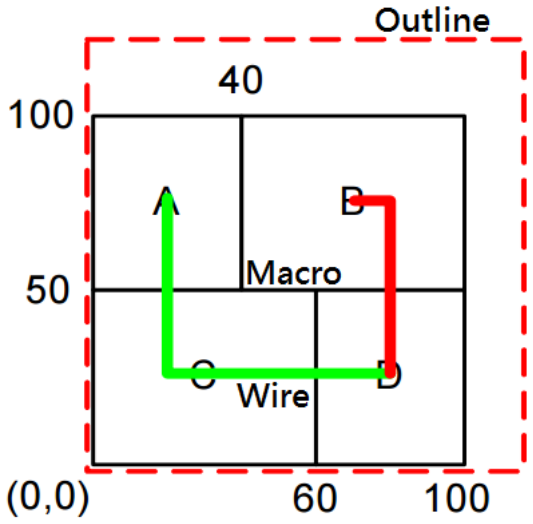
Introduction to EDA Final Project：
Fixed-outline Chip Floorplanning

Group Members

電機三 B02901024 鄭佳維
電機三 B02901026 莊佾霖

Problem Description:

This programming assignment asks you to write a fixed-outline chip floorplanner that can handle hard macros. Given a set of rectangular macros and a set of nets, the floorplanner places all macros within a rectangular chip without any overlaps. We assume that the lower-left corner of this chip is the origin (0,0), and no space (channel) is needed between two macros. The objective is to minimize the area of the chip bounding box and the total net wirelength.

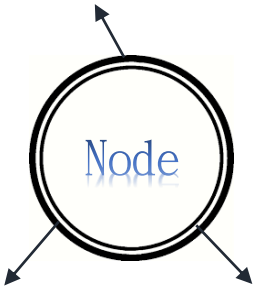


The objective for this problem is to minimize Cost = αA + (1-α)W, where A denotes the bounding-box area of the floorplan, W denotes the total net wirelength, and α, 0≦α≦1, is a user-defined ratio to balance the final area and wirelength. Note that a floorplan that cannot fit into the given outline is not accepted.

Data structure:

這次的 project 當中，我們使用的資料結構主要有兩個，第一個是 Node，也就是節點；而第二個是 floorplanning 的關鍵，由 Node 組成的 B* Tree

1. Node(構成 B* tree)

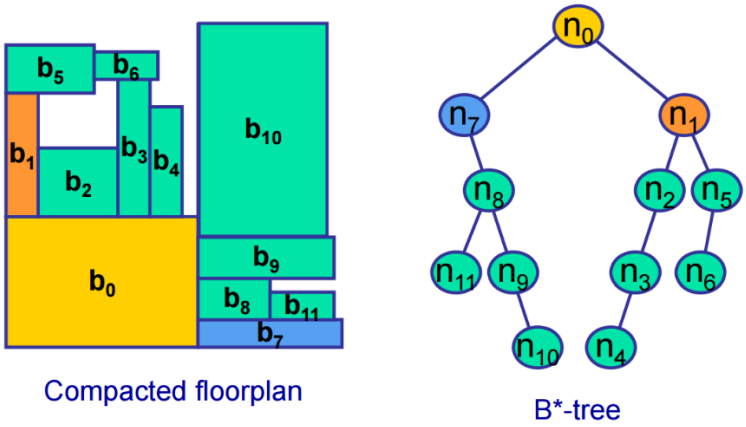


(1)Name and index of the node
(2)Pointer to parent
(3)Pointers to left and right subtrees
(4)Width and Height of corresponding macro
(5)Coordinates of macro position

2. B* Tree

Floorplanning 的主角，當然是 B* tree。

我們使用的是 Physical design 課程中，在講到 floorplanning 時，教授教我們的 B* tree 資料結構。B* tree 簡單來說，對於樹上的任何一個 Node，左子樹代表其右方的 Macro，而右子樹代表其上方的 Macro。這是一個簡潔簡單，卻又強而有力的結構！



而 B* tree 這個物件要存的內容，包括：

(1)Root of B* tree

(2)Cost of B* tree (Which is dependent of α)

(3)Skyline: 利用 vector 紀錄 B* tree 的 skyline，有助於算 y 座標

(4)Node list:建立一個能夠透過 node's index 找到該 node 的機制

此外，B* tree 也內建一個計算 cost 的含式。透過 Tree 的 pre-order traversal 可以計算面積，而透過 HPWL 可以計算 wirelength。

另外，我們為了計算 wirelength，也另外增設了兩個資料結構，分別為 terminal 以及 net：

3. Terminal

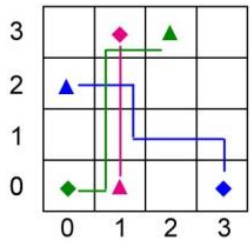
Terminal 此一資料結構的目的在於，記錄所有並非我們之前在第一個資料結構 node 當中記錄過的 Terminal。

(1)Name of the terminal

(2)Coordinates of the terminal

4. Net

Net 這一資料結構的目的在於，我們需要紀錄輸入進來所有包含在該 net 的 terminal，

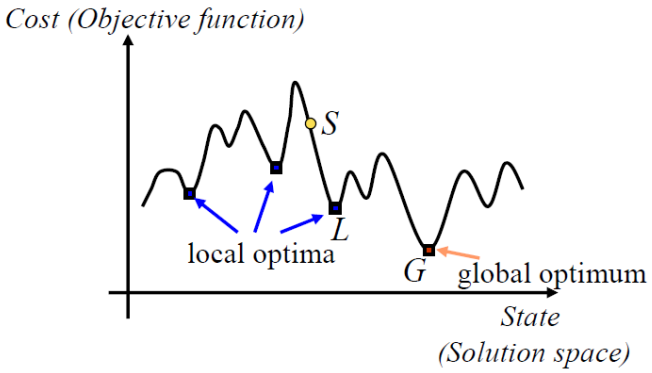


(1)Vector consists of nodes in net
(2)Vector consists of terminals in net

Algorithm:

演算法方面，我們使用的是 Simulated Annealing。

模擬退火，是 EDA 導論課程中，Physical design 一章所教授的重要演算法之一。基本上就是一個透過不斷的嘗試以及機率去尋找全局最佳解的過程。



而 Simulated Annealing 這一演算法，是由四個重要的成分所組成，而我們接下來將會一一介紹。

(1)Solution space：

Solution space 為所有的可行解所構成的集合。

因此，在我們這個 floorplanning 的 project 當中，Solution space 即是由 Input 得到的 node 所構成的，所有可能的 B* tree。

(2)Neighborhood structure：

Neighborhood Structure 即是和一棵 B* tree 鄰近的 B* tree 們；而鄰近的定義為，兩者之間相隔一個 operation。以下為我們所使用的 operation：

(I). Left rotate of binary tree
(II). Right rotate of binary tree
(III). Rotate of the macro
(IV). Delete and insert a macro (and its subtree)
(V). Swap position of two macros

(3)Cost function：

我們使用的 Cost function 當然和題目一樣，是 αA + (1-α)W。

然而，我們除了要盡可能的去最小化 area 跟 wirelength，題目有另一個相當重要的要求，那就是 Fixed-outline。因此，我們必須增加考慮目前的 B* tree 是否有確切地落在 outline 之中。

一開始，我們使用的條件，是 B* tree 所計算長寬值分別和 outline 所設定長寬值之差距，然而效果不彰。之後，我們參考 EDA 的課本，改使用了 Aspect ratio 的差，為 Outline 長寬比和 B* tree 長寬比的差值。

再來就是 normalization 的部分。如果不去做 normalize，那麼在我們這個題目的情況下，B* tree 之 cost 和 Aspect ratio 相差多個數量級，相加在一起完全沒有意義。我們的作法是一開始會先跑個 1000 筆左右的 B* tree，然後將 Cost 以及 Aspect ratio 的最大值算出，作為 Normalize 的基準。

(4)Annealing Schedule：

參考右方的 Simulated annealing 之 pseudocode：

(1)Let initial T=100

(2)Let k=100

(3)Let r=0.99

(4)Frozen~ $T=10^{-8}$

另外，我們取得 initial floorplan 的方式，則是將 input 的 macro 們隨機排序，之後就依序由左往右堆疊，若超出 outline 我們就往上堆一層。(當然，理論上 initial case 不應該影響 simulated annealing 之結果，但是從一個較好的 initial case 理論上對時間複雜度有助益。 <T,k,r 值僅是參考，在程式中可以調整>

Algorithm 10.1 Simulated Annealing Algorithm for Floorplanning

```
1:  Get an initial floorplan S;
2:  Get an initial temperature T > 0;
3:  while not "frozen" do
4:      for i = 1 to k do
5:          Perturb the floorplan to get a neighboring S' from S;
6:          ΔC = cost(S') − cost(S);
7:          if ΔC ≤ 0 then // downhill move
8:              S = S';
9:          else    // up-hill move
10:             S = S' with the probability e^(−ΔC/T);
11:     end if
12: end for
13:     T = rT; // reduce temperature
14: end while
15: return S;
```

Analysis:

(1)Time complexity

Time complexity 的話，我們基本上分兩塊討論，第一個是計算每個 B* tree 的 cost 時所花費時間。我們令 Node 總數 V 個，Terminal 總數 T 個，而 Net 總數 N 個，另外任意 B* tree 之中最大的 macro 寬度為 W。

(a).算 area：會利用 preorder-traversal 去算 B* tree 的 Area， 基本上每個 Node 要看一次其寬度大小的 skyline，因此整體複雜度就是 O(V*W)

(b).算 wirelength：基本上針對每個 Net 去掃每個 Net 當中的 terminal，統計 HWPL 需要的上下左右之極值。因此，需要的時間就是 Net*Terminal，而計算其 Worst Case 為 O(N(V+T))

因此，我們得到的 Count_cost 之 Time complexity 為 O(N(V+T)+VW)

第二個則是 Simulated annealing 的 Complexity：

Simulated annealing 下來，總共需要進行 i 次的 O(N(V+T)+VW)

其中，$i=\log_r 10^{-8}/T*k$

但在此 Case 之中，基本上 i 為 Constant，故總體複雜度為 O(N(V+T)+VW)

(2)Space complexity

Node 的複雜度：O(1)

B* tree 的複雜度：O(V+WW) <WW 為 skyline 的整體寬度>

Terminal 的複雜度：O(1)

Net 的複雜度：O(V+T)

(3)Pros and Cons of our design

*Pros：

1. 新增一些 Operation，使 neighborhood structure 更多樣
2. 一開始建立不只一顆 B* tree，等於是 parallel 去跑 SA
3. 有針對 Cost 和 aspect ratio 去進行 normalization
4. 使用 DEFINE 的機制，調整 annealing schedule 不會太複雜
5. 寫這份程式經過一番長時間的奮鬥，精神上值得嘉許

*Cons：

1. 不知為何，有些 Case 就是無法短時間就進入 Outline
   <這件事令我們相當挫折，我們已經嘗試了相當多可能性，再來就只能將 annealing schedule 調的很慢>
2. 可能有些函數使用了程式上較好寫但複雜度較高的方法
3. 不確定最後得到是否確實為 Global Optima，不過這應該是 Simulated Annealing 此一演算法的性質。

Performance:

我們使用 α=0.5，一次同時跑 5 棵 B*tree

而 T=100,k=100,r=0.99，對五個 Case 分別進行測試：

Benchmark	In outline?	Cost	Time(s)
Xerox	Yes	1.045 e7	220.183
Apte	Yes	2.424 e7	292.86
Hp	Yes	4.89 e6	235.945
Ami33	No	1.11 e6	105.239
Ami49	No	2.886 e7	445.657

<Ami33 和 Ami49 可能需要在一個比較慢的 annealing schedule 下才能成功進入 outline>

Thoughts:

當時選擇這份 Project 時，一來是因為我們兩個都對 Physical Design 這塊比較有興趣；另外，也是因為對張耀文老師在上課時所介紹的 B* tree 感到相當新奇—透過一個看起來很隨機的方法，Simulated Annealing，將 B* tree 使用隨機的方法去改動，居然就能得到最佳的解答！這令我們兩個感到相當驚奇，因此，我們就踏上了寫這個 fixed-outline floorplanning 之途：

一開始我們兩個的確有點掉以輕心，於是我們在 proposal 上給了自己一個相當緊湊的行程，想說兩個晚上以內應該就可以完全解決，畢竟我們已經討論出所有要使用的演算法。然而，事與願違，我們在努力兩個漫漫長夜，卻有需多 benchmark 連 outline 都進不去，真的相當挫折。於是我們不斷的 modify 程式，然後跑 SA，然後再 modify，再跑 SA…

報告的時候，我們將我們的內容講給老師聽，也得到了不少相當實用的 feedback，其中又以 normalization 為最。的確，我們在寫程式的過程中也有感覺到 normalize 是一個必須，但是我們卻始終無法找到最佳的 normalize 方式，於是報告完之後我們調整我們的作風，繼續努力，終於，情況獲得了些許改善。

期末考後又努力到現在，我們最後的結果，依然不慎完美。但是我們依然認為，這次的報告，我倆收穫頗豐。如果寫的是一個馬上就能攻克的程式，不也有點太沒意思了些？我們看著一次次 SA 的結果，優化的成功率慢慢增加，內心的雀躍感是不言而喻的—這也是這次 Project 我認為最重要的意義，大三了，我們也慢慢地脫離制式化的課程，以後所解決的問題，許多都是 open，都需要經過百般嘗試。這份練習，我們不但更熟悉了 floorplanning 一塊、精進了寫程式的技巧，更重要的是，我們在不斷嘗試中調整與學習，這個精神，大概才是我們最大的收穫！

References:

1. Slides of physical design in class

   Yao-Wen Chang

2. Electronic Design Automation: Synthesis, Verification, and Test~Ch10 Floorplanning

   Yao-Wen Chang, Tung-Chieh Chen

3. Simulated annealing - Wikipedia