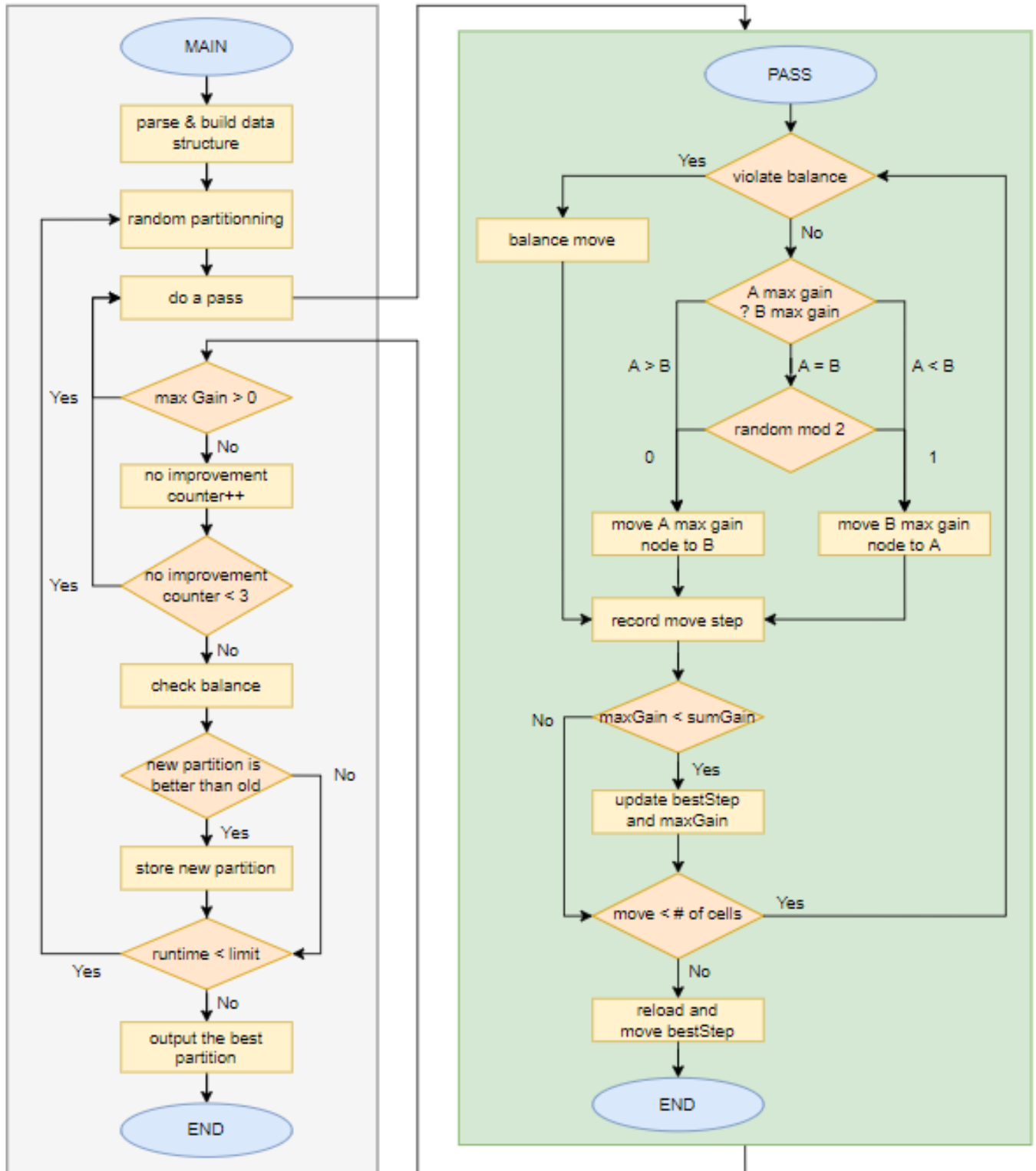


110521167_PA3_Report_Partitionning

1. Flow chart



2. How to compile and execute your program

模式	指令	備註
Basic	<code>\$./PA3 case1 case1.out</code>	Default runtime = 3000 sec.
With runtime limit	<code>\$./PA3 case1 case1.out 30</code>	Runtime = 30 sec.
Use makefile	<code>\$ make run INPUT=case1 OUTPUT=case1.out</code>	

3. The completion of the assignment

我基本的 FM algorithm 都有完成，除了助教給的 testcase 以外，我另外抓了兩個較大的 testcase 來測試，還有另外找了 shmetis 做為 cut size 的參考對象。

我的程式在 GainBucket_A 的 max_gain 與 GainBucket_B 的 max_gain 相等的情況下採用 random 的方式選擇移動的節點，加上 initialize 的方式也是採用 random，因此每一次產生的 partition 優劣不如 shmetis 穩定，若運氣好能夠以更短的 runtime 達到和 shmetis 一樣好 partition，但絕大多數時候需要經過進行一輪又一輪的探索，若採用預測後面幾步的方式來判斷這一步移動哪一個 cell 是更好的選擇，或許能夠達到更好的效果，由於時間不夠這是我唯一沒有實現的地方。


在終止條件的部分，我首先在跑完一個 PASS 之後回到 local optimum 的 function 中嘗試了 simulate annealing 的方式如下。

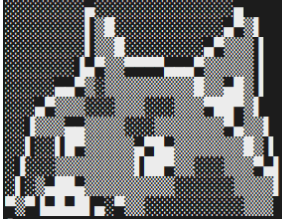
SA process	<pre>if(deltaCost >= 0 && acceptUphill(deltaCost)) reloadPartition(rndStep); else reloadPartition(bestStep); coolDown();</pre>	
Cost function	$\Delta Cost = 0.9\Delta Gain + 0.1\Delta Unbalance$	Random step 與 local optimum 比較，透過加權達到以下兩點。 1. cost 優先考慮 cut size 2. cut size 相等時較平衡的 partition 得到較低的 cost
Accept uphill	<pre>double rndStandard = unif(generator); // random [0, 1) return rndStandard < exp(-deltaCost / T);</pre>	
Temperatures	$T_0 = 10Avg.Gain$ $T(n+1) = 0.7 T(n)$ $T_{end} = 0.01$	
Result (case1)	<pre>PS D:\C++\NCU_MS_1-2\NCUCAD\PA3> .\a.exe .\case1 case1.out cut size: 210 -> 156 -> 65 -> 25 -> 23 -> 46 -> 58 -> 33 -> 20 -> 36 -> 39 -> 31 -> 18 -> 23 -> 29 -> 12 -> 12 -> 12 -> 12 -> 12 -> 12 -> 12 -> 20 -> 13 -> 16 -> 13 -> 12 -> 8 -> 8 -> 6 -> 6 -> 6 -> 6 -> 6 -> 6 -> 6 -> 6 -> 6 cut size = 6 runtime = 0.017s</pre>	

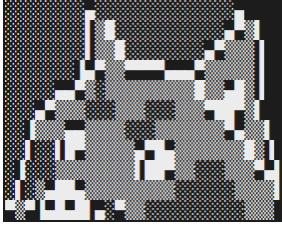
但或許是 SA 參數設計不夠精準，往往最後收斂的解只能夠近似 optimum，因此我最後採用了從多個 initial state 進行 greedy-FM 的方式，再從中挑選出最佳的解，並以 runtime limit 作為終止條件的方案。


另外原版 FM algorithm 採用 $\text{maxGain} \leq 0$ 作為終止條件，但我的作法略有不同，因為在選擇移動哪一個 Cell 的步驟使用了 random，即使經過一個 pass 後 $\text{maxGain} \leq 0$ ，下一個 pass 仍然有機會產生更好的 partition，因此我增加了變數 noImproveCnt 來控制要額外做幾次嘗試，也確實能夠加快產生更少 cut size 的 partition 的速度。

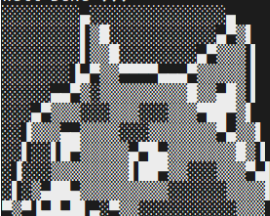
4. The results and runtimes for all test cases

Result of case0	Compare with Shmetis
<pre>[110521167@eda359_forclass code2]\$./PA3 case0 case0.out cut size = 1 runtime = 0s round = 1 [110521167@eda359_forclass code2]\$./PA3_CHECKER case0 case0.out error count: 0 Well Done !!!</pre> 	<pre>Summary for the 2-way partition: Hyperedge Cut: 1 (minimize) Sum of External Degrees: 2 (minimize) Scaled Cost: 6.25e-02 (minimize) Absorption: 6.50 (maximize) Partition Sizes & External Degrees: 4[1] 4[1] Timing Information ----- Partitioning Time: 0.001sec I/O Time: 0.001sec</pre>

Result of case1	Compare with Shmetis
<pre>[110521167@eda359_forclass code2]\$./PA3 case1 case1.out cut size = 5 runtime = 0s round = 1 [110521167@eda359_forclass code2]\$./PA3_CHECKER case1 case1.out error count: 0 Well Done !!!</pre> 	<pre>Summary for the 2-way partition: Hyperedge Cut: 5 (minimize) Sum of External Degrees: 10 (minimize) Scaled Cost: 1.42e-04 (minimize) Absorption: 355.74 (maximize) Partition Sizes & External Degrees: 188[5] 187[5] Timing Information ----- Partitioning Time: 0.043sec I/O Time: 0.001sec</pre>

Result of case2	Compare with Shmetis
<pre>[110521167@eda359_forclass code2]\$./PA3 case2 case2.out cut size = 4 runtime = 0s round = 1 [110521167@eda359_forclass code2]\$./PA3_CHECKER case2 case2.out error count: 0 Well Done !!!</pre> 	<pre>Summary for the 2-way partition: Hyperedge Cut: 4 (minimize) Sum of External Degrees: 8 (minimize) Scaled Cost: 1.60e-01 (minimize) Absorption: 6.00 (maximize) Partition Sizes & External Degrees: 5[4] 5[4] Timing Information ----- Partitioning Time: 0.002sec I/O Time: 0.001sec</pre>

Result of input2 (6211 nets, 6291 cells)	Compare with Shmetis
<pre> [110521167@eda359_forclass code2]\$./PA3 input2 input2.out cut size = 142 cut size = 93 cut size = 59 cut size = 45 cut size = 39 cut size = 36 cut size = 33 runtime = 45.76s round = 1440 [110521167@eda359_forclass code2]\$./PA3_CHECKER input2 input2.out error count: 0 Well Done !!! </pre> 	<pre> Summary for the 2-way partition: Hyperedge Cut: 33 (minimize) Sum of External Degrees: 66 (minimize) Scaled Cost: 3.34e-06 (minimize) Absorption: 6501.92 (maximize) Partition Sizes & External Degrees: 3201[33] 3090[33] Timing Information ----- Partitioning Time: 0.296sec I/O Time: 0.006sec </pre>

Result of input3 (89187 nets, 85013 cells)	Compare with Shmetis
<pre> [110521167@eda359_forclass code2]\$./PA3 input3 input3.out cut size = 77 cut size = 57 cut size = 22 cut size = 18 cut size = 15 runtime = 22.79s round = 39 [110521167@eda359_forclass code2]\$./PA3_CHECKER input3 input3.out error count: 0 Well Done !!! </pre> 	<pre> Summary for the 2-way partition: Hyperedge Cut: 15 (minimize) Sum of External Degrees: 30 (minimize) Scaled Cost: 8.45e-09 (minimize) Absorption: 89184.52 (maximize) Partition Sizes & External Degrees: 48066[15] 36947[15] Timing Information ----- Partitioning Time: 4.603sec I/O Time: 0.066sec </pre>

5. The hardness of this assignment and how you overcome it

我在 PA3 上遇到最大的困難是 SA 參數的設計，雖然有實現 SA 的概念但是並沒辦法消除最後收斂在 local optimum 的缺點。第二個是在做 GainBucket 的時候刻完了 doubly linked list 才發現 STL 中有 list 可以直接使用，或許下次遇到類似的情境可以用來加快開發，並比較效能差異。

6. Any suggestions about this programming assignment?

從 PA2 的經驗中我了解到，獲得相同質量的解的人絕對不在少數，因此我認為 runtime 是可以進一步提高鑑別度的因素，也是追求程式能力以及高分的一個誘因。另外我 report 一直被扣分，希望助教能在 DEMO 時告訴我被扣分的原因讓我能夠在下一份 report 修正，一直拿不到滿分很難過 QQ。

7. Data structures

(1) Node & Net

Node、Net 間彼此用 pointer 互相指著，在進行移動時方便找到 adjacent nodes，並能夠透過 Net 中 LSize、RSize 兩個變數判斷 Net 是否為 critical net，快速計算移動對 Node 的 gain 造成的影響。

```
class Node
{
private:
    string name;
    Node *LLink, *RLink;

    int gain;
    bool side, locked;
    vector<Net *> nets; // connected nets
}

class Net
{
private:
    string name;
    int LSize, RSize;
    vector<Node *> nodes; // connected nodes
}
```

(2) Bucket

在 Bucket 中，size 是用來輔助嘗試 FIFO 以外的可能性，但實驗結果 FIFO 較容易得到較少的 cut size。在這個 class 實現了 insert()、remove()、overloading operator[] 方便 GainBucket 使用。

```
class Bucket
{
private:
    int size = 0;
    Node *head = nullptr;
```

(3) GainBucket (bucket list)

使用 vector<Bucket> 實現 bucket list，以及使用 maxGainIdx 來指向 bucket list 中非空的 max gain bucket。

```
class GainBucket
{
private:
    static std::vector<GainBucket*> allGainBuckets;
    static int PMax; // maximum degree of all nodes
    int maxGainIdx = -1; // index of max gain node

public:
    std::vector<Bucket> bucketList;
```