CS6135 VLSI Physical Design Automation

Homework 3: Fixed-outline Floorplan Design

Student id:110033638 **Name:林哲宇**

I. Compile and execute

File:

header.h
structure.h
structure.cpp
main.cpp

--How to Compile

In this directory, enter the following command:

$ make

It will generate the executable file "hw3" in "HW3/bin/".

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

$ make clean

--How to Run

In this directory, enter the following command:

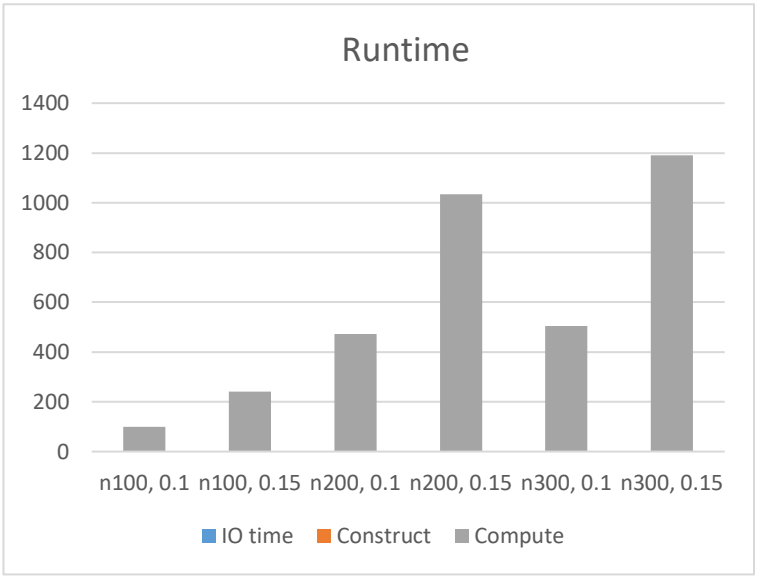
$ ../bin/<exe> <testcase.blocks> <testcase.nets> <testcase.pl> <testcase.floorplan> <dead_space_ratio>

e.g.:

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

II. Runtime

Testcase	IO time	Construct	Compute	Total time	Wirelength
n100, 0.1	0.007	0.46	98	98.96	288114
n100, 0.15	0.007	0.46	240	241.33	269328
n200, 0.1	0.016	0.9	472	473.02	493385
n200, 0.15	0.016	0.9	1033	1034.92	269328
n300, 0.1	0.021	1.125	504	505.70	728204
n300, 0.15	0.021	1.125	1190	1193.02	696410



Population of B*- tree are generate in construct time. With larger number of modules will cost more time in cinstruct time. And compute time are much larger than IO time and construct time.

III. Dead space ratio

Testcase	Smallest dead space ratio
n100	0.1
n200	0.1
n300	0.15

IV. My Algorithm

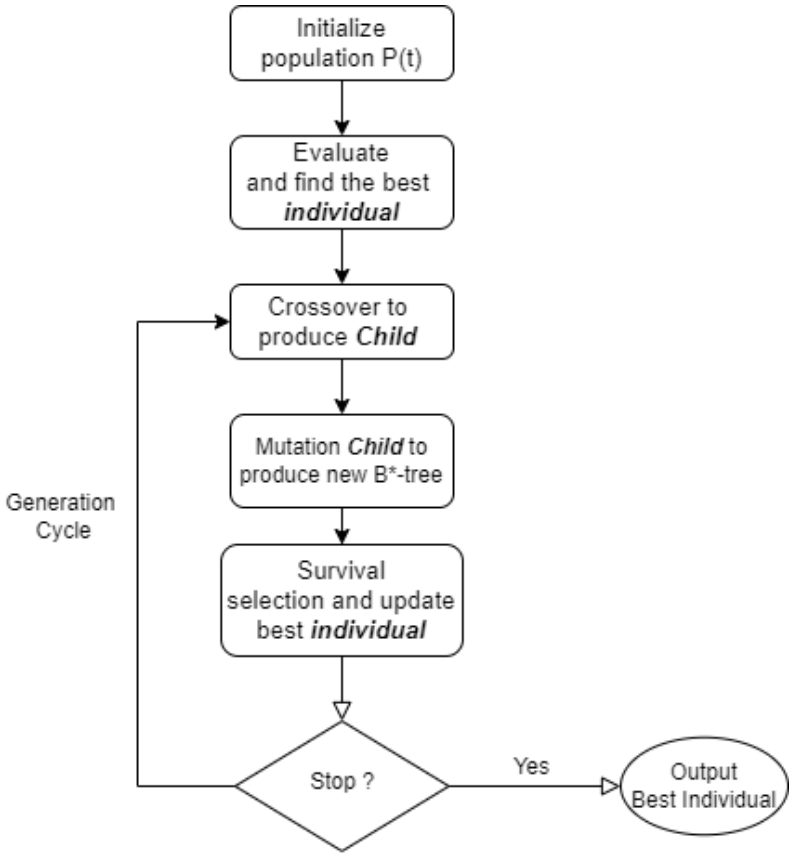
Using representation B*-tree with genetic algorithm. The fitness function is like the cost function in simulated annealing talked in the lecture. And I also add the penalty function in fitness function to handle the fixed outline constraint.

A **genetic algorithm** is a search heuristic that is inspired by Charles Darwin's theory of natural evolution. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction to produce offspring of the next generation.

Five phases are considered in a genetic algorithm:

1. Initial population
2. Fitness function
3. Selection
4. Crossover
5. Mutation

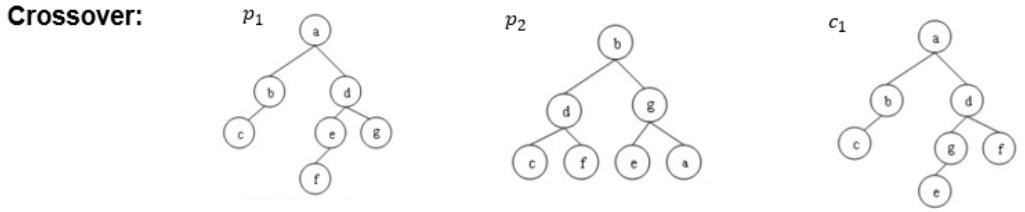
Genetic Algorithm Flow chart



Crossover Operator

Applied on two parents that are selected to generate an offspring. The generated floorplan inherits some characteristics from both its parents in a way like natural evolution.

Left Subtree Crossover	Replace p1 right subtree by p2



Mutaion operator

Rotate module	Rotate degree of 90
Move module	Move module to another place
Swap two module	Swap two module
Rotate branch	Swap select node left and right chile

The advanrage of Genetic Algorithm is that the solution can jump out of the local optimum. Because initial population are generated by random global search.

Fitness Function

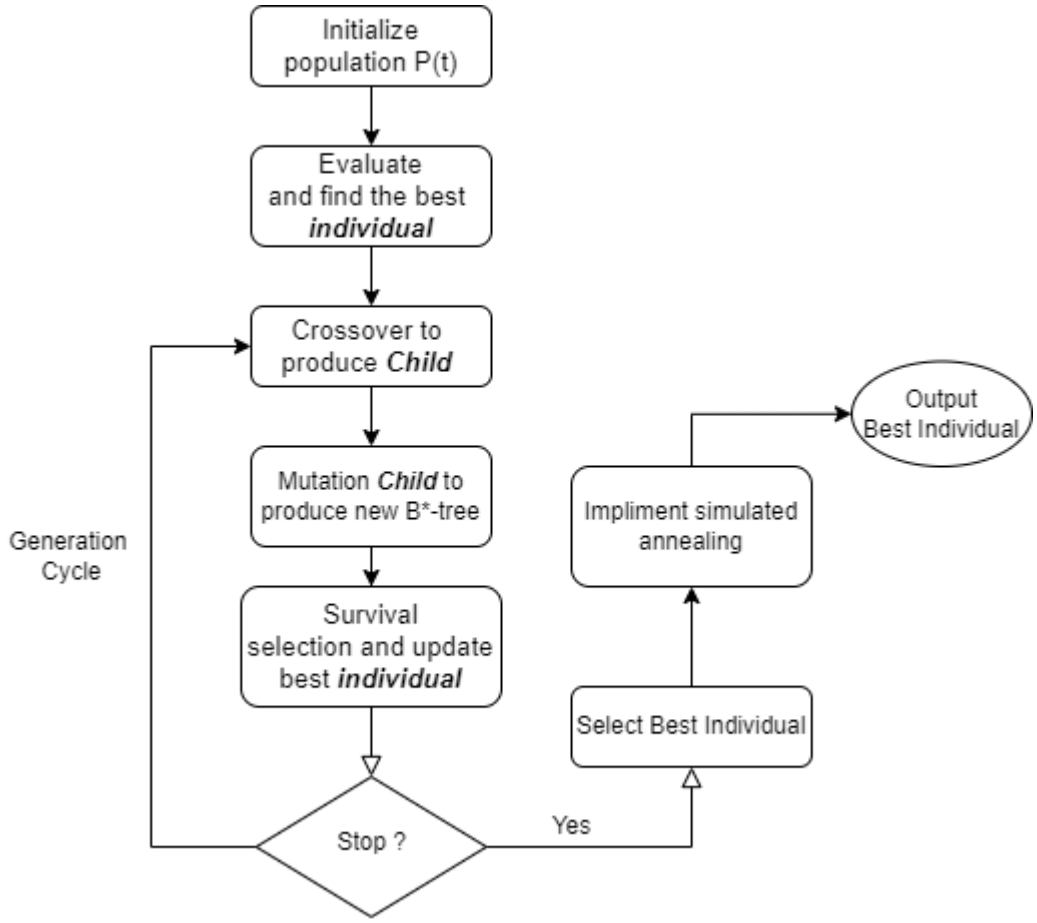
$$\alpha * \frac{Area}{Area_{outline}} + \beta * \frac{Total\ Wirelength}{Wirelength_{avg}} + \delta * (aspect\ ratio)^2 + \gamma * Penalty$$

Where $\alpha, \beta, \delta, \gamma$ are user specify.

V. Speed up strategy

由於 Genetic Algorithm 收斂速度比 Simulated annealing 慢許多，單相對比 Simulated annealing，但是因為 Genetic Algorithm 有 perform random global search，因此更容易找到 global optimal，為了加速整個流程，利用 two stage 的方式，先在 stage 1 實行一般 Genetic Algorithm，找到 rough floorplan 之後，進入 stage 2 實行 Simulated annealing，利用 perturb 進行 floorplan 的 local search。以加快收斂速度。

Two stage Genetic Algorithm Flow chart



VI. Result of each testcase

Classical Simulated Annealing

```
grading on 110033638:
  testcase |      ratio | wirelength |    runtime | status
      n100 |       0.15 |     277116 |     237.65 | success
      n200 |       0.15 |     500079 |     461.94 | success
      n300 |       0.15 |     696178 |     533.09 | success
      n100 |        0.1 |     256634 |     566.77 | success
      n200 |        0.1 | verifier: verifier.cpp:142: void Floorplan::Verify(): Assertion `bottom>=0&&top<=whTier' failed.
      fail |    1144.44 | n200 has some cells out of outline.
      n300 |        0.1 | verifier: verifier.cpp:142: void Floorplan::Verify(): Assertion `bottom>=0&&top<=whTier' failed.
      fail |    1142.94 | n300 has some cells out of outline.
```

Testcase	n100	n200	n300
Wirelength	277116	500079	691678
Runtime	237.65	461.94	533.09

Classical Genetic Algorithm

```
grading on 110033638:
  testcase |      ratio | wirelength |    runtime | status
      n100 |       0.15 |     270157 |     290.40 | success
      n200 |       0.15 |     478470 |     545.65 | success
      n300 |       0.15 |     674302 |    1024.50 | success
      n100 |        0.1 |     247612 |     440.50 | success
      n200 |        0.1 | verifier: verifier.cpp:142: void Floorplan::Verify(): Assertion `bottom>=0&&top<=whTier' failed.
      fail |    1143.98 | n200 has some cells out of outline.
      n300 |        0.1 | verifier: verifier.cpp:142: void Floorplan::Verify(): Assertion `bottom>=0&&top<=whTier' failed.
      fail |    1147.96 | n300 has some cells out of outline.
|-------------------------------------------------|
|                                                 |
```

Testcase	n100	n200	n300
Wirelength	270157	478470	674302
Runtime	290.40	545.65	1024.5

Two stage Genetic Algorithm

```
grading on 110033638:
  testcase |      ratio | wirelength |    runtime | status
      n100 |       0.15 |     288114 |      98.96 | success
      n200 |       0.15 |     493385 |     473.02 | success
      n300 |       0.15 |     728204 |     505.70 | success
      n100 |        0.1 |     269328 |     241.33 | success
      n200 |        0.1 |     460421 |    1034.92 | success
      n300 |        0.1 | verifier: verifier.cpp:142: void Floorplan::Verify(): Assertion `bottom>=0&&top<=whTier' failed.
      fail |    1193.02 | n300 has some cells out of outline.
```

```
grading on 110033638:
  testcase |      ratio | wirelength |    runtime | status
      n100 |       0.15 |     274606 |     186.85 | success
      n200 |       0.15 |     474735 |     541.78 | success
      n300 |       0.15 |     696410 |     623.51 | success
      n100 |        0.1 |     274755 |     427.54 | success
      n200 |        0.1 | verifier: verifier.cpp:142: void Floorplan::Verify(): Assertion `bottom>=0&&top<=whTier' failed.
      fail |    1193.12 | n200 has some cells out of outline.
      n300 |        0.1 | verifier: verifier.cpp:142: void Floorplan::Verify(): Assertion `bottom>=0&&top<=whTier' failed.
      fail |    1192.99 | n300 has some cells out of outline.
```

Testcase	n100	n200	n300
Wirelength	288114	493385	728204
Runtime	98.96	473.02	505.70

Two stage Genetic Algorithm speed up efficient with the larger number of hardblocks.

VII. Parallelization

本次作業並無使用平行化

VIII. Compare

在這次的比較上，不論是 runtime 或是 wirelength，均沒有比去年前五名好，由於這次我認為最困難的地方在於擺進 fixed-outline，並且在 deadspace ratio 0.1 的狀況下，n200 及 n300 是有滿大的機會超出 fix-outline，將 wirelength 的權重放到最小，導致最後解果的 wirelenth 都沒有很好。而速度也是其中一個原因，由於計算過程中不斷會用到 trasverse tree，以及利用 map 來計算 wirelength，而利用 map 來搜尋 element 的速度相對較慢，而未來能改進的地方為改良或是增加 operators 以增加每個 B*-tree 的 diversity。

Top 5 students' results last year (dead space ratio = 0.15)

	Wirelength			Runtime(s)		
Ranks	**n100**	**n200**	**n300**	**n100**	**n200**	**n300**
1	200956	372143	516906	**24.63**	**47.29**	**65.81**
2	198593	368731	535257	200.25	308.06	226.42
3	**194369**	**354107**	**491069**	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

IX. Learn from homework

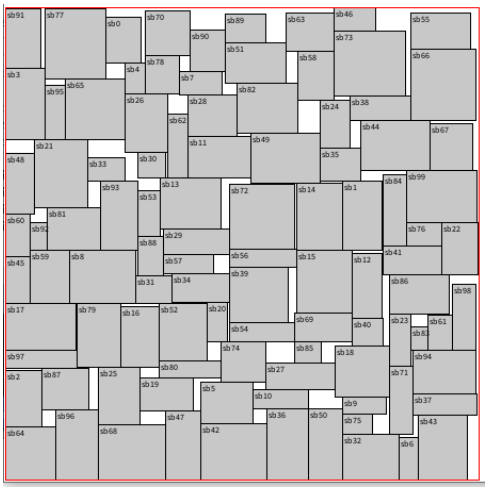
這次作業花費許多心力在完成，最困難的是在於擺進 fix-outline，由於這次有諸多地方使用 random，並且不像之前作業有給定演算法，這次整個架構都需要自己研究設計，但相信大部分的同學都是實作 B*-tree with Simulated annealing，由於我碩士研究方向專攻在 Evolutionary Computing 的領域，因此我使用了 Genetic algorithm 作為我的主要演算法去設計，結合 Genetic algorithm 及 Simulated annealing，改良成新的演算法。雖然在做 grading 時仍有一部分機率使得 floorplan 超出 fixed outline，但是仍可看出擺出來的樣子，改良的演算法是有效果的。

另一個困難的地方在於每次實驗都會經過很長時間，導致在最後的實驗都必須要等很久才能在實驗下一組參數。

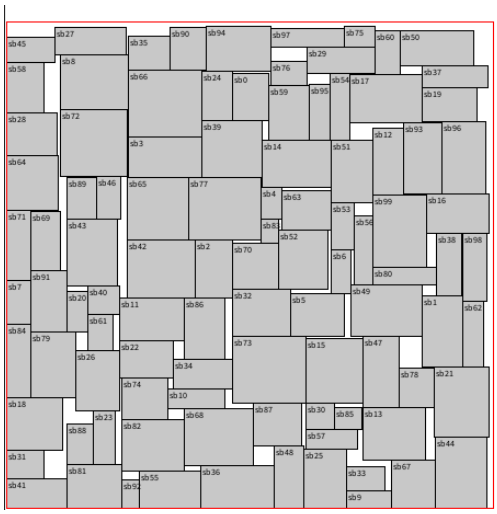
而這次作業中學到了，B*-tree 的基本架構，及了解 Simulated annealing 演算法和做 floorplanning 的步驟，並且也更熟悉 C++語法，收穫良多。

X. Floorplan graph of each testcase

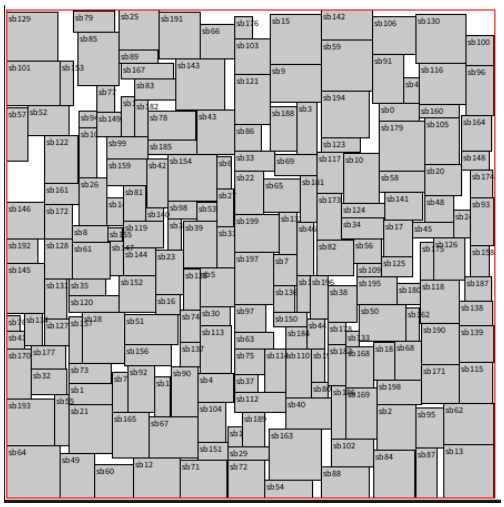
n100 with dead space ratio 0.1



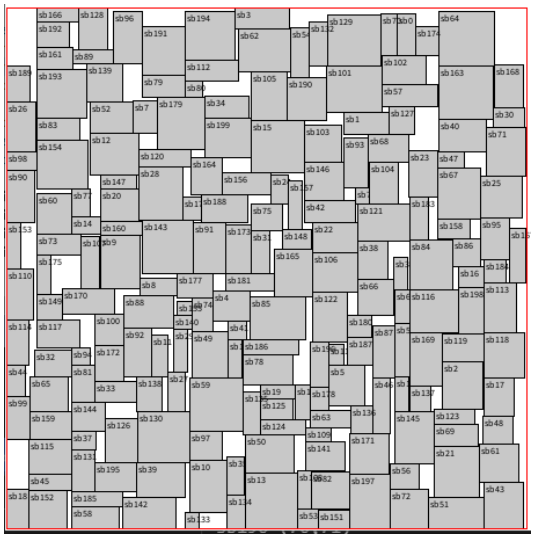
n100 with dead space ratio 0.15



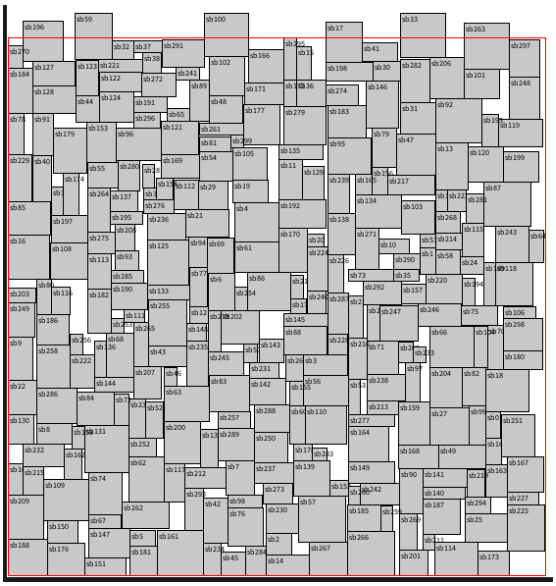
n200 with dead space ratio 0.1



n200 with dead space ratio 0.15



n300 with dead space ratio 0.1



n300 with dead space ratio 0.15