特性。
- Merge · 若相鄰矩形均有同樣的屬性(實心與空心)並可合併成較大矩形，則刪除其中一矩形並增 加另一矩形的長或寬。

由於多邊形可以被描述為多個矩形的組合，因此，即便此資料結構僅支援矩形模組的處理，也能視情形調整多邊形模組，缺點是並不能保證其高效。



虛線所包圍的空間為空矩形。旨在說明不會有空矩形其左/右側並排著另一個空矩形，如 A, B 必須被分割為更小的矩形，再進行合併[1]



黑色箭頭為左側模組的「針腳」分佈，所有的空/實心模組都有其針腳，紅色箭頭代表以右側模組為起點尋找左側模組的過程(pointFinding)。說明演算法的局部性

基本矩形模組置入與優化

本次程式將可變形模組的置入以矩形為主進行合法化與優化，並再以多邊形的延伸輔助導線調整，雖然階段性的優化並不能再最後取得最佳化的結果，但是能夠有效簡化程式的複雜度，提升效率。因此，如何初步以有效的方式置入模組影響了整體的程式效率，以下提供兩個置入的優化方式，分別降低置入的面積尋找時間，並優化結果。

尋找當下可利用最大空間(findUsableTile)

由於空矩形的紀錄特性，當特定區域的模組較為密集時，空矩形的高寬比會變得非常小，僅僅利用空矩形的面積決定是否足夠置入模組容易找不到合法的輸入，而不斷調整預設模組長寬並確認當下是否可置入又會導致置入演算法受晶片大小影響，為解決此問題，將當下可利用最大空間定義為「包含」給定座標所能得到不含任何模組的最大面積。

實際流程如下 ：

1. 固定給定座標(x_g, y_g)的水平座標，並由y_g為起點，利用當前空矩形的右上/左下座標資訊逐步向上/下尋找(pointFinding)包含水平座標x_g，垂直座標 Down(currentSpace)-1 or Up(currentSpace)+1，並且相連的空矩形，收入列表中，當前最大面積設為 0。
2. 紀錄最上側空矩形座標值y_t與最下側空矩形座標值y_d，尋找列表中所有右側座標值大於x_g中最小的水平座標值x_r，並尋找列表中所有左側座標值小於x_g中最大的水平座標值 x_l，當前可使用的最大面積為:

$$min(\{(x_r - x_l)(y_t - y_d), 2(x_r - x_l)^2,\ 2(y_t - y_d)^2\})$$

紀錄當前數值。

3. 移除當前最上/下側空矩形，回到第二步更新x_r, x_l, y_t, y_d，確認當前使用面積是否大於前者，若大於則更新紀錄數值，若非則依紀錄數值生成可利用最大矩形。

演算法雖然會與經過的空矩形的平方成正比(每次的移除都需要再遍歷剩餘空矩形)，不過能使置入的行為從晶片長寬獨立出來，維持了該資料結構演算法「局部性」的特點。

使導線長最佳化的座標選擇(findBestPos)

模組 M_a在置入於新的位置時，若使 M_i相關的導線長總和最小化，則其中心座標(x_a, y_a)在關係式 HPWL = $\sum w_i * (|x_a - x_i| + |y_a - y_i|)$ 中可以得到最小值。w_i代表與模組 M_a 與模組 M_i導線間的權重，(x_i, y_i)代表模組 Mi 當前座標，當下最佳座標點為考慮各導線權重之後 x, y 的中位數:

$$x_a = x'_i, \quad \sum_{j=0}^{i-1} w'_j \leq \frac{1}{2}\sum_{j=0}^{k} w'_j \leq \sum_{j=0}^{i} w'_j$$

x'代表對所有連線對應模組座標 x 排序後的列表。由於需要對導線列表進行排序，再進行運算，時間複雜度為 O(O(sort(k)) + k)，k 代表總導線數，程式使用 C++中 STL 所提供的 sort 函數。由於直接以此方程式所尋找的座標有可能為其它模組的中心座標，因此在挑選新的座標時會再加入隨機座標，關係式為 :

$$\text{newPos} = \sigma * \text{optimPos} + (1 - \sigma) * \text{randomPos}, \ 0 < \sigma < 1$$

當然即便加上隨機點進行平均也不能保證選到的座標為空矩形之中，可以再以矩形的「針腳」去找鄰近的可置入空間。或是再重新取得新的隨機點。

使用 force-directed method 也能得到近似的結果，且複雜度可以被簡化為 O(k)，不過實驗測試下最終的總導線長會略高於上述方式。兩種方式都能普遍取得較好的結果，但不能保證得到最佳的優化結果，存在以隨機座標選擇時結果反而更佳的情形。可視情況決定其中一項最佳座標尋找方式，若導線數量龐大，或許 force-directed method 的成效更佳。

二階段模擬退火法的搜尋

一般的平面規劃問題，其主要目的為最小化總導線長與設計時的整體使用面積。不過本次問題不需要考慮面積是否最小化，相反的，由於需要確保所有模組都能佔有足夠面積，因此，模擬退火法改以考慮晶片面積是否盡可能的被使用到。模擬退火法被分為兩個階段，首先針對位置與主要矩形長寬比，除了降低導線長，也用以獲得合法解。再將矩形模組延伸為多邊形模組，進一步降低導線長。

第一階段

第一階段的模擬退火法並沒有較簡易的鄰域結構，是以尋找未被模組占用的座標，並尋找當下可利用最大空間來做置入，本階段將合法性同時納入考量中，即退火過程會機率性地採納目前尚未合法的解，再以該狀態尋找導線長接近，但是更接近合法的擺置。

由於將合法性與導線長同時進行優化，因此需要將當前面積納入考量。接受更差解的機率定為:

$$random(0,1) < \exp\left(-\Delta * T_{init}/T_{curr}\right)$$

$$\Delta = weight_{HPWL} * \frac{HPWL_{new} - HPWL_{old}}{HPWL_{old}} + (1 - weight_{HPWL}) * (\frac{AREA_{old}}{AREA_{new}} - 1)$$

程式將$weight_{HPWL}$的初始數值定的較高，目的在於避免演算法過度注重合法性而未取得較好的優化解。不過若面對較難尋找合法解的情況，可能會使程式過度注重導線優化，反而產出不合法的擺置，為解決此問題，將退火時的溫度再分段為若干段落，當特定段落的優化結果過度傾向導線的優化，使得擺置一直落於特定不合法的情形時，重新執行該溫度區間的退火，並降低接下來的$weight_{HPWL}$。

第二階段

第二階段的退火則是依當前模組的位置進行延伸，填補擺置下所產生的空缺，由於填補空缺時，會使得模組之間更加密集，這附帶地使導線長能夠再進一步下降。而延伸的順序及方向的不同，會使得最終結果有些許差異。因此在鄰域結構的設計上，以 C++的 vector 儲存模組延伸順序，以及該順序下的延伸方向(水平，垂直，不更動)，有兩種移動方式，分別是，一、隨機調換一個模組在列表中的位置，二、隨機改變一個模組的延伸方向。此階段的合法解已經產生，因此，不對模組的面積進行考量。

M1	M2	M3	M4	M5	…
V	N	H	V	N	…

延伸方式如下:

1. 利用 findBestPos 以及水平/垂直方位決定延伸方向
2. 若為水平方向，利用隅腳編織法中的 findNeighbor 對當前模組的實體矩形尋找特定方向的所有空矩形；如果是垂直方向延伸，利用 findUsableTile 找可置入空間。
3. 依據限制計算每個空矩形當下可置入的子模組高/寬(若為水平，則高等於空矩形的高)
4. 將計算得來的所有矩形置入，並更新

對於每個模組，若決定以特定方向進行延伸，其垂直方向的座標便不會改變(假設以水平方向延伸，模組的最高點座標與最低點座標不會被影響)。目的在於計算當下可延伸空間與避免潛在的不合法風險。

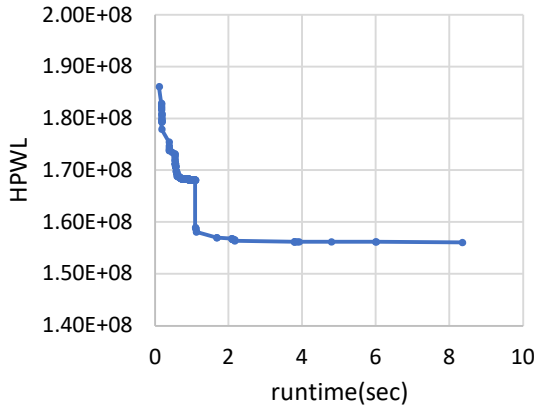
退火行程

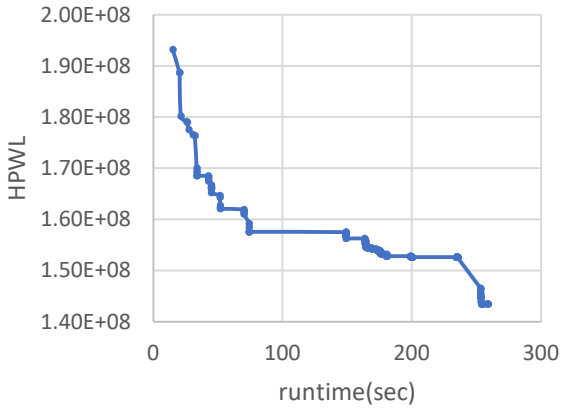
第一階段的初始溫度 Tinit 被設定為 60000，終止溫度 Tend 設為 0.01。對於 N 個模組，第一階段時，程式在每(3~1000)*N 次的嘗試後對當前溫度乘上係數ω，ω與當前溫度的關係為:

$$\begin{cases} \omega = 0.85, & T_{curr} \geq 2000 \\ \omega = 0.98, & T_{curr} < 2000 \end{cases}$$

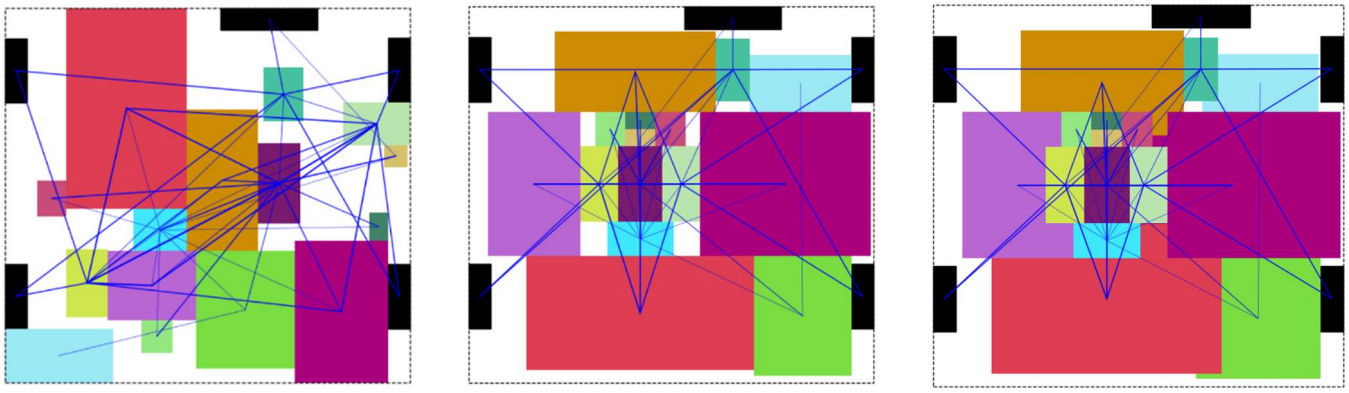
第二階段的退火初始溫度T_{init}被設定為 1000，終止溫度T_{end}設為 1，固定每 3*N 次的嘗試後

對當前溫度乘上係數 $\omega = 0.95$。因為該階段的收斂速度較快。因此無論在調整次數，冷卻係數，溫度間距的設定都小於第一階段。





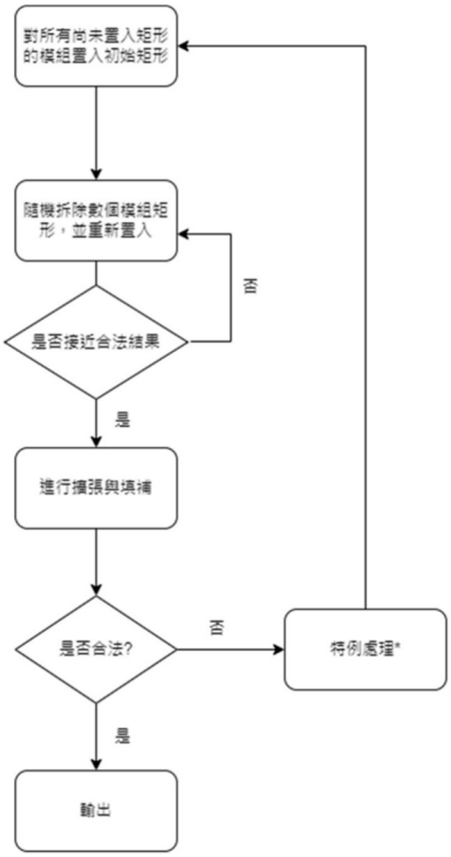
左/右圖為 Public case1 在每 3*N/1000*N 次調整後降低溫度後所產生的導線 v.s. 時間曲線。左圖曲線在約 1 秒與右圖曲線在約 200~300 秒之間有明顯的下降是源於第二階段的退火演算法。



由左圖至右圖為模組置入流程的可視化結果，左圖為第一次隨機置入的平面規劃，並非合法解，中間為進行第一階段退火的成果，此時所有模組均合法，且導線也有明顯的優化，右圖為經過第二階段退火的最終成果，原本中心的中空部分均被填補，導線也得到更進一步的優化。

多邊形的合法解尋找

本程式僅對存在模組均為矩形的合法解進行優化，因為僅有多邊形模組的情形隅腳編織法不能保證其高效。不過本程式仍然可以對此找到合法解。



程式上基於以下特性來進行:

{使大部分模組形狀接近矩形
初始置入均靠向晶片左下側

特性一使得模組形狀能盡可能的簡單，以加速執行速度(圖形越複雜，便需要越多矩形來描述)；特性二使得模組能夠在置入後不會過分分割空矩形，以降低產生合法解所需的迭代次數。

整體程式由以下四個函數進行 ：

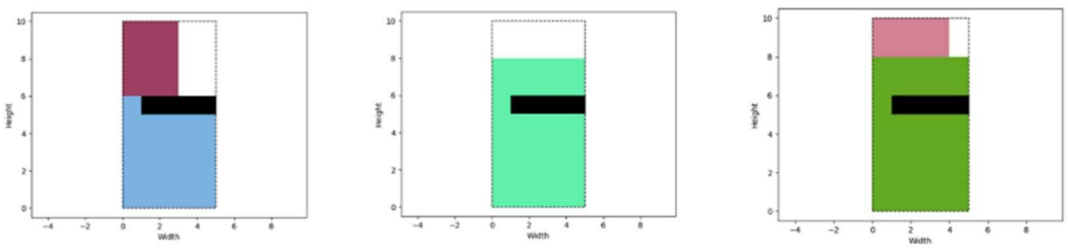
initInsert：對所有「選中模組」執行 generateRand， 計算當前分數

findNewPos：隨機拆除部分「選中模組」，並執行 generateRand 計算分數，若結果更佳則更新

patch：對於所有尚未合法的模組進行填補與擴張，直到全數合法，或是無法執行

handlingFail：對於尚未滿足面積要求的模組，拆除鄰近模 組，並嘗試填補。之後，拆除所有不合法模組與部分合法模組。將所有已拆除模組收入「選中模組」，進入下一次迴圈

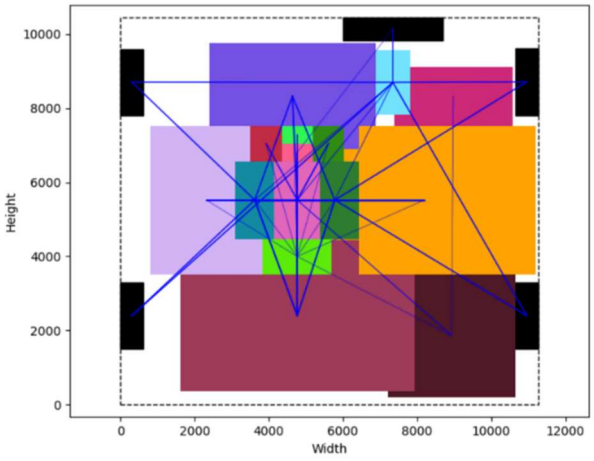
分數計算為 ：$Score = \min\{\frac{Module_i.currentArea}{Module_i.requiredArea}\}$，因為在生成合法解時，在意的為是否有任何一個模組無法完成置入。「選中模組」指的是本次迴圈將進行調整的模組集合。



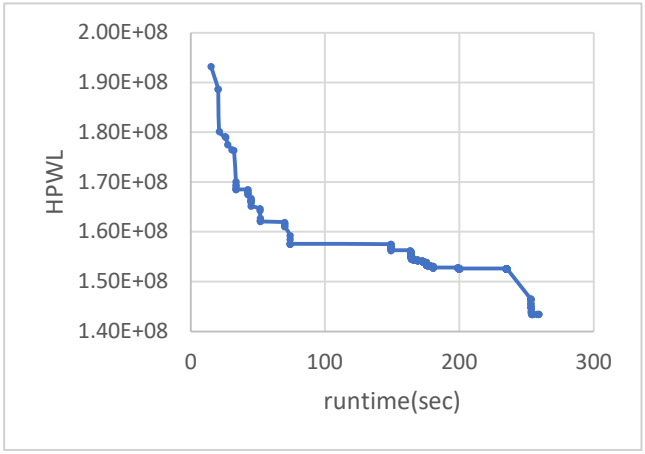
對於面積分別為 8、36 的模組置入，因為初期位置選擇的關係，下方模組無法完成合法建構。故需要 handlingFail 處理此情況。在下一迴圈時，下方的變形模組被視為固定模組，完成剩下模組的置入。

實驗結果

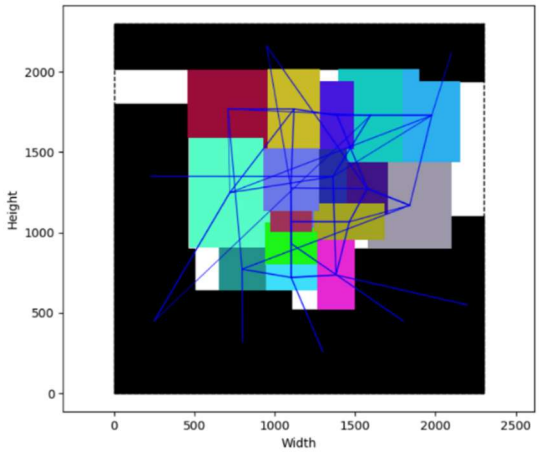
A. 可視化結果/HPWL 對時間的優化曲線（藍色線段代表模組間的連線，寬度越高代表權重越高）



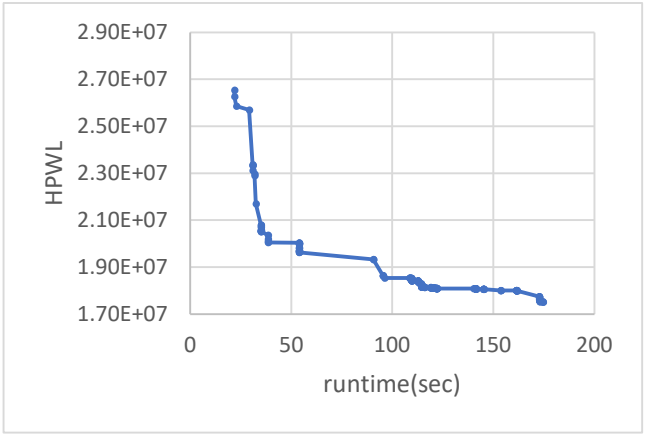
Public case 1



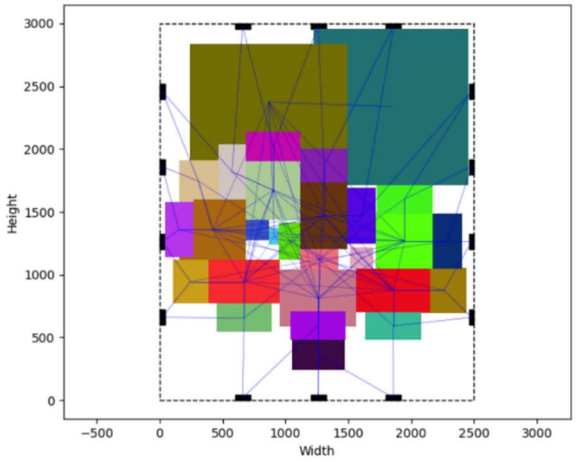
HPWL : 143469196.0



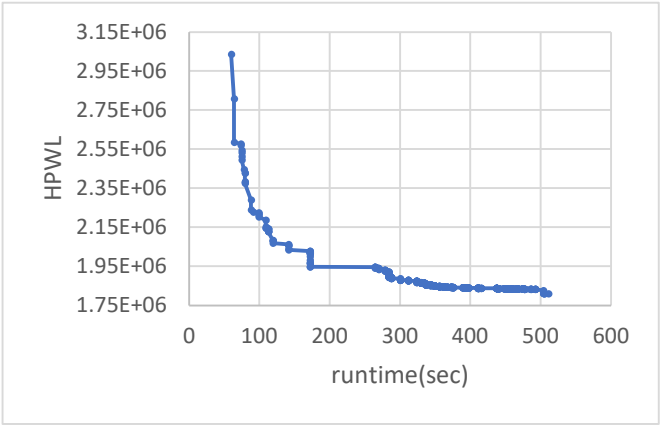
Public case 2



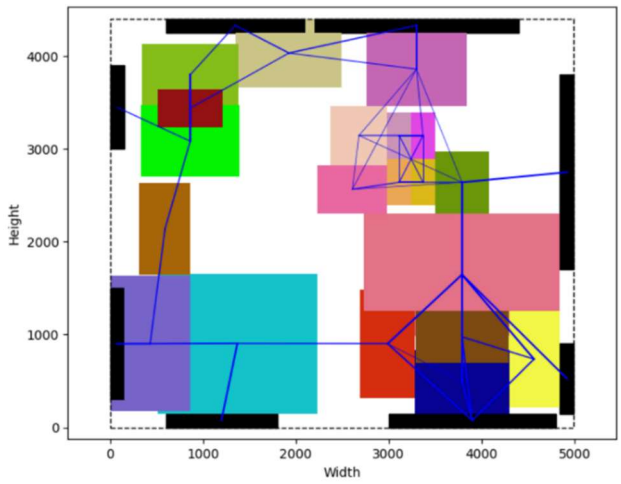
HPWL : 17506951.5



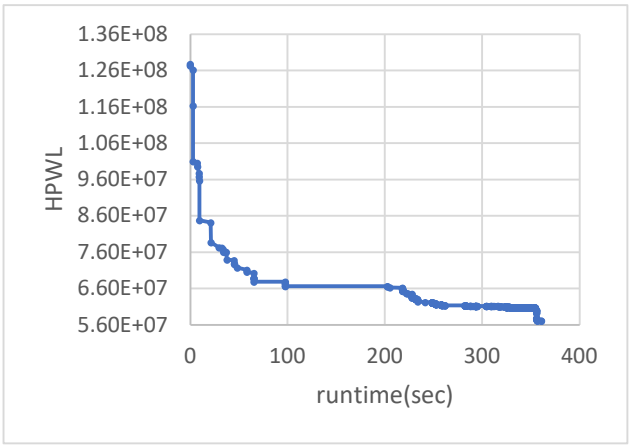
Public case 3



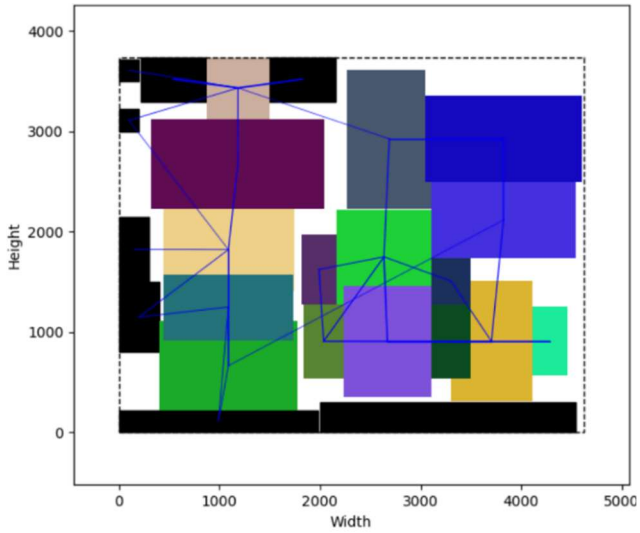
HPWL : 1808593.5



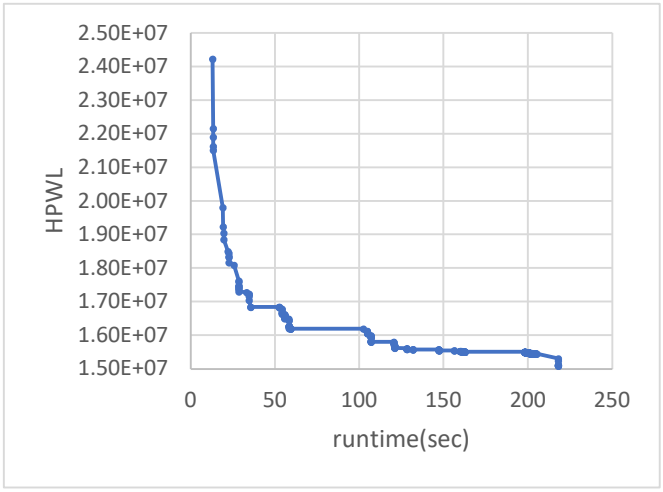
Public case 4



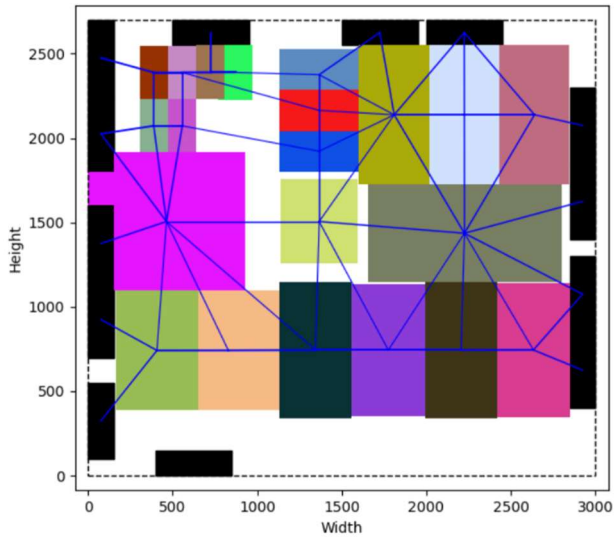
HPWL : 57128650.0



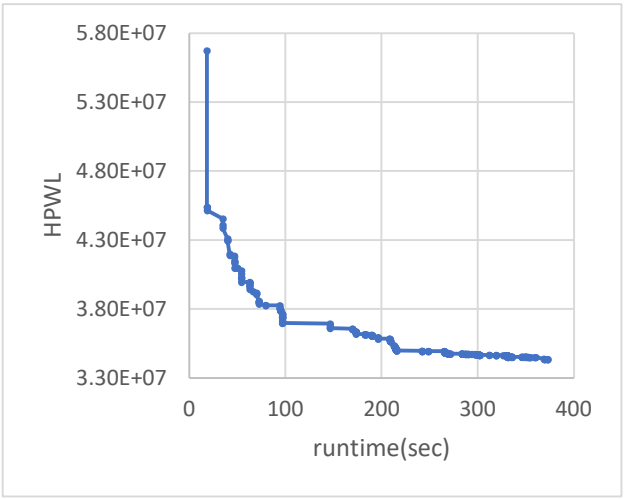
Public case 5



HPWL : 15087250.0



Public case 6



HPWL : 34344100.0

B. 競賽表現

Final Submission of Problem D (cadd0027)

Testcase	case01	case02	case03	case04	case05	case06	case07	case08	case09	case10
執行時間 (sec)	477.899	348.393	1029.149	753.206	450.552	705.498	440.066	1407.624	448.264	197.851
HPWL	143,469,196.0	17,506,951.5	1,808,593.5	57,128,650.0	15,087,250.0	34,344,100.0	371,460,550.0	94,659,900.0	81,583,750.0	17,538,150.0

參考文獻

[1] Ousterhout, John K. "Corner stitching: A data-structuring technique for VLSI layout tools." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 3.1 (1984): 87-100.

[2] S. N. Adya and I. L. Markov, "Fixed-outline floorplanning: enabling hierarchical design," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 11, no. 6, pp. 1120-1135, Dec. 2003, doi: 10.1109/TVLSI.2003.817546.

[3] C. C. N. Chu and E. F. Y. Young, "Nonrectangular shaping and sizing of soft modules for floorplan-design improvement," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 1, pp. 71-79, Jan. 2004, doi: 10.1109/TCAD.2003.819896.

[4] A. B. Kahng, “Classical floorplanning harmful?” in Proc. ACM Int. Symp. on Physical Design, pp. 207–213, April 2000.