

VLSI Physical Design Automation HW3

王領崧 107062107

P.S. 執行和測試數據都是在 ic51 上面

2) How to compile and execute your program

– Compile:

”./src” 資料夾輸入 make，即可 compile，生成執行檔 ”hw3” 在 ”/bin” 資料夾裡面。

e.g.

\$ make

```
g++ ./main.cpp Reader.cpp Floorplan.cpp -std=c++11 -O3 --optimize -o ../bin/hw3
```

– Execute:

在 ”./bin” 資料夾中，用 ./<exe> <hardblock file> <net file> <pl file> <output floorplan file> dead_space_ratio 格式輸入，即可執行。

e.g.

```
./hw3 ../testcases/n100.hardblocks ../testcases/n100.nets ../testcases/n100.pl ../output/n100.floorplan 0.15
```

P.S 助教不好意思 底下 3-1, 3-2 的數據都是在沒有 -O3 和 --optimize 跑的，所以 runtime 會和最後助教的 hw3 executable 所跑出的執行結果有所差異，但各個 stage 時間佔比是相同的。

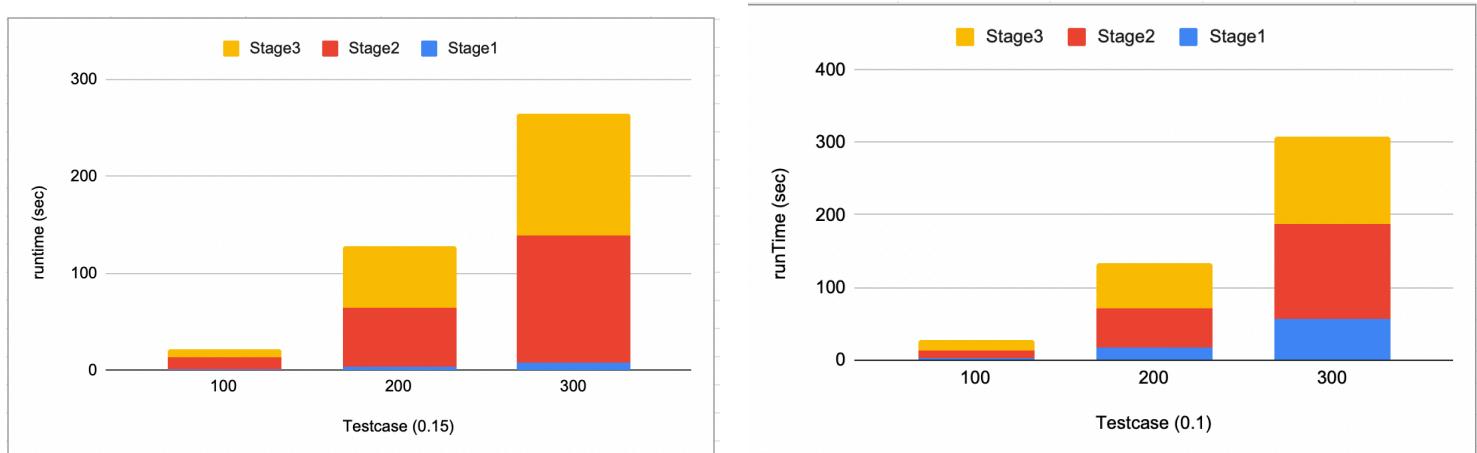
3-1) The wirelength and the runtime (sec) of each testcase with dead space ratio 0.15

	n100	n200	n300
Seed	-179645290	-1221175410	1420872313
Wirelength	201294	363039	490731
Total Runtime	22.251	128.231	265.14
I/O time	0.001	0.001	0.01
SA time (stage1)	0.31	3.96	7.62
SA time (stage2)	13.13	59.9	131.22
SA time (stage3)	8.81	64.37	126.29

3-2) The wirelength and the runtime (sec) of each testcase with dead space ratio 0.1

	n100	n200	n300

Seed	-1701349740	-21782245	-1296314180
Wirelength	208687	372548	505299
Total Runtime	27.021	133.621	308.46
I/O time	0.001	0.001	0.001
SA time (stage1)	1.99	16.37	56.05
SA time (stage2)	10.04	54.71	130.95
SA time (stage3)	14.99	62.54	121.46



從圖表可以清楚地看到 sa 的 Stage2, Stage3 的執行時間最久，並且總執行時間隨著 testcase 的增大而變長。兩點都是很合理的，因為 Stage2, Stage3 為主要尋找最小 wirelength 的地方；而 testcase 越大，NPE (normal polish expression) 也會越大，SA 的 N 也會越大，calculate cost 要計算的 hardblocks 數量自然也會越多，因此 total execution time 也會越長。

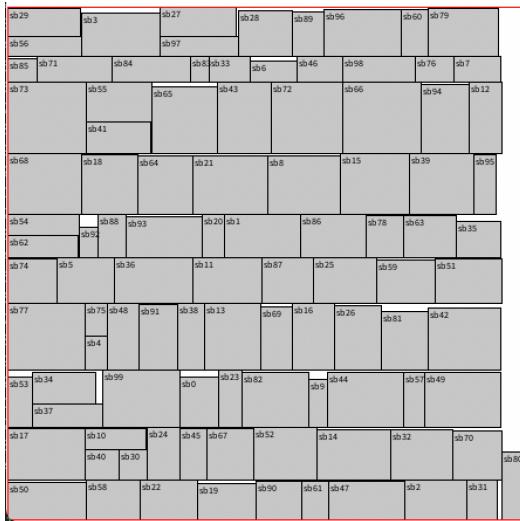
(I/O time 因為佔比太少就沒有放到圖表上面。sa Stage1, 2, 3 在第 6. 點有更詳盡的解釋)

- 4) show that how small the dead space ratio could be for your program to produce a legal result

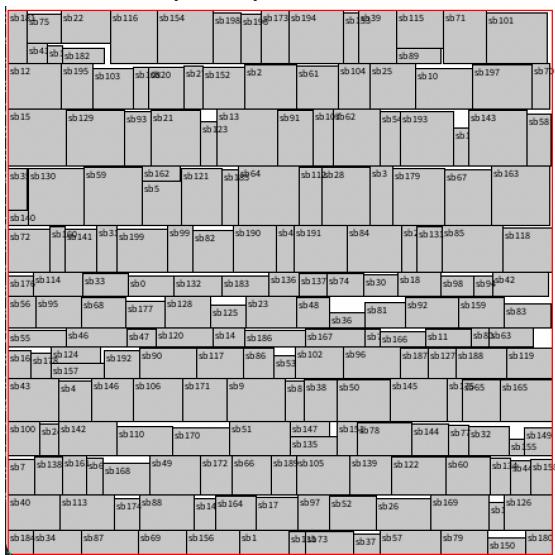
	n100	n200	n300
DeadSpaceRatio	0.07	0.055	0.065
Total Runtime (sec)	146.18	779.87	769.21

(Printer 在下一頁)

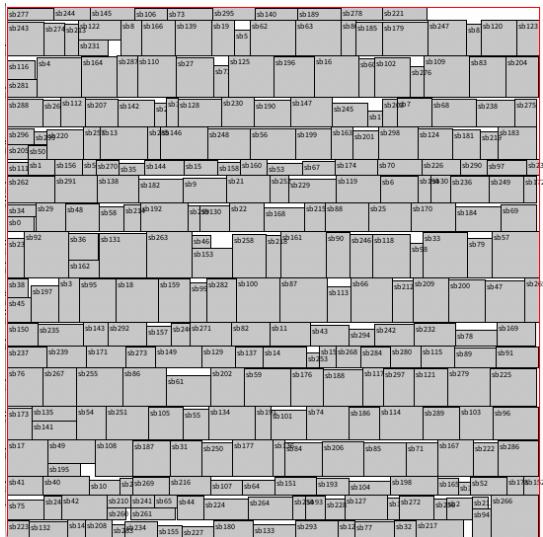
n100 floorplan printer



n200 floorplan printer



n300 floorplan printer



5) The details of your algorithm

此次實作的方法使用講義上的 Slicing Floorplan Design by Simulated Annealing。

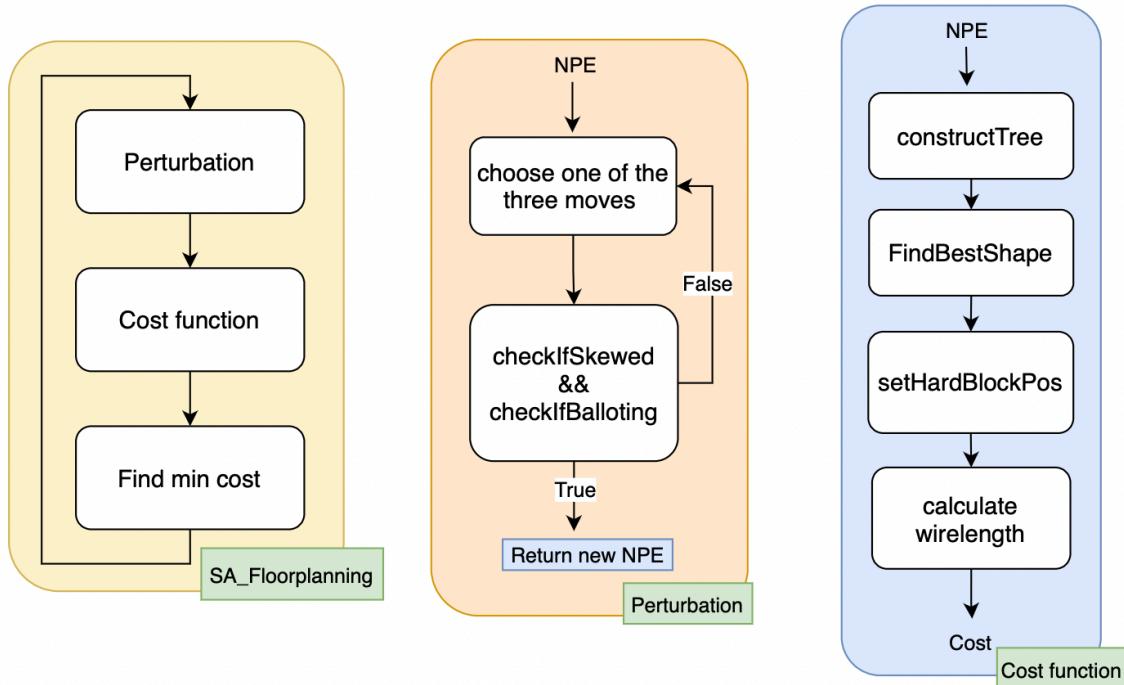
主要可以分成 Initial 和 SA 兩個部分。

Initial: 使用所有的 hardblock 擺出初始的 NPE。雖然講義上是寫 1V2V3V4V...，但是實作後發現，放幾個 hardblocks 後，就會超出 fixed area 的 width，所以如果用講義的方式當作 initial status，需要跑很多 iteration 的 perturbation 才能放進 fixed area 中，很浪費時間。因此後面改成如果同一排的 width 已經超過 fixed area 的 width，那就往上一排擺（也就是多一個 H），這樣 initial 的結果只會在 height 有些微 violate 的情況，能大大減少排進 fixed area 所需的 perturbation 次數。

SA: 與講義上的 pseudo code 相同，重複 perturbation + cost function 的循環（左）。

Perturbation 會先隨機一種 move，根據每一種 move 記錄下所有符合的 position，然後隨機選出 position 進行 move。Type 3 move 需要多判斷會不會違反 skewed & balloting。如果不會的話就執行並回傳新的 NPE。反之就重新選擇 position，再做一次（中）。

Cost function 則會先根據 NPE 建造出一個 tree，建造 tree 的同時，也要不斷更新每個 node 所有可能的 possible shape (講義 stockmeyer 那一頁)。接著從 root 的 possible shape 中找出最好的 case，並且根據這個 case 去擺放每一個 hardblocks 的座標位置，最後再計算 area & wirelength 就得到此次 floorplan 的 cost (右)。



6) What tricks did you do to speed up your program or to enhance your solution quality?

- 講義的 initial solution 為 12V3V4V...，然而 fixed area 為正方形的區域，單放一排一下子就會超出範圍，導致需要耗費很多輪的 perturbation 才能將 hardblocks 擺進 fixed area 內，甚至怎麼調都擺不進去。因此，我改成疊積木的概念，從最底下開始擺，當一排的 width 超過 fixed area width 時，就會往上堆。這樣完成後雖然也無法保證會在 fixed area 中，但是只會在 height 有些微超出的情況，並且能大大降低 perturbation 到 fixed area 的次數。
- 講義上的 cost 會把 wirelength + weight * area，然而經過實驗後，發現 wirelength 常常會 dominate cost，使得產出最好的結果根本沒有擺進去 fixed area 內。所以捨棄講義只跑一次 SA 的作法，將 SA 分為兩個階段，第一階段為尋找 fixed area 的 floorplan，並且找到符合要求的 floorplan 就停止。第二階段為在 fixed area 藉由 perturbation 尋找最短的 wirelength，原本是三種 type 的 move 都可以使用，但是實驗後發現由於 type2, type3 會改變 slicing-structure，導致新的結果有蠻大的機率會超出 fixed area，因此為了兼顧 runtime & wirelength 後，最後選擇單做 type 1 的 move 而已。
- Perturbation 的 type 1 move (swap adjacency operands) 有規定是交換相鄰的 hardblocks，然而我覺得這樣對 floorplan 的變化性太低，畢竟我的 initial solution 是依序從 sb0 – sb99 去放置，所以我分成兩個階段 (兩次的 SA) 執行，第一階段是任兩個 hardblocks 都可以互換，以此增加變化，在 fixed area 內找出最短的 wirelength。由於第一階段已經花相當大的 effort 找最短的 wirelength floorplan，基本上 wirelength 能進步的幅度已變得非常小，所以第二階段變成 refinement 的工作，只調整 adjacency 的 hardblocks (與講義相同)，看看能不能找到更好的結果。

n100, 0.15	上課	我的
Wirelength	296250	201294
Total Runtime	36.01	22.251

7) compare your results with the top 5 students

Ranks	Wirelength			Runtime(s)		
	n100	n200	n300	n100	n200	n300
1	200956	372143	516906	<u>24.63</u>	<u>47.29</u>	<u>65.81</u>
2	198593	368731	535257	200.25	308.06	226.42
3	<u>194369</u>	<u>354107</u>	<u>491069</u>	385.75	709.61	926.55
4	204001	367298	499733	330.42	576.15	793.26
5	208575	378187	567794	26.72	120.73	247.22

	n100	n200	n300
--	------	------	------

Wirelength	201294	363039	490731
Total Runtime	22.251	128.231	265.14

n100: wirelength(第4), runtime(第1)

n200: wirelength(第2), runtime(第3)

n300: wirelength(第1), runtime(第4)

綜觀來看，我覺得我比第 3 名還好，跟第 2 名各有好壞。

雖然第 3 名的同學 wirelength 除了 n300 的比我好，但是他的 runtime 太過糟糕，是我的 3–10 倍不等，因此如果要同時考量 wirelength & runtime 的話，我覺得我是優於他的。

第 2 名同學儘管只有 n100 的 wirelength 比我好，但是他三個 testcase 的 runtime 都很平均，並沒有隨著 testcase 的增長而變大，相反地，我的 runtime 是越來越大，因此我覺得我們兩個各有優缺點。

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

1. 讀取 input file

此次的檔案不像 HW2 格式那麼整齊，除了有一些不必要的資訊，像是 NumHardRectilinearBlocks、NumTerminals，可以藉由 load 完所有的 blocks、terminals 就能得知外，還有冒號 (:)、以及四角座標那邊的 (x,y) 要處理。因此一開始的實作其實是靠某些單字來分辨此排的資訊，然後宣告很多 string, int 來接取所有對應的字符，最後再儲存需要的資訊，然而這樣實作上相當麻煩，並且如果輸入上有許多一個空格、或是奇怪的字元，就會發生錯誤，程式容錯的機會相當低。

後面聽取學長的建議後，改成先使用 getline() 將整行都吃下來，然後利用 sscanf 支援正規表示法來處理四角座標的問題，以及 stringstream 讀取不同 type 的 data。不但 code 變得十分簡潔易懂，也多了一些容錯的空間。讓我對 C++ 讀取 file 能用的函數和寫法又更加了解。

2. SA process 的 parameter, cost function 計算方式

上課的時候就有聽老師說過 SA process 中各個參數的調整、cost function weight 設定十分重要。實作過 HW3 後，充分體會到這些參數的重要性。一開始我將 cost function 中 area 的 weight 設定很低，結果發現 cost 很容易被 wirelength dominate，因此有時候會出現擺不進的組合卻是最好的結果。在 SA 的 parameter 也是同樣的道理，要讓溫度降的多快、降到多少，並且接受 uphill 的機率，都會影響最後 output 的 quality。不過我覺得這邊除了思考外，更多的 effort 是要花時間做實驗，找出最好的參數搭配。

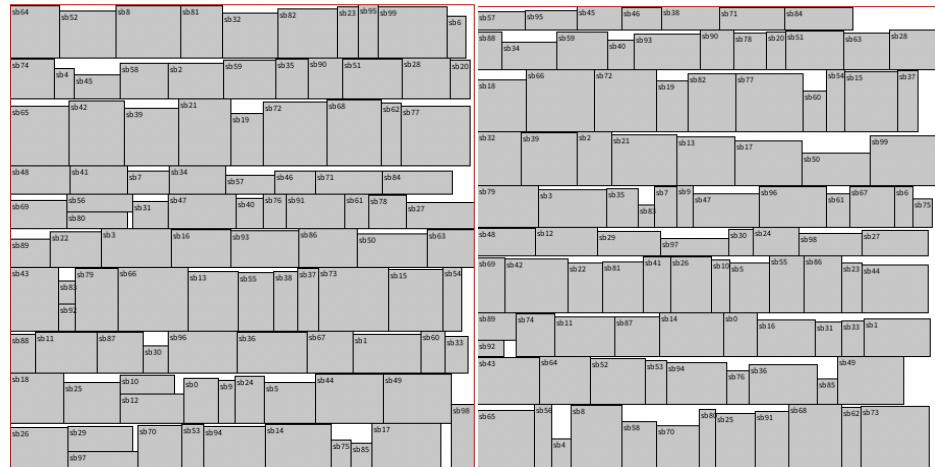
3. STL 的使用更熟練

由於 HW3 的實作是採用 slicing-structure 的 floorplan 下去實作，因此需要用到很多 data structure 來儲存資料，除了 HW2 用到的 vector, unordered_map 之外，

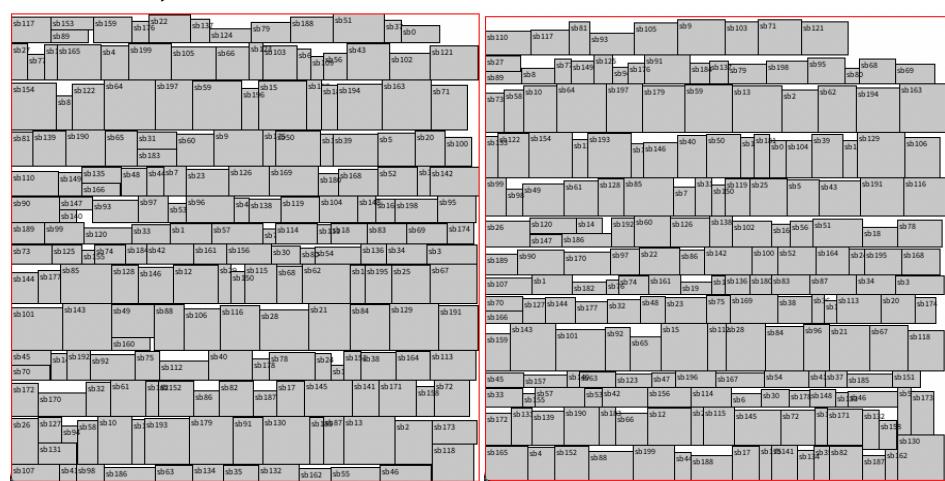
還有解析 postorder 使用到的 stack，以及 build tree 要用到的 Node，和其他自己創造的各種 class。經過這兩次作業的練習，對於這些 STL 的使用上更加熟練，也希望在寫 final project 上能有幫助！

9) Best Result printer (因為有 6 張圖所以放到這裡來)

n100 (0.1, 0.15)



n200 (0.1, 0.15)



n300 (0.1, 0.15)

