# CS6135 HW2 Report

**108062537 魏聖修**

## (2)

In the directory src,

compile: make

execute: ./FM_Partitioner cell_file net_file output_name

example: ./FM_Partitioner ../testcases/p2-1.cells ../testcases/p2-1.nets ../output/p2-1.out

## (3)

| | Cut Size | Runtime(sec) |
|---|---|---|
| P2-1 | 14 | 0.01 |
| P2-2 | 198 | 0.06 |
| P2-3 | 5528 | 1.11 |
| P2-4 | 45495 | 2.13 |

## (4)

對我寫的程式來說，速度一開始很慢，時間主要花在要把每條 net 上連接到的 cells 在 vector_cell 內相應的 index 找出來。

EX: NET n1 { c12 c78 } 在 vc[0]和 vc[1]內就要記錄 index 0 和 1。

要跑三層迴圈比較花時間 (cell_count) * (net_count) * (# cell in this net) 這麼多次，我把這段時間算在 IO。

不過只要有這些足夠的資訊，之後再跑 FM 就跑得很快。

| | IO (sec) | Computation(sec) |
|---|---|---|
| P2-1 | 0.02 | 0 |
| P2-2 | 0.76 | 0.01 |
| P2-3 | 357.4 | 0.60 |
| P2-4 | 1107.31 | 1.16 |

不過經過一點調整後，速度進步很多。主要是新增一個 map
map[cell_name] = the index of the cell in vector_cell
有著這個 map 就不用跑三層迴圈了，剛剛最耗時的那部分變成只要做
(net_count) * (# cell in this net)次迴圈。

| | IO (sec) | Computation(sec) |
|---|---|---|
| P2-1 | 0 | 0 |
| P2-2 | 0.3 | 0.2 |
| P2-3 | 0.45 | 0.64 |
| P2-4 | 0.92 | 1.20 |

## (5)

### I.

基本上講義上提到的 initial gain 和 update gain 都是依照講義上的演算法實作的，所以大致相同。Bucket list 也是依照講義用 linked list 來做，並把 cell 都指到在 bucket 上相應的位置。
比較不同的部分是一開始 partition 成 A、B 兩邊，我的方法是先都擺到 A 直到擺到 cell size 總和之一半，剩下一半再擺到 B，這是我試過一開始 Cut size 會最小的方法。還有我會對全部的 cell 都做試著做 FM，除非 cell 換邊會導致不平衡才不做，這樣雖然要做比較多次，但其實不會花太多時間，而且可以得到較小的 cutsize。

### II.

我有做 bucket list，也是依照講義的圖示用 linked list 做的，並把 cell 指到在 bucket list 相應的位置。數量會依每個 case 不同，不過基本上都是建 2*Pmax + 1 條。然後每次都是從最大 gain 的往下拿，然後都是拿那一條的第一個 cell。

### III.

我的 FM 會把每個 cell 都跑過一次，所以會做完全部。然後我會從一開始就記錄目前為止最大的 partial sum、那是第 K 次 pass 和用 vector 紀錄 pass 的順序。這樣要 recover 就非常方便，只要把 vector 內第 K+1 個到最後一個 index 相應的 cell 做復原，就是在 A 的移回 B，在 B 的移回 A，而且也不需要做 bucket list 的操作，很簡單即可完成。

### IV.

和之前 Top5 的同學比較，我的 FM cutsize 結果普遍較差，最可能是因為我的 initial partition 並沒有用一個比較有效的分法，不然基本上 FM 是會把全部的 cell 都跑完了。不過在速度上我的程式算是非常快的，都比去年的快。

**V.**

對於 runtime 優化方面，我程式原本的 bottleneck 如上面**(4)**所說，一開始要跑 三層迴圈比對 cell 比較花時間，不過換個資料結構來儲存 data 就能夠改善這個問題。所以 runtime 進步非常多。此外，我有紀錄最大 partial sum，這樣就不必花時間計算兩次 cutsize（最初和最後），不用重新 refresh 所有的資訊，只要一開始算一次，最後再減 partial sum，即為最後的 cutsize，這樣也是省了點時間。

對於 cutsize 方面，我覺得由於後面的 FM，大家的作法應該都類似於講義上的，而且我也把全部的 cell 都跑完了。所以我覺得影響較大的因素應該是 initial partition 方面，不同的 initial partition 之後所運算的結果相差非常多。

例如在 case1 用我目前的方法 initial cutsize 是 90，final cutsize 是 14。我實驗若換成看哪邊 total size 小就把這個 cell 填去那邊的方法，initial cutsize 是 252 final cutsize 是 78，兩者間就有很大的差距，所以我覺得 initial partition 將影響很大，如果要一個較好的 initial partition，也許就要更深思熟慮一點的方法。

**VI.**

由於一開始並沒有把演算法了解的很透徹，所以邊做邊想，多做了很多不必要的 function，不過最後有把全部整合在一起。經過這次的作業算是比較了解 FM 在幹嘛了，果然要了解一個演算法就是要先實地做一次，而且把很久沒用的 linked list 拿出來複習了一下。遇到的問題就是一開始看不太懂講義，只有算 gain 那邊比較明確，其他部分都要自己先去想該怎麼做，然後再看講義才會知道講義到底在講甚麼。