CS6135 VLSI Physical Design Automation

Homework 3: Fixed-outline Floorplanning with Fixed and Soft Modules

Name: 賴琮翰

Student ID: 112062542

Date: 2023/11/26

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

Compile

In "HW3/src/", enter the following command:

$ make

An executable file "hw3" will be generated in "HW3/bin/".

If you want to remove it, please enter the following command:

$ make clean

Execute

$ ./hw3 <txt file> <out file>

E.g., in "HW3/bin/", enter the following command:

$ ./hw3 ../testcase/public1.txt ../output/public1.floorplan

Execute example: 以 public1.txt 為例

```
[chlai22@ic51 src]$ make clean
rm ../bin/hw3
[chlai22@ic51 src]$ make
g++ -o ../bin/hw3 -Ofast -std=c++11 main.cpp
[chlai22@ic51 src]$ cd ../
[chlai22@ic51 HW3]$ cd bin/
[chlai22@ic51 bin]$ ./hw3 ../
bin/                HW3_grading/        output/             testcase/
CS6135_HW3_spec.pdf  HW3_printer/        src/                verifier/
[chlai22@ic51 bin]$ ./hw3 ../testcase/publ
public1.txt  public2.txt  public3.txt  public4.txt
[chlai22@ic51 bin]$ ./hw3 ../testcase/public1.txt ../output/public1.floorplan
==============================
Parse the input file
```

[1] Chang, Chang, Wu, and Wu, "B* tree: a new representation for nonslicing floorplans," DAC'00.

2. The wirelength and the runtime of each testcase.

- **multicycle-SA**

		Runtime (ms)			
	HPWL	**input**	**Initial tree**	**SA**	**output**
Public1	187705842	0	0	570002	1
Public2	36337936	1	1	570015	2
Public3	3416431	1	1	570002	4
Public4	110782200	0	0	570006	2

Note: 這邊的做法為在時間(10分鐘，timer設570000ms)內連續跑多次SA，因此SA的執行時間無論testcase大小都會到570000左右。

- **為了觀察bottleneck只做一次SA**

		Runtime (ms)			
	HPWL	**input**	**Initial tree**	**SA**	**output**
Public1	237044430	0	0	41472	103
Public2	34633808	1	1	191647	200
Public3	3755099	1	1	30770	2
Public4	134301325	1	1	11418	1

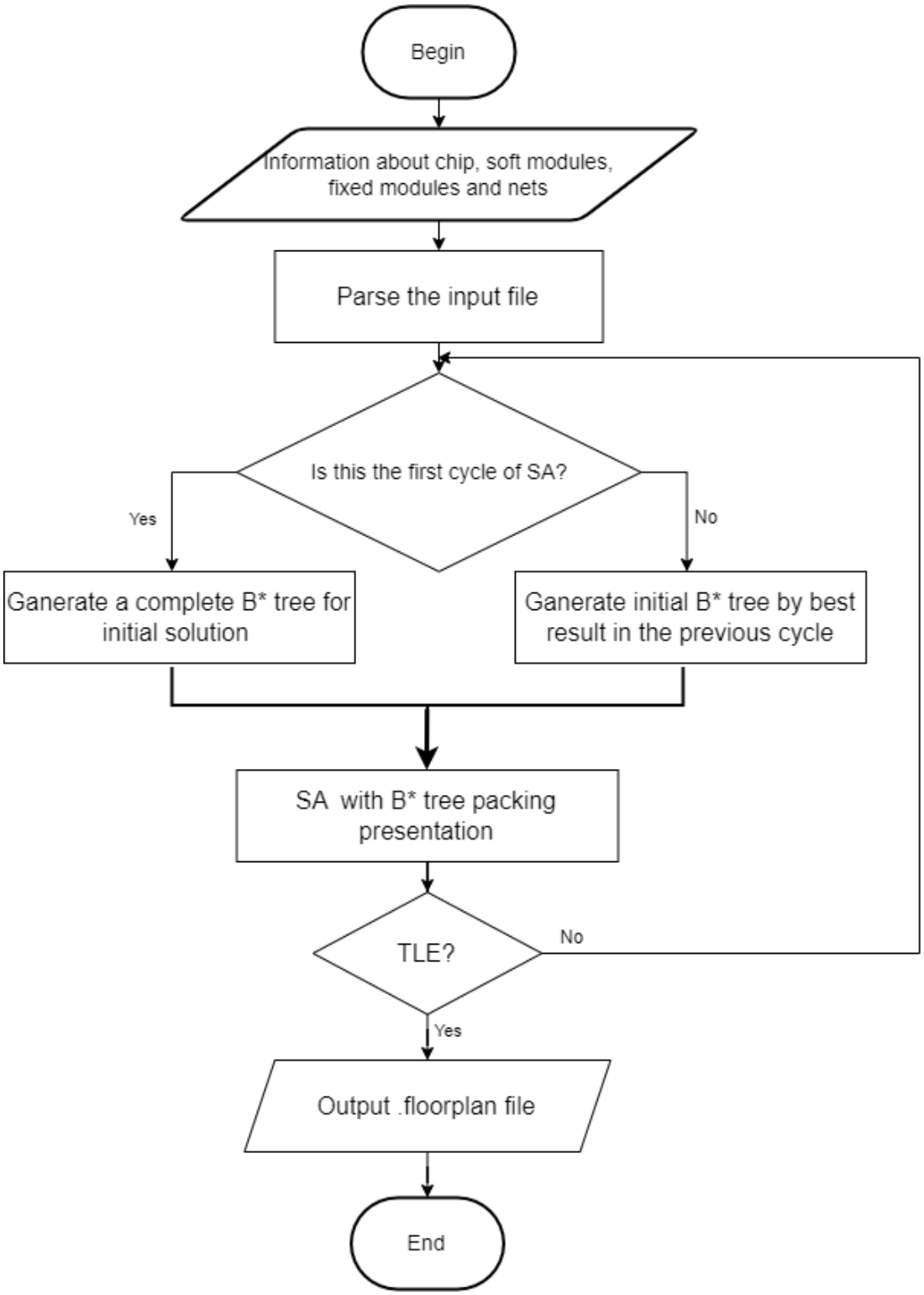
Note: public 2為infeasible solution，能看到花費最長時間的部分為SA。

3. How did you determine the shapes of the soft modules? What are the benefits of your approach?

對於每個Soft Modules題目會給定其最小面積，結合spec上給出的aspect ratio需介在0.5到2之間的條件，可以算出Modules的最小和最大高度，我將可選擇的高度間距切成100等分，在每個高度下利用最小面積算出其寬度，這個長寬即是一種Soft Modules的合法形狀，把所有形狀資訊都存進Soft Modules，當Soft Modules需要決定或改變其形狀時(ex. Initialization, Perturb)，從這些形狀中隨機取一個即可。這樣做的好處第一是易於實現Module形狀改變，只需要亂數取出Module的shape，另一個好處是在產生shape時已經檢查過是符合條件的形狀，不會有因為重新計算長寬而導致超出限制的情況。

[1] Chang, Chang, Wu, and Wu, "B* tree: a new representation for nonslicing floorplans," DAC'00.

4. The details of your algorithm.



在讀取輸入檔案部分，依照前面說明的方式，我會建立每個Soft Modules所有合法的Shapes，並隨機選擇其中一個Shapes作為初始形狀。之後，依照Soft Modules讀入順序，建立一個Complete B* tree供後續SA perturb使用。

[1] Chang, Chang, Wu, and Wu, "B* tree: a new representation for nonslicing floorplans," DAC'00.

此處的SA演算法實作我是參考講義上的SA結合B* tree的那篇論文[1]，其所有的perturb都是對B* tree操作，本次作業中我共寫了4種perturb的方式，分別是

1. 旋轉module 90度: 只要將module的長寬數值交換即可，不用動到B* tree。
2. 改變Soft Modules形狀: 因為已經預先建立好vector儲存Module所有可能的形狀，因此這個操作可透過隨機一個index，並將Modules長寬依對應的shapes設定實現，同樣不改變B* tree結構。
3. 移動Module位置: 移動Modules位置從B* tree的角度去思考，其實是移動B* tree上的一個結點至樹上的任意位置，因為每個節點代表一個Module。更白話的說，此操作便是刪除和插入兩個tree操作的結合。刪除節點我並沒有透過尋找Successor的方式去刪，因為這樣比較複雜，我依據原paper的方法，隨機拿左子樹或右子樹的點往上補，直到葉結點為止。
4. 交換兩Modules位置: 找到兩個B* tree上的節點後，交換即可。

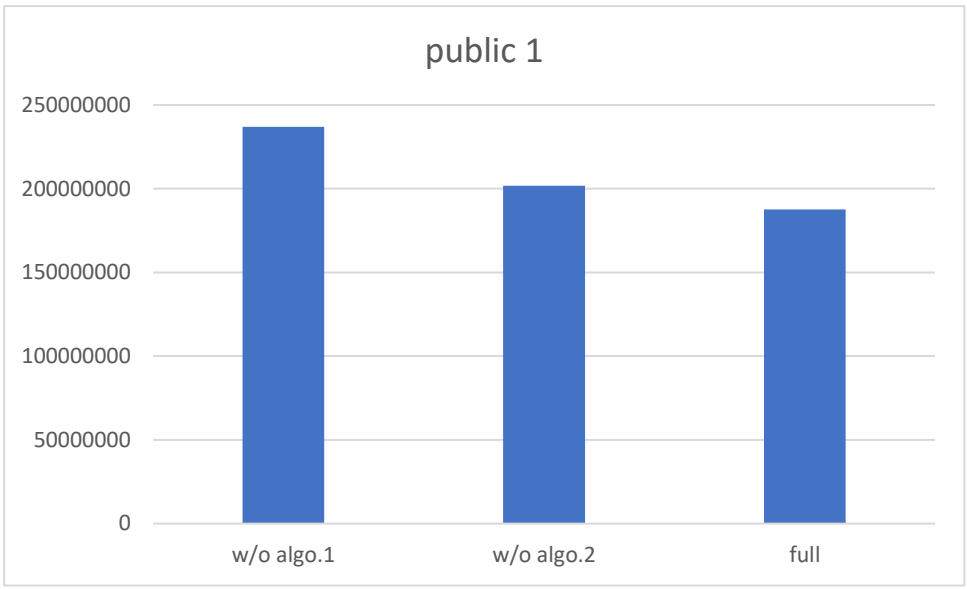
SA主體基本上是參考講義上第21頁的虛擬碼實作出來的，至於超參數的設定則是參考第20頁，再根據case的結果進行微調。

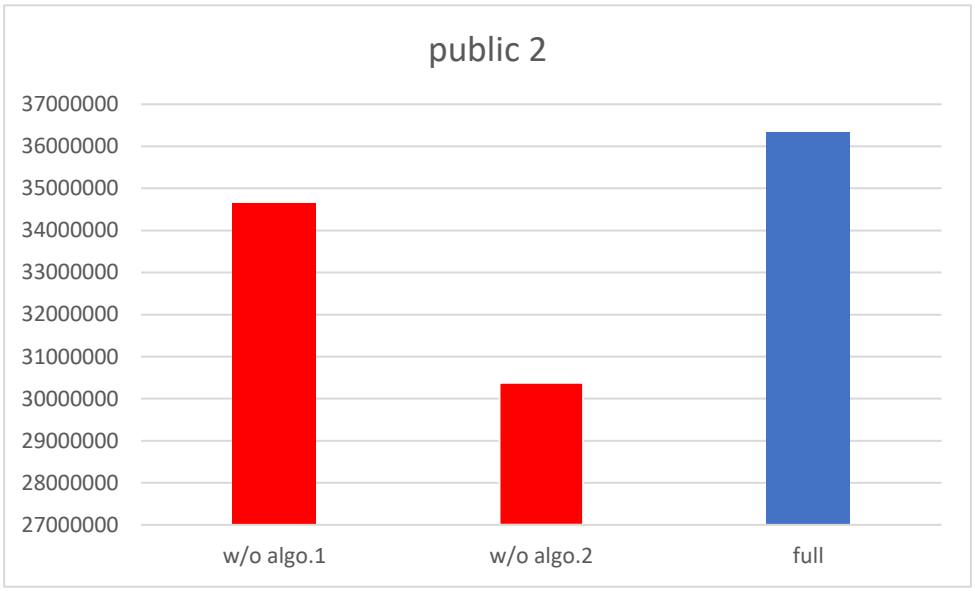
和原paper上的SA只跑一次不同，由於case2經過一次SA後無法收斂至合法解，再加上本次的time limit足足有10分鐘，因此只要時間未達上限，我就會讓SA重新再跑一次，而新的initial solution為上輪SA的最佳解，透過這種方式來找到更好的解。至於cost function我是設定50*(超出outline長寬總合)+HPWL，透過把超出outline的係數調大使SA優先考慮合法解，而若已經跑出合法解，只要跑到下一個Cycle的SA，由於初始解已經會是合法解，其長寬總合為0，此次的SA就會以HPWL為優化目標。

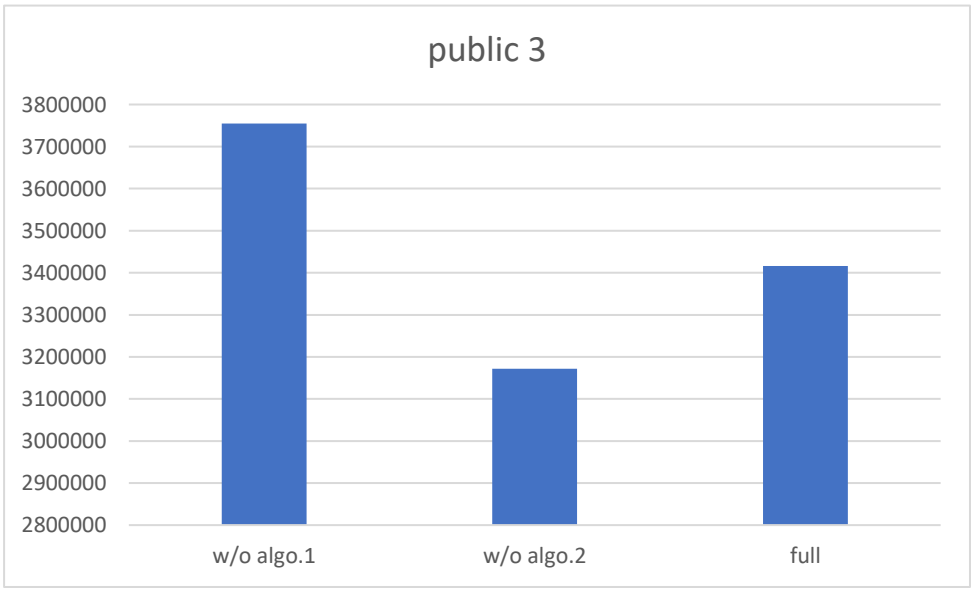
5. What tricks did you use to enhance your solution quality?

我認為有兩個地方對於成績提升的幫助巨大，第一點(algo. 1)是前面提到的多輪SA，不僅讓case2有機會收斂，其他case的HPWL也有所降低，缺點應該是執行時間較長。第二點(algo. 2)我認為是cost function係數的設定，要將超出outline的懲罰調大，才會有較高機會收斂到合法解，因為本次作業還是需要優先找出合法解，因此HPWL應設為次要優化目標。紅色代表infeasible solution。

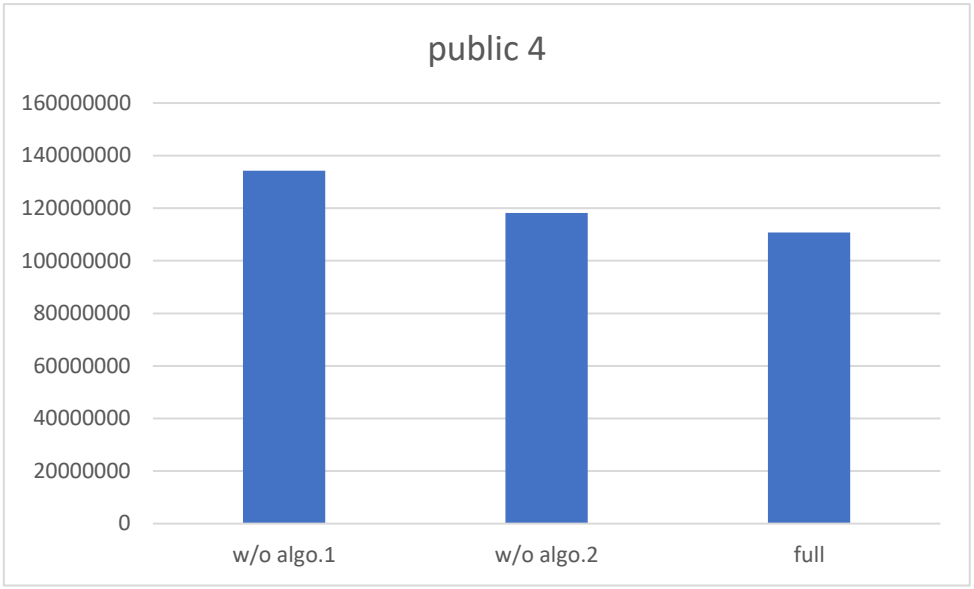
[1] Chang, Chang, Wu, and Wu, "B* tree: a new representation for nonslicing floorplans," DAC'00.







[1] Chang, Chang, Wu, and Wu, "B* tree: a new representation for nonslicing floorplans," DAC'00.



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

本次作業中我學習到如何寫一個好的floorplan演算法，以及SA與B* tree的實作。不得不說雖然上課聽SA感覺概念很簡單，只要跑一些數字就能得出正確答案，但實際實作下來才知道不容易。尤其本次作業需要結合B* tree表示floorplan結果，因此需要開當前狀態的備份當新解不被accept時還原，而B* tree上的一堆指標操作此時便令人非常頭痛，印象中光是寫perturb B* tree裡的move node功能就寫了快兩個小時，還好當時資料結構學得算紮實，慢慢寫下來沒有出甚麼重大的Bug。最後就是要再說一次時間管理的重要，雖然上次作業我也曾經寫過類似的感觸，不過這次因為打算先看完上課影片的緣故，最後只剩下不到4天可以寫，這四天加起來可能睡不到12小時，所以也算學到熬夜趕工的提神方法吧!

問題1: fixed modules如何處理? 這次題目除了fixed outline以外，又多了fixed modules的問題，這讓問題變得很複雜。因為是採用SA，而上課有教過怎麼處理fixed outline，所以一開始我自己的想法是將Fixed modules也合進去cost function內做考量，純靠SA收斂，但我實在想不到有甚麼好辦法能表示fixed modules大小，因此卡非常久。另一方面，fixed modules的存在讓講義上透過B* tree產生floorplan的方式變得無效，經過同學的指點，我才知道原來就直接建一棵B* tree就好，當modules擺放時遇到fixed modules就往上或往右移，這樣可能會擺出邊界，這時候就可以用fixed outline的cost function來處理了。

問題2: 如何實作Move module operation? 在perturb B* tree的四種方法裡，其中一種是將一個module移到其他地方，我思考以後認為應該可以視作將B*的某個node移到另一個位置，這涉及兩種樹的操作: 插入和刪除，前者較簡單，但後者

[1] Chang, Chang, Wu, and Wu, "B* tree: a new representation for nonslicing floorplans," DAC'00.

對於有兩個child的node刪除是困難的，還好回去翻原paper，他們很單純的每次都用left child或是right child補當前被刪除的node，實作起來比較容易一些。

[1] Chang, Chang, Wu, and Wu, "B* tree: a new representation for nonslicing floorplans," DAC'00.