HW2 Report

102062139 羅嘉諄

Q1 : How to compile and execute your program, and give an execution example.

--HOW TO COMPILE

In this directory, if you enter the command "make HW2", the executable file named "HW2" will be generated in the same directory.

If you want to delete the executable file, please enter the command "make clean".

--HOW TO RUN

Usage : ./HW2 ../testcase/<.cell> ../testcase/<.nets>

e.g. : ./HW2 ../testcases/p2-1.cells ../testcases/p2-1.nets

Q2 : The final cut size and the runtime of each testcase.

CASE1 : runtime = 0.006 sec , cut size = 6

CASE2 : runtime = 0.771 sec , cut size = 134

CASE3 : runtime = 530.076 sec , cut size = 2201

Q3 : Runtime = $T_{IO} + T_{computation}$. For each case, please analyze your runtime and find out how much time you spent on I/O and how much time you spent on the computation (FM Algorithm).

CASE1 : T_{IO} = 0.003 sec , $T_{computation}$ = 0.003 sec

CASE2 : T_{IO} = 0.032 sec , $T_{computation}$ = 0.739 sec

CASE3 : T_{IO} = 0.739 sec , $T_{computation}$ = 537.702 sec

Q4 : The details of your implementation containing explanations of the following questions:

(I) Where is the difference between your algorithm and FM Algorithm described in class? Are they exactly the same?

Ans ： 首先，在課堂上提到的 FM 每一次都會從 gain 最大的裡面挑出一個移動，但是在實作中不一定每一次都會如此幸運；我的方法會繼續往 gain 更小的 cells 找(最多找到 gain 值為 max gain-10)，直到找到一個或是確定都不能被移動為止。第二，課堂上提到的 FM 會一直做到不能再更好為止，但我的方法在最糟糕的情況下，可能每次移動都把所有 cells 都拜訪過一遍，因此時間複雜度比較接近 $O(P^2)$，為了讓程式執行時間不要那麼久，我有設定輪數限制在 15。

(II) Did you implement the bucket list data structure?

Ans ： 在實作過程中曾經嘗試過，但後來發現只要在自訂義的資料結構 cell 中加入兩個整數變數，用來記錄前後 cell 的 index 就可以模擬一個 bucket list，至於模擬多個 bucket lists 只需再使用一個整數陣列，用來記錄每個 bucket list 的第一個 cell(若值為零，就代表此 bucket list 為空)。

(III) How did you find the maximum partial sum and restore the result?

ANS ： 我使用一個 vector 將被移動的 cells 記錄下來，並且在移動過程中紀錄最大的 partial sum（若移動一個 cell 後的 partial sum 比原先的最大 partial sum 大，則更新)，同時記下最大 partial sum 的移動次數。有了這些資訊後就可以將不想要移動的 cell 恢復成原本的 group。

(IV) Please compare your results with the top 3 students' results from last year and show your advantage either in runtime or in solution quality. Are your results better than them?

ANS ： CASE1 的結果 CUTSIZE 為 6，與最好之結果相同，CASE2 的結果贏過 TOP3，CASE3 之結果贏過 TOP3 的其中一個，至於執行時間則是都輸。我認為原因在於我的方法複雜度接近 $O(P^2)$，即使有限制輪數，執行時間仍然會因為 case 的 size 上升而劇烈攀升；而且我沒有比較好的尋找 initial solution 之方法，也可能是導致結果不夠好且程式要執行較久才能結束之原因。

p.s. 輪數限制只會替 case3 減少兩輪的運算，對於 case1、case2 沒有影響，但若是測資更大，此舉應有顯著效果。

(V) What else did you do to enhance your solution quality or to speed up your program?

(1) 輪數限制(最大 15)。

(2) 將 bucket lists 融合入自訂義的資料結構 cell。

(3) 將 cell list 及 net list 進行排序，再使用 binary search 的方法為每一個 cell (or net) 建立與之相聯的 nets'(or cells') index list。

(e.g. 若 cell5 屬於 net1、net2、net3，則使用 binary search 為 cell5 建立一個 list 以紀錄 net1、net2、net3 在 net list 中的 index，當需要相關資訊時，抓取資料的時間就會比較少)。

(4) 當 current max gain 的 cells 都因為平衡限制而不能移動時，會往 gain 值更小的 cells 移動，但若是 gain = max gain-10 時仍然找不到，則結束尋找之動作。

(VI) What have you learned from this homework? What problem(s) have you encountered in this homework?

(1) 以前在做作業(除了麥老師的積體電路輔助設計)時通常對於執行時間較不太在意，因為時間都不會超過 10 秒，但在此次作業中，當問題變大時，時間消耗也會有明顯的上升，因此在每一個可以省時間的細節都不能放過；

(2) 即使經典的演算法的概念看起來很簡單明瞭，但實作卻不一定如此的輕鬆，且許多小細節可能會有些微的不同，要稍微修改應變，當然資料結構的選用也是一大學問，找到合適的資料結構也需要花一些時間構思。

(3) 之前只寫過兩行的 makefile，這次有機會花點時間了解 makefile，但還是需要再找其他資料來做更深入的了解。