Physical Design Report in PA1

R08921053 電機研一 梁峻瑋
r08921053@ntu.edu.tw

＆設計的資料結構與演算法

首先，我把整個程式作業的工作拆成了下列的函數來分工運作.

```
void buildPart();
void buildGain();
bool setMaxGainCell();
void updateGain();
void removeNode(Node* rmNode);
void addNode(Node* adNode);
void moveCell();
void backToBest();
void reportbList();
```

接下來，我將來說明使用到的資料結構等等.

問題: 遍歷所有 pin 的實作方式—超過線性時間的複雜度?

舉 buildPart()函數為例子. 在這個函數中，最重要的一點是: 如何走遍所有的 pin，在不同的 cell 和 net 組合上，決定 net 的 PartCount 數值. 很直觀的方法是，可以走遍所有的 cell，再在與每個 cell 相鄰的 net 上，去決定 net 的哪一個 PartCount 要+1.

然而，由於我們的資料結構限制，必須在過程中，在每個 cell 上，做 map 的動作得到連結的 net，即 _netArray 的使用. 因此，時間複雜度並非如同預期的是:

$$\text{Sigma}_{\{Cell\}}\ (O(\#pin)=O(\text{sigma}_{\{Cell\}}(\#pin))=O(P),$$

反而應該是

$$\text{Sigma}_{\{Cell\}}\ (O(\log(\#pin))=O(\text{sigma}_{\{Cell\}}(\log(\#pin)))>=O(P).$$

基於兩個原因，這點相當的重要:

(i) 由於本演算法的背景是線性時間(to P:=#pin)的演算法，所以這個操作的時間複雜度可能會超過 O(P)，到達非線性的複雜度，形成演算法瓶頸.

(ii) 這個操作將在下列的運作中出現超過十次以上，因此他在時間複雜度中的常數項將會相當大. 由於他至少是 O(P)以上的複雜度，故此演算法的領導係數會相當大，造成運算可能還是算的出結果，但會拖到數分鐘甚至數小時的長度.

改進的想法與點子

基本上，我認為在預設的資料結構當中，我們根本可以把 map 改成 unordered_map. 基本上，我們只有在一處使用到 map 的排序功能—在挑選 _maxGainCell 的時候，也就是 setMaxGainCell()函數. 但，在這裡我們可以用線性掃過 unordered_map，來找到最大 gain 的 cell，花費時間是

O(max pin number)=O(P)

而且，我們也只需要在每個回合 update Gain 之前，找一次 maxGainCell()就好. 整體看起來，多出的花費並不很高.

然而，這樣一來, find 和 erase 的時間複雜度都是 O(1)，常數時間. 如此一來, 所有遍歷 pin 的實作將都會簡化成 O(P)的時間複雜度. 儘管遍歷的次數很多，領導係數相當大的問題依舊沒解決，但至少在這個部分，我們能夠確定他是 O(P)的線性演算法.

此外，如果要再進行優化，尋找最大值的演算法也可以參考 Divide and Conquer 或是 quick sort 的方式來進行優化. 當然，花費的空間可能會更多就是了

最終的結果表格

	Cutsize	Cell Num	Net Num	PartA Num	PartB Num	Time
Example on Note	1	6	5	1	5	0.00289s
Input_0	20704	150750	166998	78118	72632	204.985s
Input_1	1450	3000	5000	1487	1513	0.04139s
Input_2	2575	7000	10000	3430	3570	0.15179s
Input_3	31124	66666	88888	30000	36666	39.3086s