CS6135 VLSI Physical Design Automation

Homework 3: Fixed-outline Floorplan Design

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

➢ Compile

進入 HW3/src/ 資料夾下，輸入以下指令，就會在 HW3/bin/ 資料夾下產生執行檔 hw3 : $ make

➢ Execute

進入 HW3/bin/資料夾下，輸入以下指令:

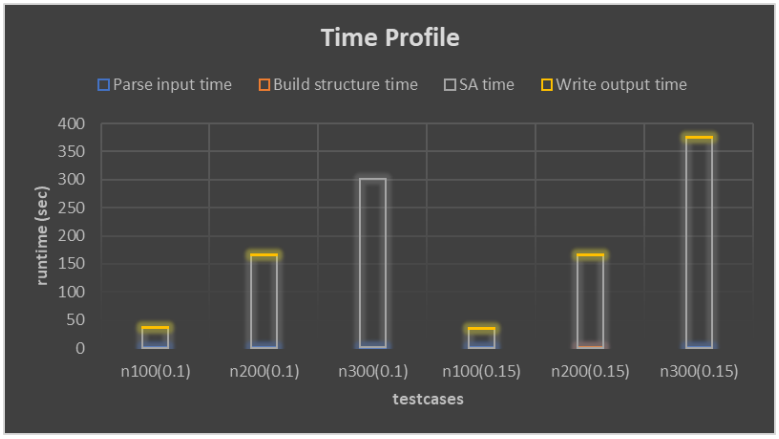
$ ./hw3 <hardblocks file> <nets file> <pin file> <output> <dead space ratio>

e.g. $ ./hw3 ../testcases/n100.hardblocks ../testcases/n100.nets ../testcases/n100.pl ../output/n100.floorplan 0.1

2. The wirelength and the runtime of each testcase with the dead space ratios 0.1 and 0.15, respectively.

Dead Space Ratio	WireLength			Runtime (sec)		
	n100	n200	n300	n100	n200	n300
0.15	192711	351747	476204	34.14	152.53	350.86
0.1	196287	361126	503451	33.54	151.07	351.19

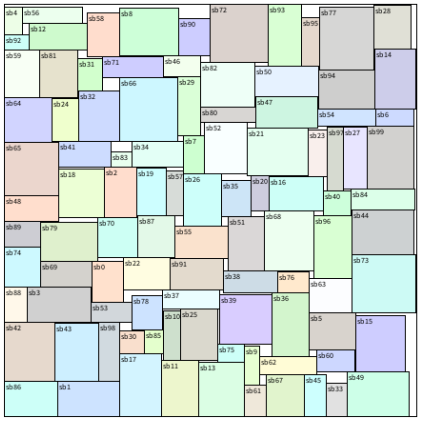
從下圖中的花費時間看到，runtime 幾乎等於 SA time，因此要降低 SA time 的方式就是調整 SA 的參數，以達到一個 wirelength 和 runtime 都不錯的結果，實驗結果如第五點圖表所示。



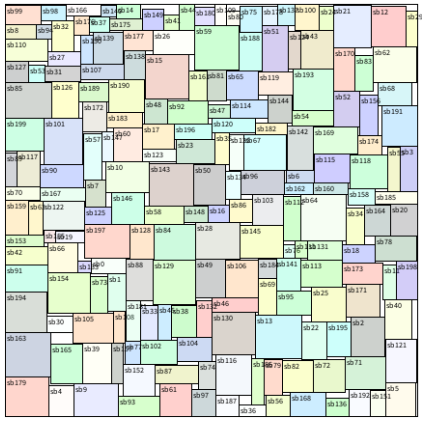
3. **Please show that how small the dead space ratio could be for your program to produce a legal result in 20 minutes.**

	n100	n200	n300
Dead space ratio	0.045	0.038	0.045
Wirelength	223504	437505	596954
Runtime	848.415	838.261	988.5

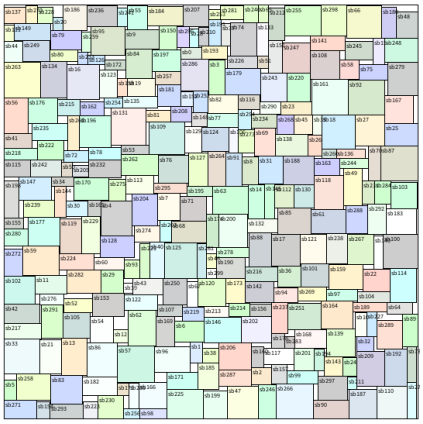
由上表可以看到，三個 case 都可以擺到很密的程度，尤其是 non-slicing floorplan 在 dead space ratio 較小的時候還能夠找到擺得進去的 solution，壓的很密的情況之下也能將 wirelength 下降到蠻好的 performance



n100(dead space ratio = 0.045)



n200(dead space ratio = 0.038)



n300(dead space ratio = 0.045)

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

我所實作的演算法是課程所提到的 B* tree，initial solution 的做法使用相當直覺地一行一行往上排列，意即從最左下角(root)開始擺放，先往右擺(往左子樹)，擺滿過 outline 之後再往上一行繼續擺(root 的右子樹)，依此類推將 module 擺成一個比較好的排列。

而此時的 module 尚未決定他們的座標，因此接著是 contour 的部分，藉由計算當下 contour 來決定每一個 module 的位置，我在這裡使用的是 double linked list 來實作，contour 由當下上邊界最高的 module 連接成一個 double linked list，我的做法是利用 dfs 去尋訪 B* tree，走訪一顆 curNode 時，會先

根據其 parent node 將 curNode 的 x 找出，並判斷 curNode 在 x 方向有和哪一些 contour node 重疊，並將這顆 curNode 的 y 設為重疊的 module 中最高的高度，最後再將這些有被 curNode「完全重疊到」的 contour node 換成 curNode，此時即更新完成 contour node，可以繼續走訪下一顆 node。

其中比較不一樣的是我 perturb 的方式，課程提到的四種分別是 1.rotate a module 2.flip a module 3.move a module to another place 4.swap two modules，考量到這次作業中第二個 flip module 並不影響線長而選擇不實作此 perturbation，另外是第三個 move module 在我的 tree structure 中不好實作且擾動的幅度過小，因此將這個 perturbation 改成直接 swap 兩個 subtree，藉此增加答案的變化性，也較可能更快速的收斂。

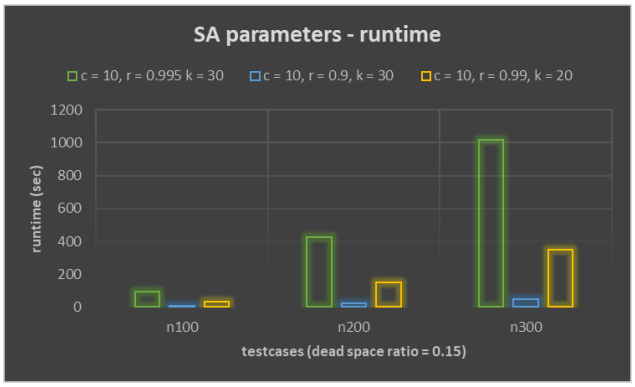
5. What tricks did you do to speed up your program or to enhance your solution quality?

在 SA 的參數上設定的不同以及 cost function 的計算方式大大影響了我的 performance，由於這次作業雖然是要下降 wirelength，但首要條件還是要將 area cost 降到 0 (即擺得進 outline)，一開始我將 cost function 設為 area cost + wirelength，但這樣的作法可能會找到一個 wirelength 不錯但實際上擺步入 outline 的結果，因此我將 area cost 的 weight 調大，也就是說，一旦當下的 floorplan 已經超出 outline 太多則他被接受的機率就會下降許多，這樣的做法除了可以成功的找到 area cost = 0 的答案，也大大降低了 runtime 的大小(收斂速度加快)。

另外一項增加結果 quality 的方式，則是在 initial solution 的擺入順序做 random 的 shuffle，藉由骰很多不同的 random seed 去跑程式，找到一個 wirelength 最好的 seed 用在特殊的 case 上。

我在參數調整的部分做了一些實驗如下圖所示，圖表顯示三種較具代表的參數組合，分別為(1) c = 10, r = 0.995, k = 30 (2) c = 10, r = 0.9, k = 30 (3) c = 10, r =0.99, k = 20，初始溫度 T0 都是 1000。第一個參數組合作出來的 wirelength 是最好的，然而他所花費的時間非常久；而第二個參數組合的時間非常短，幾秒的時間就可以完成 n300 的 case，但其 wirelength 不佳；第三個參數組合是較折中的版本，既可以有不錯的 wirelength 也有不錯的 runtime，因此選擇了這組參數。而第三個參數較好的原因在於溫度下降的速

度(r)足夠慢，且不會在同一個溫度搜尋過久的時間(k)，因此有不錯的結果。



6. **Please compare your results with the top 5 students' results last year for the case where the dead space ratio is set to 0.15, and show your advantage either in runtime or in solution quality. Are your results better than theirs?**

以下是 Dead space ratio = 0.15 的 Ranking:

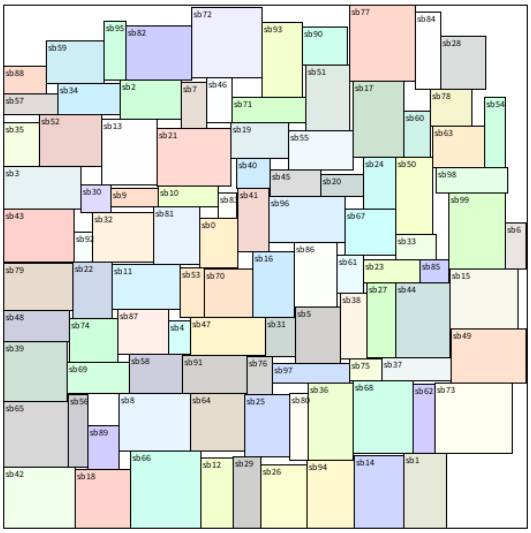
	Wirelength			Runtime(sec)			
Ranks	**n100**	**n200**	**n300**	**n100**	**n200**	**n300**	**Score**
1	200956 [3]	372143 [2]	516906 [3]	24.63 [6]	47.29 [6]	65.81 [6]	42
2	198593 [4]	368731 [3]	535257 [2]	200.25 [3]	308.06 [3]	226.42 [5]	38
3	194369 [5]	354107 [5]	491069 [5]	385.75 [1]	709.61 [1]	926.55 [1]	48
4	204001 [2]	367298 [4]	499733 [4]	330.42 [2]	576.15 [2]	793.26 [2]	36
5	208575 [1]	378187 [1]	567794 [1]	26.72 [5]	120.73 [5]	247.22 [4]	23
my	**192711 [6]**	**351747 [6]**	**476204[6]**	**35.21 [4]**	**166.24 [4]**	**376.61[3]**	**65**

上表為 performance 排名計算，score = wirelength * 3 + runtime * 1，可以從中看到，經過調整 cost function 以及找出最好的 initial 擺放順序，在已知的 testcase 有很不錯的成績，

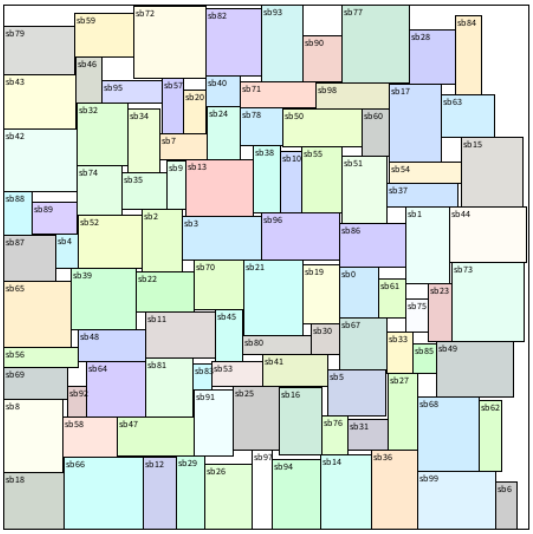
由於這次評分方式有將 runtime 列入考量，因此我花了點時間找出一個比較好的 SA 參數，調整到一個可以蠻快收斂的 annealing schedule，並且也有不錯的答案，從上表的比較來看的話，我覺得我在 wirelength 和 runtime 的表現都蠻不錯的，尤其在 300 個 module 的情況也在 300 多秒完成，我認為自己在這個 Rank 中可以排在第一名的位置。

7. Floorplan Result

➢ n100

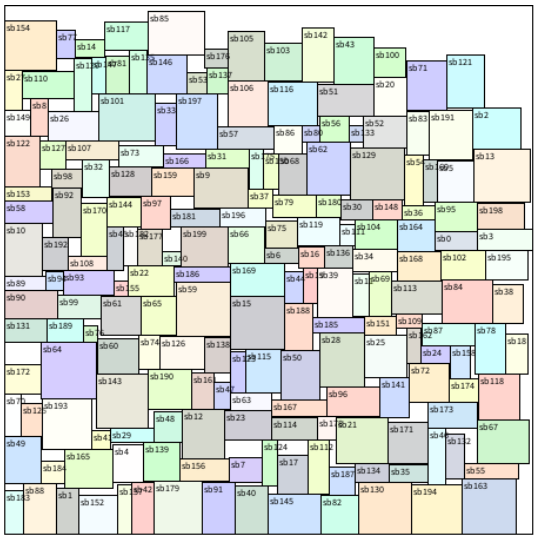


dead space ratio = 0.15

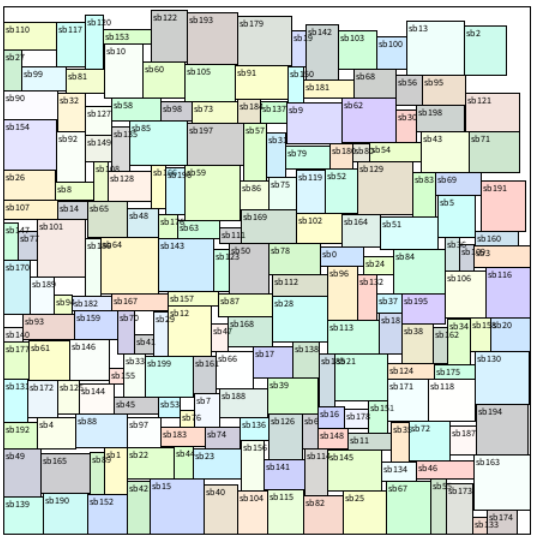


dead space ratio = 0.1

➢ n200

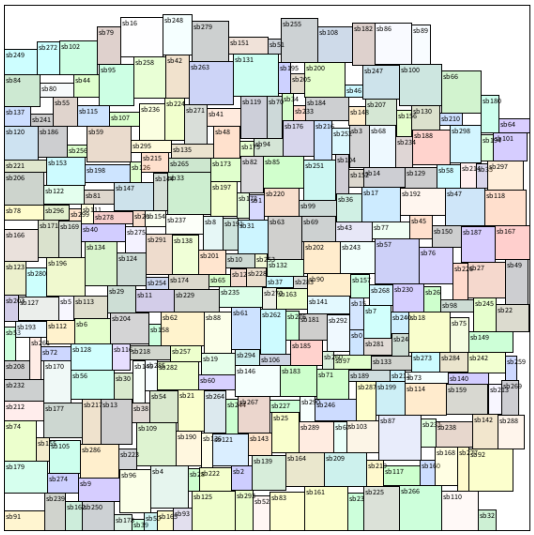


dead space ratio = 0.15

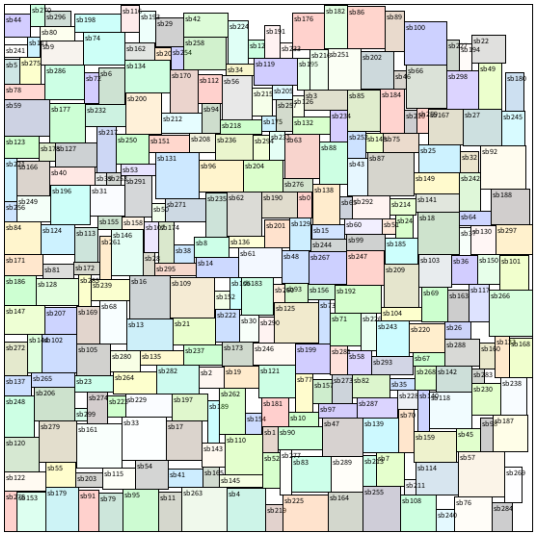


dead space ratio = 0.1

➢ n300



dead space ratio = 0.15



dead space ratio = 0.1