# Homework 2: Two-way Min-cut Partitioning

**109062556 李濬安**

---

## ● How to compile and execute your program?

可以根據以下資料來去 compile & Run. (src/README)

```
--  How to Compile the
    In this directory, enter the following command:
    $ make
    It will generate the executable file "hw2" in "HW2/bin".

    if you want to remove it, please enter the following command:
    $ make clean

--  How to Run
    In this directory, enter the following command:
    Usage: ../bin/<exe> <net file> <cell file> <outout file>
    e.g.:
    $ ../bin/hw2 ../testcases/p2-1.nets ../testcases/p2-1.cells ../output/p2-1.out

    In "HW2/bin/", enter the following command:
    Usage: ./<exe> <net file> <cell file>
    e.g.:
    $ ./hw2 ../testcases/p2-1.nets ../testcases/p2-1.cells ../output/p2-1.out
```

## ● The final cut size and the runtime of each testcase

| Testcase | P2-1 | P2-2 | P2-3 | P2-4 | P2-5 |
|---|---|---|---|---|---|
| Cut size | 6 | 219 | 3596 | 47086 | 130591 |
| Runtime | 0.012 | 0.141 | 6.897 | 14.356 | 29.818 |

## ● Analyze your runtime. (Runtime = $T_{IO}$ + $T_{computation.}$)

| Testcase | P2-1 | P2-2 | P2-3 | P2-4 | P2-5 |
|---|---|---|---|---|---|
| $T_{IO}$ (s) | 0.002493 | 0.026437 | 0.489314 | 0.942929 | 3.39629 |
| $T_{computation}$(s) | 0.011222 | 0.116757 | 6.5239 | 13.6071 | 26.7455 |
| Runtime(s) | 0.012 | 0.141 | 6.897 | 14.356 | 29.818 |

可以發現到，大部份時間主要是花在 FM algorithm 上。但我的 FM algorithm 時間會比我上述的 $T_{computation}$ 來的短，因為在我的方法中，我執行了 17 次 FM algoritm，所以計算一次 FM 時間大概分別是{ 0.004, 0.027, 3.59, 8.47, 21.12(s)}

## Question&Answer

1. Where is the difference between your algorithm and FM Algorithm?
   實作的 FM algorithm 和課本的一樣。

2. Did you implement the bucket list data structure?
   有，但和課本的略為不同!
   不同處是，我沒有分 bucketlistA, bucketlistB，我是統一存到同一個 bucketlist。目的為，在實作時發現當要選擇 gain 最大的那 array 裡的 cell 來當作移動的 cell 時，會需要先對 bucketlistA 該 gain 值裡的 cell 去看看是否移動後會 balance，但如果沒有可移動的 cell，就換檢查 bucketlistB 該 gain 裡的 cell，那如果通通都沒有，又回到 bucketlistA 找第二大的 gain 的 cell，根據上述我發現如果存在同一個 bucketlist，然後只要有存 cell 目前是在 set A 還是 set B 就好了，這樣不僅可以省掉很多的比對，也較容易去 debug。

3. How did you find the maximum partial sum and restore the result?
   對於 max partial sum 會有一變數 currentpartialsum 紀錄是否加了現在移動 cell 的 gain 值會比之前的 greatpartialsum 還來的大，如果比較大的話，更新 greatpartialsum，並且記錄該移動 cell 是第幾步。

```
if(greatpartialsum <= currentpartialsum){
    greatpartialsum = currentpartialsum;
    targetstep = move_order.size();
```

   對於 restore the result，會有一 vector 存 move order，那根據上述的 targetstep，可以找到說從 targetstep 後面移動的 cell 是多餘的，因此只要把那些 step 所移動的 cell 移回來就行。

```
for(int i = move_order.size()-1 ; i > targetstep-1 ; i--){
    if(move_order[i]->set == 'A'){//move A to B…
    }else{//move B to A…
}
```

4. What else did you do to enhance your solution quality or to speed up your program?
   對於 Speed:
   - 有設一個 countdown 來倒數做了幾次 move cell 後，greatpartialsum 並沒有變好，就直接 early stop。
   - 在同一個 function 裡，對於常只用的指標會統一設一個指標變數來代表他，這樣就不用每次寫到他的時候，要再去跑一次所指的位置的值。
     e.g.: `vec_ptr = &bucketlist[cell_ptr->gain];`

對於 Quality:

- 我發現 initial partition 對於結果是相當影響的，因此我是根據 net 所連接的 cell 來先將她們分 set，意思就是同一條 net 的 cell 在 initial partition 時會先盡量讓他們同一個 set，這對我的 cut size 有一定程度的影響。
- 而根據上述的道理，我加入了另一個想法是將跑完 FM algorithm 的分群結果，再去做一次 FM algorithm，這樣是因為經由前一次 FM algorithm 出來的 data cut size 有一定的減少，而將此 data 再當作下一次的 initial partition，重複執行 17 次，發現 cut size 有顯著再下降。

5. <u>If you implement parallelization, please describe the implementation details and provide some experimental results.</u>
   未實作平行化

- **Please compare your results with the top 5 students' results from last year.**

| | cutsize | | | | | runtime | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Ranks | p2-1 | p2-2 | p2-3 | p2-4 | p2-5 | p2-1 | p2-2 | p2-3 | p2-4 | p2-5 |
| 1 | 6 | 191 | 4441 | 43326 | 122101 | 0.01 | 0.07 | 3.05 | 5.01 | 42.06 |
| 2 | 6 | 161 | 1065 | 43748 | 125059 | 0.01 | 0.1 | 3.11 | 9.84 | 18.77 |
| 3 | 6 | 358 | 2186 | 45430 | 122873 | 0.04 | 0.78 | 21.21 | 115.38 | 59.78 |
| 4 | 5 | 302 | 1988 | 46064 | 124862 | 0.03 | 0.17 | 7.04 | 6.93 | 8.22 |
| 5 | 6 | 411 | 779 | 46356 | 125151 | 0.01 | 0.16 | 5.49 | 12.31 | 13.27 |
| mine | 6 | 219 | 3596 | 47086 | 130591 | 0.01 | 0.14 | 6.89 | 14.35 | 29.81 |

Cutsize:

- P2-1 跟之前一樣小
- P2-2 比之前一些人還小
- P2-3 只有比第 rank 1 的還小
- P2-4, P2-5 都比之前的大，但沒有大太多

Runtime:

- P2-1 跟之前一樣快
- P2-2 比之前一些人還快
- P2-3 跟大多數人一樣快
- P2-4 跟大多數人一樣快
- P2-5 比 rank1, rank5 還快

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

在此次作業，實作了 Two-way Min-cut Partitioning，學到相當多實作上的經驗，包括資料結構多變的應用，抑或是演算法的完整度，也花很大的心力去完成這項作業並盡力去優化，是相當充實的!
需要進步的還很多，重要的是對於 cut size。雖然在 runtime 上表現算是很好的，但是最重要的是演算法對於 cut size 的結果並不是很優的，我想可以再去思考如何讓 cut size 降低，這是我需要再去 survey 的。