CS6135_HW2_112065504_report

112065504 徐祈

A. How to compile

1. cd ./HW2_grading/student/112065504/HW2/src
2. make
   - This will compile and generate executable file
3. ../bin/hw2 ../testcase/{testcasename}.txt ../output/{testcasename}.out
   - Ex: ../bin/hw2 ../testcase/public1.txt ../output/public1.out
   - Execute the program with given input file under testcase folder.

B. final cut size and the runtime

1~6 means the public testcase. The run time is measured by time ./hw2 …
The program is tested on ic21.

	cutsize	Runtime (elapse time)
public1.txt	232	0.01s
public2.txt	3771	0.17s
public3.txt	7111	2.39s
public4.txt	5194	0.07s
public5.txt	1509	0.61s
public6.txt	10479	18.64s

C. Algorithm detail

我的 code 大致上參考 Fiduccia–Mattheyses algorithm (A Linear-Time Heuristic for Improving Network Partitions)來實作。

以下用 pseudo code 表示一些重要的 Functions，部分說明在 code 註解中:

```
void init_partition()
    for (autp& c: cellArray)
        if (cell has smaller size on A)
            if (tryPutOn(A))
                put on A
            else
                put on B
        else
            if (tryPutOn(B))
                put on B
            else
                put on A
```

在 public testcase 中，這種無腦擺的方法可以擺上去，但無法保證 private testecase 也可以，因此我參考 greedy & SA 的概念，在發現 initial partition invalid 後，跑以下 function 調整: 概念是我先隨便找兩顆 cell，如果他們在不同 Die 上，看看交換後會不會"更好"，而我將更好定義成，至少其中一塊 Die 合 constraint，且另一塊在 swap cell 後會降低 utilization。另外如果只寫成這樣，很可能會卡住或走很慢，因此給他一個很小的機率(ex. 0.01)會不管 swap 後的結果，直接換下去，實驗後發現可以很快繞過 local minima。

```
void greedy_fix(){
    while(True){
        Randomly pick two cells c1, c2
        bool randomSelectBad = rand()%100 == 0; // small prob that select bad answer
        if (c1 on B and c2 on A){
            if (After swap, one die is valid,
                 the other is invalid but utilization improves ||
                randomSelectBad
            ){
                Swap c1, c2
                if(Die B satisfied area constraint)
                    break;
                else
                    continue;
            }
            else{
                continue;
            }
        }
        else if(c1 on A and c2 on B)
             By anology
```

再來後面就沒什麼特別的，照著老師教的慢慢刻~

```
void FM()
    while(1)
        BucketList bListA, bListB; // build bucket list before each pass
        pass(bListA, bListB);
        auto [moveTo, Gk] = getBestMove(); // get largest prefix of cell gain
        if(moveTo == -1 or Gk <= 0)// keep doing pass until Gk<=0 or can't move
            Reverse effect after best move that giving Gk
            break;
        Reverse effect after best move that giving Gk
        Unlock all cells
```

```
void pass(BucketList& bListA, BucketList& bListB){
    init_distribution();  // Calculate initial cell's distribution
    init_cellGain(bListA, bListB); // initialize cell gain
    while((baseCell = maxGainCell(bListA, bListB))){ // keep find max gain cell, if
not null
        steps.emplace_back(baseCell) // record baseCell
        updateGain(baseCell); // update gain having relation with baseCell
    }
}
```

D. Enhance solution quality

原本 FM 中找 max gain 時，如果真正有 max gain 的 cell 在移動後面機會不合法，應該會被留在 bucket list 裡面，只是繼續往下找有沒有其他合法的 cell，因為你不會知道是否在選擇其他人移動後，會讓當前這顆真正有 max gain 的 cell 也可以合法移動。但這樣做會讓效能變超差，以下為每筆 public testcase 實測結果:

	cutsize	Runtime (elapse time)
public1.txt	245	0.12s
public2.txt	3646	1.51s
public3.txt	7444	9min 24.69s
public4.txt	5197	0.09s
public5.txt	1861	17min 7.08s
public6.txt	10521	55min 54.62s

於是我改成，再找 max gain cell 時，只要看到 bucket list 裡面積不合法的 cell，就馬上移除，雖然會讓解空間變小，但實際上沒有讓 cutsize 變差多少，甚至大部分的公開測資中，都讓 cutsize 在略為下降。而且改完後大幅改善程式效率問題。最大測資 6 也能在 18 秒左右跑完:

	cutsize	Runtime (elapse time)
public1.txt	232	0.01s
public2.txt	3771	0.17s
public3.txt	7111	2.39s
public4.txt	5194	0.07s
public5.txt	1509	0.61s
public6.txt	10479	18.64s

另外，使用 flush 可以較大測資的 output 寫入速度從幾分鐘降到幾秒鐘，快非常多。

E. What have you learned from this homework?

- 學到如何使用最適合資料結構完成 bucket list。
- 如何找到合法的 initial partition
- 學到如何把論文實做出來
- 學到用 flush 可以讓 output 速很多
- 編譯用-o3 可以最優化編譯出來的程式，增進效能