CS6135 VLSI Physical Design Automation

Homework 2: Two-way Min-cut Partitioning

1.

學號: 113062632

姓名: 吳晟光

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

   輸入$ tar zxvf CS6135_HW2_113062632.tar.gz 解壓縮

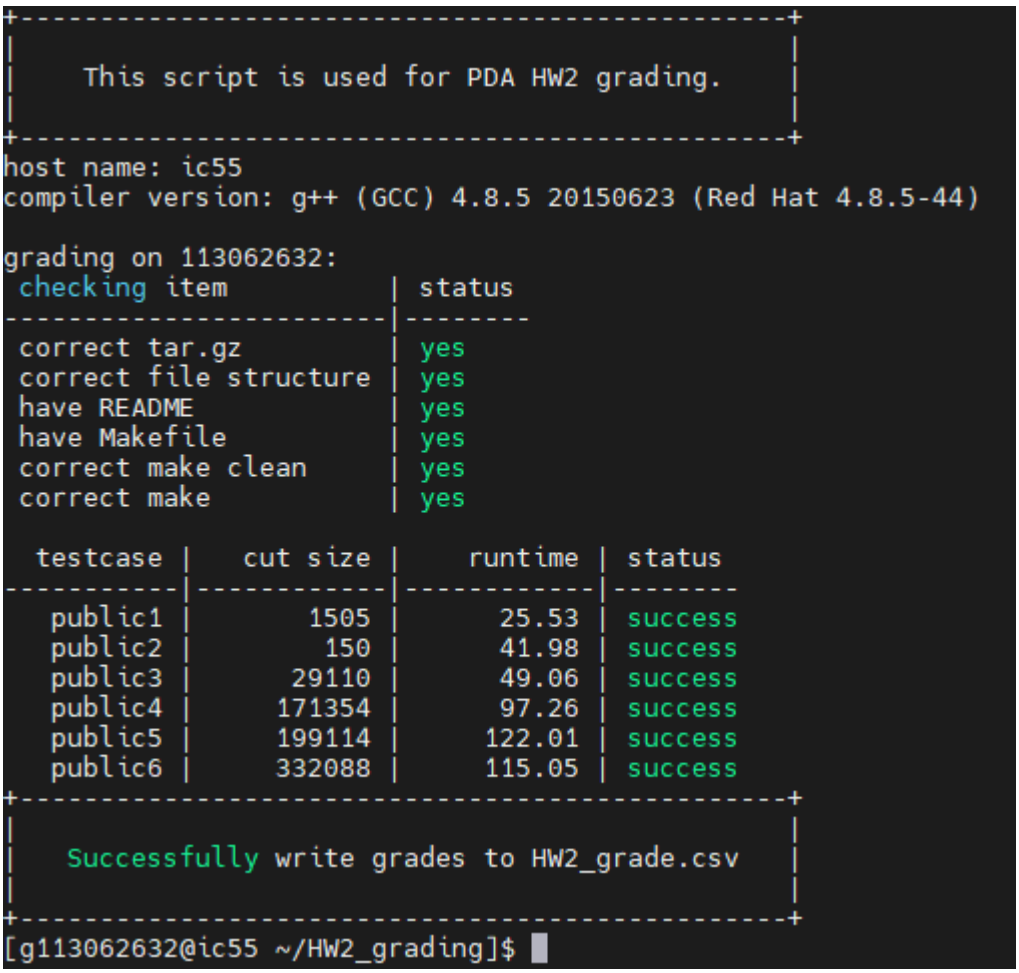
   進入 HW2/src，輸入$ make，再輸入

   $ ./bin/hw2 ../testcase/{input.txt} ../output/{output.out}

   便能執行程式，輸入 make clean 可以清除檔案

   Ex:
   $ tar -zxvf CS6135_HW2_113062632.tar.gz
   $ cd HW2/src
   $ make
   $ ./bin/hw2 ../testcase/public1.txt ../output/public1.out
   $ make clean

3. The final cut size and the runtime of each testcase. Paste the screenshot of the result of running the HW2_grading.sh as the picture shown below.



public1	1505	25.53	success
public2	150	41.98	success
public3	29110	49.06	success
public4	171354	97.26	success
public5	199114	122.01	success
public6	332088	115.05	success

4. The details of your algorithm. You could use flow chart(s) and/or pseudo code to help elaborate your algorithm. If your algorithm is similar to some previous work, please cite the corresponding paper(s) and reveal your difference(s).

   我使用的演算法是 simulated Annealing，cut size 變小即 accept，否則以$e^{-\frac{\Delta C}{T}}$的機率 accept，但不同於講義上的例子，我每次隨機選取時會從 dieA 和 dieB 各選一點做交換，因為我覺得假如真的

隨機取我怕每次都取 dieA 放到 dieB 把 dieB 放太滿，導致違反 utilization，用交換的就可以把 area 控制在一定範圍內，只是兩邊 cell 的個數會被固定，因此我寫了 8 種不同版本的 initial partition，每一種的放法都不一樣擺放進 dieA 和 dieB 中 cell 的個數也大相逕庭，我會在下一題介紹我的 initial partition，除此之外，我的 cutsize 的算法不是用直接用 net 的 weight 做計算，而是先把 hyper edge 換回普通的 edge，假設 Net 1 的 weight = 12 連接到 5 個點，那我會為每個連接的點兩兩之間都接上 weight = 12/(5-1)的 normal edge 成為 5 個 vertices 的 complete graph，只在最後 output file 時才會把真正的 cutsize 算出來，以下是我 simulated Annealing 的 pseudo code:

SA psuedo code

```
1. temprature = 150
2. cooling rate = 0.9999
3. min_temprature = 0.0001
4.
5. 1. randomly choose cellA from DieA, cellB from DieB
6. 2. Calculate new_cutsize if we accept the swap
7. 3. if (new_cutsize < old_cutsize)      swap
8.      else if( rand() / RAND_MAX < exp((old_cutsize - newcutsize) / temp
   erature))      swap
9.      else      reject
10. 4. temprature = cooling rate*temprature, update old_cutsize
11. 5. if temprature>min_temprature, back to 1.
12. 6. return old_cutsize
```

5. What techniques did you use to improve your solution's quality? Additionally, analyze how they contributed to the improvements. Plot the effects of those different settings like the ones shown below

最一開始我的 initial partition 是直接把 dieA 放滿再換 dieB 放，

然而這樣做跑到第 3 個 public case 就無法滿足 utilization，因此

我換成助教提示的做法，我一開始全部丟到 B，先用放進 dieB

所需的面積和放進 dieA 所需面積的差 sort，從改善最多的開始

丟進 A，這樣做確實讓我找到 valid initial partition，但是 cutsize

卻遠超過 baseline 和我原本 partition 得到的 cutsize，下圖為

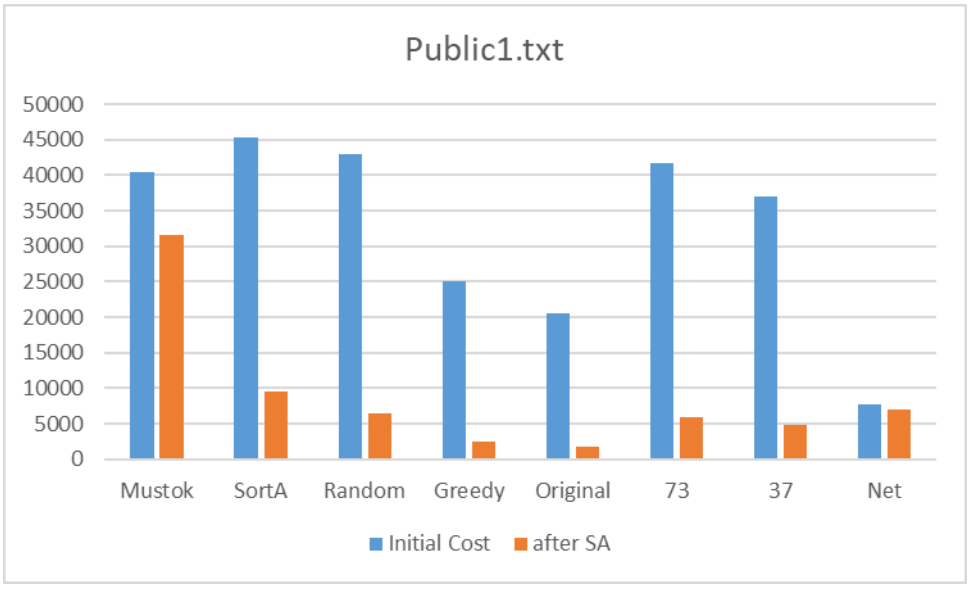
public2.txt 的結果，差距非常大

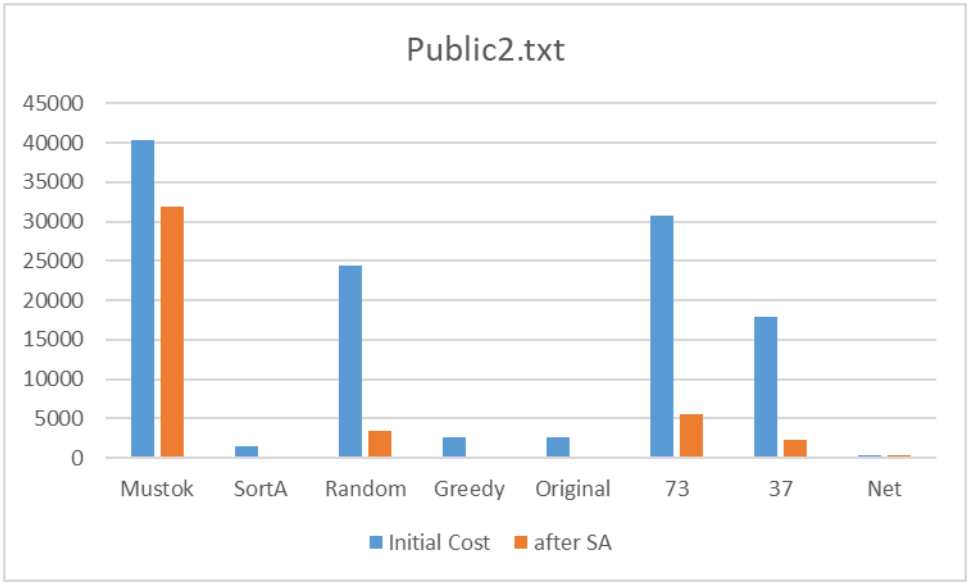
```
Mustok strategy initial cost: 40268.7
Original strategy initial cost: 2622.55
Simulated annealing cost: 32075.7
Simulated annealing cost: 239.548
```

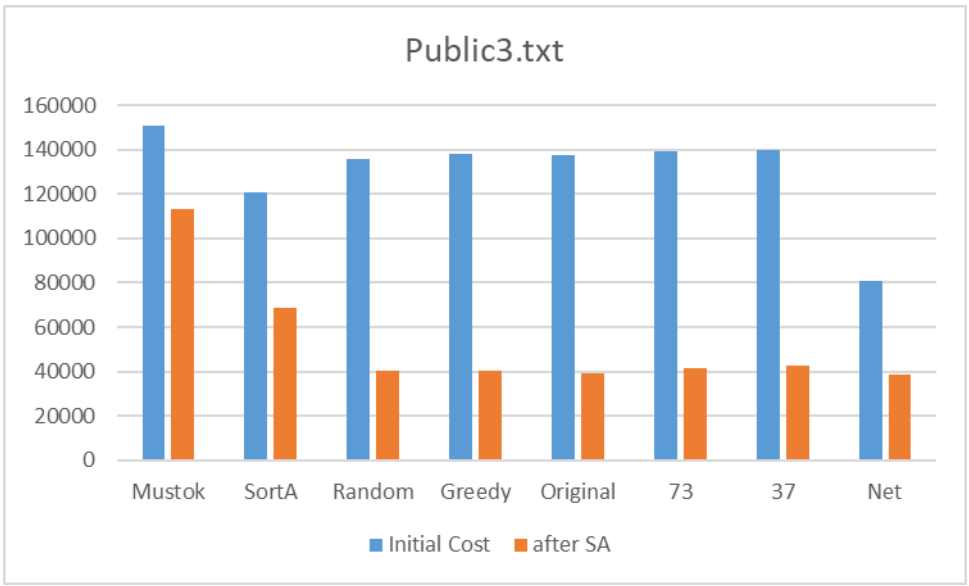
為此我寫了一個 function 幫忙修正 utilization 無法滿足的問題，這個 function 會一直把 utilization 無法滿足的那邊 sort 選出改善最多的和另一個 die 的 cell 交換或是直接丟到另一邊，因此解決了我最初的 partition 違反 utilization 的問題。

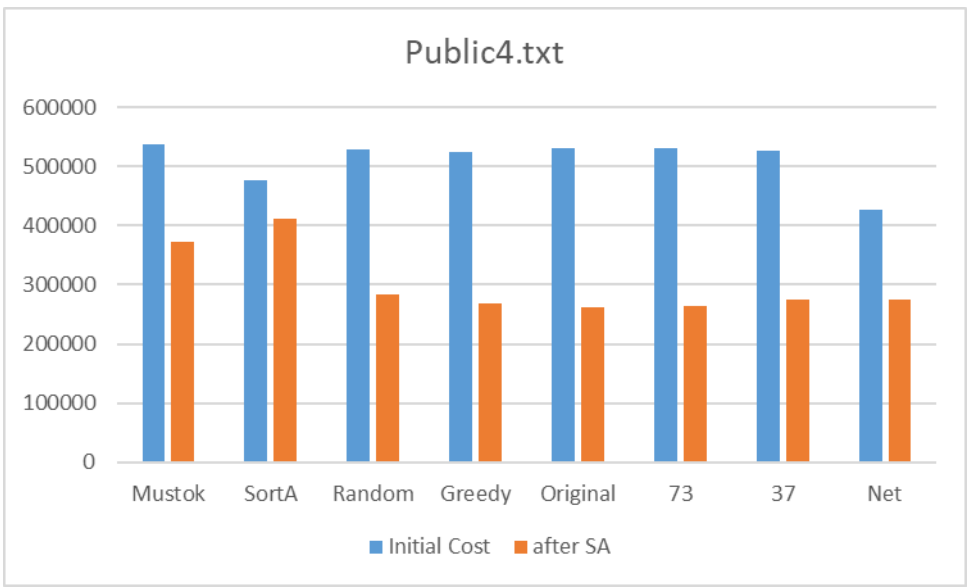
並且因為我現在有這個修正的 function，我就隨便寫了各種 initial partition 讓每一個都跑一遍 SA 找出結果最好的回傳，其中最特別的是 initialPartition_net()，做法是把 net weight 較大的盡量放同一邊，這個 initialPartition_net()得到的初始 cutsize 很小沒錯，但是跑 SA 卻沒辦法減少很多，我猜測是因為隨機選取較難找到能改善 cutsize 的 cell，得到的答案也很一般。

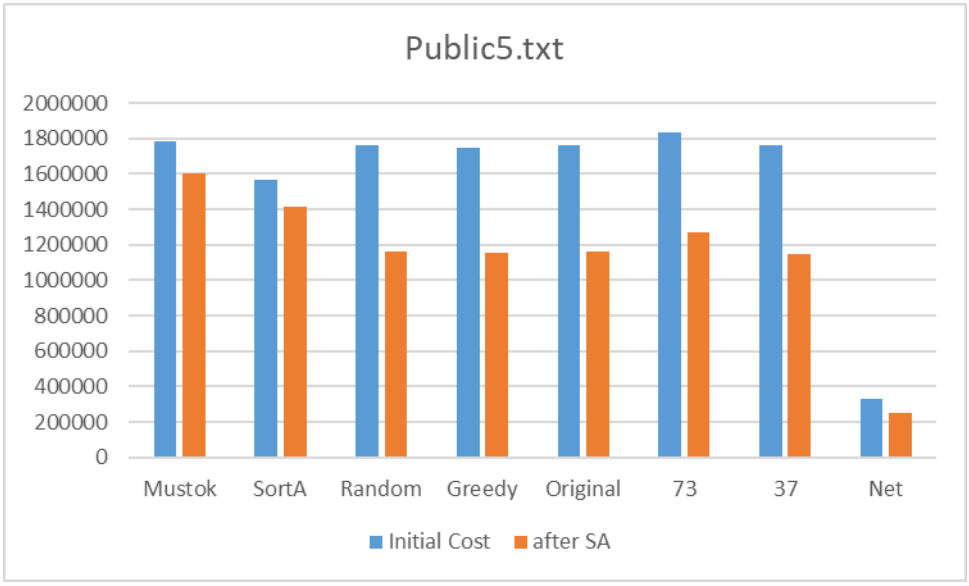
下圖是我的八種產生的初始結果和 SA 結果長條圖:

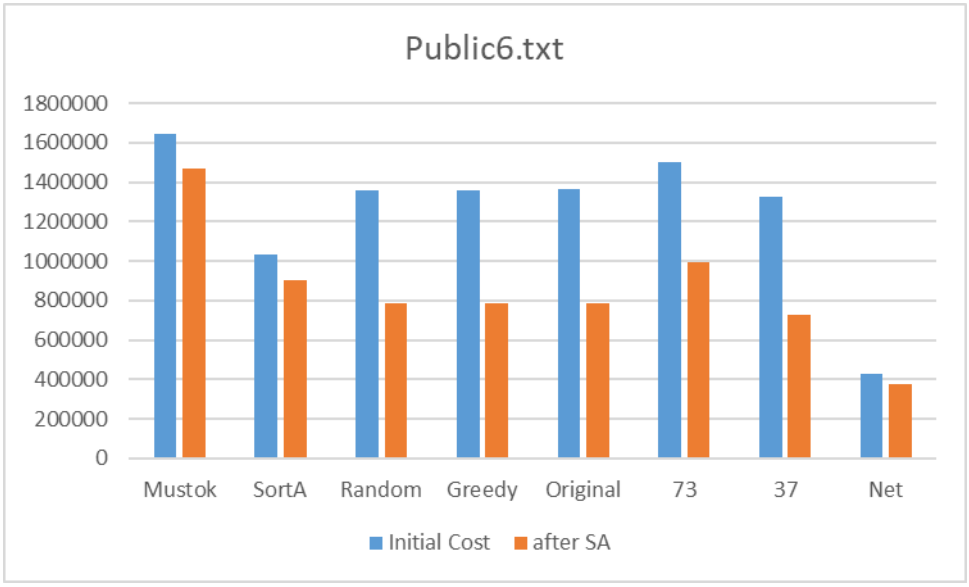



Public2.txt
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
Mustok
SortA
Random
Greedy
Original
73
37
Net
Initial Cost
after SA


Public3.txt
160000
140000
120000
100000
80000
60000
40000
20000
0
Mustok
SortA
Random
Greedy
Original
73
37
Net
Initial Cost
after SA


Public4.txt
600000
500000
400000
300000
200000
100000
0
Mustok
SortA
Random
Greedy
Original
73
37
Net
Initial Cost
after SA





可以看到 initialPartition_net()丟進 SA 的進步有限，但它可以在一開始找到很不錯的 partition，沒有它的話有 public5, 6 的 baseline 我甚至過不了。

6. If you implement parallelization (for algorithm itself), please describe the implementation details and provide some experimental results.
   No, I did not use parallelization.

7. What have you learned from this homework? What problem(s) have

you encountered in this homework?

我學到 Simulated Annealing 演算法的寫法，以前就常常聽到這個依靠機率的演算法，之前就覺得聽起來很有趣，竟然有看運氣的演算法，所以這次才會挑選這個演算法來實作，寫起來其實也很簡單，效果也還不錯，雖然對某些 case 沒有甚麼顯著效果，例如第 5 題圖中的 Mustok partition，可以看出效果很差。

我遇到最大的問題就是 utilization 的部分了，再一開始找到 Mustok 這個 valid partition 數值這麼差後我本以為要重刻演算法了，好在後面試了其他的 partition 得到的還不差，也讓我知道 initial partition 有多麼重要，差的 initial partition 根本救不回來。