Report

111061590 林學謙

1. **How to compile and execute your program, and give an execution example.**

先切換位置到/src 資料夾

```
[u111061590@ic55 ~/HW2]$ cd src
[u111061590@ic55 src]$
```

$ make

```
[u111061590@ic55 src]$ make
g++ -O3 -std=c++11 -I /usr/local/include/boost/ -c main.cpp -o main.o
g++ -O3 -std=c++11 -I /usr/local/include/boost/ main.o -o ../bin/hw2
[u111061590@ic55 src]$
```

他會創造出 make.o and hw2.exe

$ cd ../bin

並且將下列的 command 複製並貼上，即可執行 (HW2/src/README)

```
Run command:
$ ./hw2 ../testcase/public1.txt test.hgr ../output/public1.out
$ ./hw2 ../testcase/public2.txt test.hgr ../output/public2.out
$ ./hw2 ../testcase/public3.txt test.hgr ../output/public3.out
$ ./hw2 ../testcase/public4.txt test.hgr ../output/public4.out
$ ./hw2 ../testcase/public5.txt test.hgr ../output/public5.out
$ ./hw2 ../testcase/public6.txt test.hgr ../output/public6.out

Verify command:
$ ../verifier/verify ../testcase/public1.txt ../output/public1.out
$ ../verifier/verify ../testcase/public2.txt ../output/public2.out
$ ../verifier/verify ../testcase/public3.txt ../output/public3.out
$ ../verifier/verify ../testcase/public4.txt ../output/public4.out
$ ../verifier/verify ../testcase/public5.txt ../output/public5.out
$ ../verifier/verify ../testcase/public6.txt ../output/public6.out
```

將其中一列複製並貼上

```
[u111061590@ic55 bin]$ ./hw2 ../testcase/public1.txt test.hgr ../output/public1.out
start
*********************************************************************************
 HMETIS 1.5.3  Copyright 1998, Regents of the University of Minnesota

HyperGraph Information ----------------------------------------------------
 Name: test.hgr, #Vtxs:  2735, #Hedges:  2644, #Parts: 2, UBfactor: 0.20
 Options: HFC, FM, Reconst-False, No V-cycles, No Fixed Vertices

Recursive Partitioning... -------------------------------------------------
-------------------------------------------------------
[  2735   2642   2.92   3.02   0.97   0.16,   2642      0   2642   0.97]
[  1640   1571   3.50   3.66   0.96   0.26,   1572   1070   2642   0.96]
[   983   1076   4.36   3.98   1.09   0.34,   1087   1555   2642   1.11]
[   589    755   5.44   4.24   1.28   0.42,    784   1858   2642   1.33]
[   353    490   6.69   4.82   1.39   0.49,    557   2085   2642   1.58]
[   211    434   9.91   4.82   2.06   0.63,    537   2105   2642   2.55]
[   126    357  13.75   4.85   2.83   0.82,    495   2147   2642   3.93]
[    75    290  19.07   4.93   3.87   0.97,    458   2184   2642   6.11]


 ---------------------------------------------------------------------------
  Summary for the 2-way partition:
              Hyperedge Cut:        92               (minimize)
      Sum of External Degrees:      184              (minimize)
                 Scaled Cost:  5.56e-05              (minimize)
                  Absorption:   2623.54              (maximize)

      Partition Sizes & External Degrees:
          1832[  92]    903[  92]

Timing Information --------------------------------------------------------
  Partitioning Time:              0.016sec
          I/O Time:               0.004sec
*********************************************************************************
hemetis end
start the objective partition
original cutsize: 92
8151.00 10175.00 70.00
8151.00 10175.00 75.00
areaA_constraint: 58055497.50
areaB_constraint: 62202318.75
after move vertex to fulfill the objective cutsize: 92
fulfill the objective
[u111061590@ic55 bin]$
```

就可以跑出結果 為了驗證是否跑出結果 可以使用助教提供的 Verify 工具

```
[u111061590@ic55 bin]$ ../verifier/verify ../testcase/public1.txt ../output/output1.out
[Success] Your output file satisfies our basic requirements.
```

使用助教給予的 HW2_grading.sh 去跑結果

```
[u111061590@ic55 ~/HW2_grading]$ bash HW2_grading.sh
|-------------------------------------------------|
|                                                 |
|    This script is used for PDA HW2 grading.     |
|                                                 |
|-------------------------------------------------|
grading on 111061590:
  testcase |    cutsize |    runtime | status
   public1 |         85 |       0.99 | success
   public2 |      11129 |       7.94 | success
   public3 |      38367 |      48.59 | success
   public4 |       3451 |       3.09 | success
   public5 |      25091 |     129.23 | success
   public6 |     186904 |     192.67 | success
|-------------------------------------------------|
|                                                 |
|    Successfully generate grades to HW2_grade.csv|
|                                                 |
|-------------------------------------------------|
```

2. The final cut size and the runtime of each testcase

	Public1	Public2	Public3	Public4	Public5	Public6
Cut size	82	10902	2424	3194	14065	48997
Runtime (s)	3.29	27.36	159.61	11.05	361.75	641.09

可以看出最後一個問題的 cut size 特別大，因為其有較大的 cells，至於問 public5，我覺得應該是有比較特殊的 cell，所以需要相當多得時間在計算上面，有時一個 episode 就要 30 秒左右，對於我的方法而言，有好的 initial partition 很重要。

3. The details of your algorithm

先透過讀取資料並將資料分割，拆解成想要的形式

```
struct LibCell {
    string name;
    int x, y;
    int area;
};

struct Tech {
    string name;
    int numLibCells {};
    vector<LibCell> libCells;
};
struct DieSize {
    long double width, height;
};
struct DieAssignment {
    string dieName;
    string techName;
    long double utilization;
};
struct Cell {
    string name;
    string libCellName;
    int vertex_index;
    int partition = -1;
    long long both_area[2] = {0,0};
    vector<int> cell_net;
    bool lock = false;
};
struct Net {
    string name;
    int numberconnectedCell;
    vector<string> connectedCells;
    vector<int> edge_index;
    map<int,int> vertex_index;
};
```

使用 structure 將重要的資料儲存

演算法的部分，我先使用 hmetis 算出比較好的 initial partition，總共會跑 100 次 iteration，讓其找到最小 cutsize，藉由多次的結果，找出最好的那一個結果，並輸出成.out 檔。

```
if (tot_area[0] > areaconstraint[0] && tot_area[1] > areaconstraint[1] && cells[rand].partition == 1)
{
    if (cells[rand].both_area[0] < cells[rand].both_area[1])
    {
        //move
        cells[rand].partition = 0;
        tot_area[0] += cells[rand].both_area[0];
        tot_area[1] -= cells[rand].both_area[1];
        continue;
    }
}
else if (tot_area[0] > areaconstraint[0] && cells[rand].partition == 1) {
    continue;
}else if(tot_area[0] > areaconstraint[0] && cells[rand].both_area[1] < cells[rand].both_area[0] ) {
    cells[rand].partition = 1;
    tot_area[0] -= cells[rand].both_area[0];
    tot_area[1] += cells[rand].both_area[1];
    continue;
}
else if (tot_area[0] > areaconstraint[0] && tot_area[1] + cells[rand].both_area[1] < areaconstraint[1]) {
    cells[rand].partition = 1;
    tot_area[0] -= cells[rand].both_area[0];
    tot_area[1] += cells[rand].both_area[1];
    continue;
}
```

```
if (tot_area[1] > areaconstraint[1] && tot_area[0] > areaconstraint[0] && cells[rand].partition == 0) {

    if (cells[rand].both_area[1] < cells[rand].both_area[0])
    {
        //move
        cells[rand].partition = 1;
        tot_area[1] += cells[rand].both_area[1];
        tot_area[0] -= cells[rand].both_area[0];
        continue;
    }
}
else if (tot_area[1] > areaconstraint[1] && cells[rand].partition == 0) {
    continue;
}else if (tot_area[1] > areaconstraint[1] && cells[rand].both_area[0] < cells[rand].both_area[1] ) {
    cells[rand].partition = 0;
    tot_area[1] -= cells[rand].both_area[1];
    tot_area[0] += cells[rand].both_area[0];
    continue;
}
else if (tot_area[1] > areaconstraint[1] && tot_area[0] + cells[rand].both_area[0] < areaconstraint[0]) {
    cells[rand].partition = 0;
    tot_area[1] -= cells[rand].both_area[1];
    tot_area[0] += cells[rand].both_area[0];
    continue;
}
```

當兩塊 die 的面積都超出範圍，會先用 random 的方式去找任一個 cell(按照順序會導致其 local minima)，找到後根據幾個條件，第一個條件為如果兩面積皆不符合，則根據其在哪一位置擁有較小的面積，則移過去，如果另一邊已經調整完畢，已經小於面積限制，下一步則是找其 cell 的另一邊的面積是否會比較小(但通常使用率高的芯片擁有較大的 cell size，所以每一個 cell 的面積通常都會比另一邊還來的大，但總有一兩個是不一樣的，所以必須將此條件寫上)，最後

則是如果另一邊還有空間，就直接移過去，這樣即可找到符合問題的解。

如果要符合助教所提供的 HW2_grading.sh 的要求，100 次會導致時間過長(圖表在第四部份)，所以把次數改成 30 次做為結果。

4. What tricks did you use to enhance your solution quality?

有時有一些問題是難以解決的，比如說 public5，所以必須將 greedy 的概念引入，給予其閾值，如果小於它，則直接進行移動(強化學習又稱作 exploration)，這樣即可解決局部最小值的問題。

```
int force_change = distribution1(gen);
if (force_change < 8)
{
    //std::cout << "random part" << endl;
    if (cells[rand].partition == 0)
    {
        tot_area[0] -= cells[rand].both_area[0];
        tot_area[1] += cells[rand].both_area[1];
        cells[rand].partition = 1;
    }
    else {
        tot_area[1] -= cells[rand].both_area[1];
        tot_area[0] += cells[rand].both_area[0];
        cells[rand].partition = 0;
    }
    continue;
}
```

是否有 exploration?

benchmark	public1	public2	public3	public4	public5	public6
exploration	有解	有解	有解	有解	有解	有解
without exploration	有解	有解	有解	有解	無解	無解

可以看出如果沒有 exploration，public5 是找不太到解的，可以看出這是必要的機制。

今天將次數降低，可以看出其結果是會有差異的，當然 run time 的時長也會有差別。

cut size of 100 iterations	82	10902	2424	3194	14065	48997
Runtime (s)	3.29	27.36	159.61	11.05	361.75	641.09
cut size of 10 iterations	99	11962	43721	3562	36674	203876
Runtime (s)	0.35	2.83	17.29	1.16	73.17	67.01

如果沒有時間限制，會許越高的 iteration 可以找到越好的解，但時長當然成正比增加，如何取得平衡是很重要的課題。

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

我本身不是資工系的學生，只能說這樣的作業對我來說是相當的難，一開始在寫的時候，連題目讀懂都需要花相當多的時間，再來就是在邏輯設計上，需要花大量的時間研究，比如說在一開始設定 structure 的時候，該包含什麼參數，是花費相當多的時間做修改，因為永遠不知道自己缺少什麼。再來就是在處理面積問題上，花費相當多時間，最後為了克服問題，想方設法才想出了藉由 exploration 的方式，來克服難以解決的問題。

我從這個功課學到很多，或許這對於資工系學生來說是簡單的問題，比如說該如何優化計算時間，變相的可以增加 structure 的變數，雖然會增加空間，但至少可以讓程式運行時間可以短一點。除此之外，我也是第一次使用讀寫這兩個功能，所以對於其運作方式也花了不少時間。