CS6135_HW3_112065504_report

112065504 徐祈

A. How to compile
   1. cd ./HW3/src
   2. make
      - 會在 HW3/bin 下生成執行檔
   3. ../bin/hw3 ../testcase/{testcasename}.txt ../output/{testcasename}.floorplan
      - Ex. ../bin/hw3 ../testcase/public1.txt ../output/public1.floorplan
      - 會執行 testcase，並生成 output

B. The wirelength and the runtime of each testcase

	Wirelength	Runtime (sec)
public1.txt	171195568	590.01
public2.txt	28562687	590.01
public3.txt	1966113	590.02
public4.txt	63024850	590.07

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

   從頭到尾都是隨機決定的。Initial floorplan 會先隨便生成，後面 SA 還有機會在調整形狀。

D. The details of your floorplanning algorithm.

   Initial floorplan 也這次作業很大的難點之一，我的作法是:
   1. 先把所有 modules 擺到 Die 外面，而 Die 的範圍即是 chipsize 給定
   2. 把它們在 modules 陣列的位置，按 Area 的 decreasing order 排序
   3. 然後從大到小一一丟進 Die 裡面
      - 如果位置不合法(out of die, overlap with other module)
      - 就 Perturb (reshape, reposition, swap)當前 Die 中所有 module
      - 直到放進去為止，再放下一顆更小的 module

   這個方法會 work 的原因是，大的 module 比較不好放，所以要比小的先放!

```
init_floorplan()
    // initialize modules position to the outside of die
    InitializeOutOfBound(modules);
    // sort module area in decreasing order
    SortModulesByArea(modules, nets);
    // put the first(largest) module into die
    TryJumpIntoBox(modules[0], modules);
    for (auto &softMod: modules)
        while (!legal(modules))
            // perturb modules in fix outline
            perturbModulesInBox(modules);
            // try put softMod on
            TryJumpIntoBox(softMod, modules);

    originWL = calWirelength(nets, modules);
```

做完 initial floorplan 後，用 Simulated annealing (SA) 調整 modules，把 wirelength 調小。以下 pseudo code 跟老師上課講義 unit 4, p21 的幾乎一樣。在 perturb 的方式中，我用了四種:

- Reshape: 調整 soft module 形狀
- Swap: 隨機抓兩顆 module 交換位置
- Jump: 隨機把 module 移到另一個位置
- Move: 把 module 往上、下、左、右其中一個方位移動

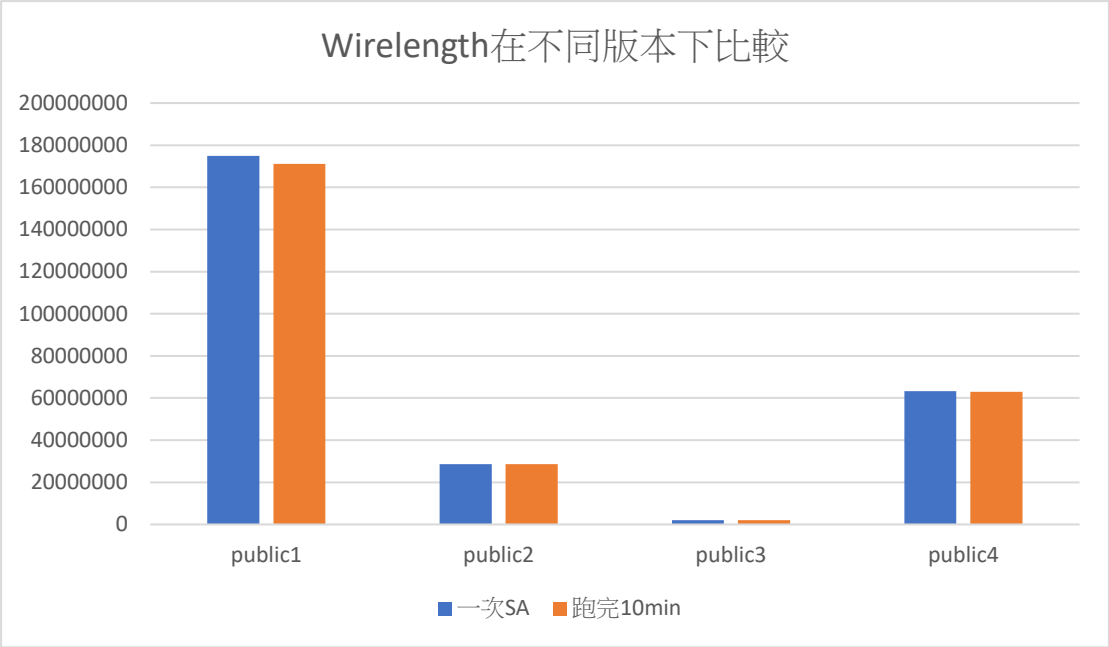
```
SA()
{
    double reject, reduceRatio = 0.9999;
    int nAns, uphill, T = 1000000, N = 70; // N: number of answer in T
    do{
        reject = nAns = uphill = 0;
        while (uphill < N and nAns < 2 * N) {
            perturb(rand() % 4);
            ++nAns;
            perturbWL = calWirelength(nets, modules);

            int deltaCost = perturbWL - originWL;
            if (deltaCost <= 0 or accept(deltaCost, T)) {
                if (deltaCost > 0)
                    ++uphill;

                originWL = perturbWL;
                if (originWL < bestWL) {
                    bestModules = modules;
                    bestWL = originWL;
                }
            }
            else{
                ++reject;
                revert(act);
            }
        }

        T *= reduceRatio;
    } while (reject / nAns <= 0.95 and T > 10);
}
```

Main function 也值得一提:

wirelength 的值取決於 seed 或 initial floorplan 的品質，做完一個完整 SA 通常不需一分鐘，加上我們有 10 mins 可以跑! 於是我讓 SA 做完後，在時間到前繼續重新做，並保留最好的結果記錄在 bestModules 中，最後輸出最佳結果。而 init_floorplan 和 SA 兩個 function 都分別定期檢查是否 timeout，如果 timeout 就 return true，break while loop。

```
main(){
    parser();
    while (Timeout())
    {
        if (init_floorplan() or SA())
            break;
    }
    output(outputPath);
}
```

E. Try your best to enhance your solution quality



從圖表看不太出差異，從數字上可以看出區別。除了 public2 的 free space 相當吃緊，很難繼續優化以外，其他都有略微進步。

- 只跑一次 SA

testcase	wirelength	runtime	status
public1	174955530	27.65	success
public2	28562687	595.03	success
public3	2010413	30.50	success
public4	63278450	19.62	success

- 跑 SA 到 10 mins（保險起見，沒跑滿 600 秒）

testcase	wirelength	runtime	status
public1	171195568	580.01	success
public2	28562687	580.01	success
public3	1966113	580.02	success
public4	63024850	580.07	success

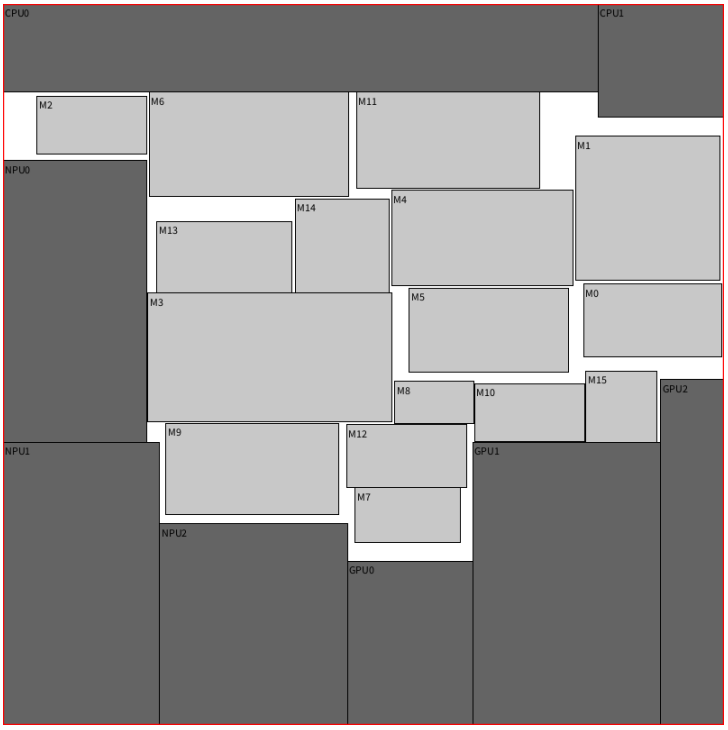
F. What have you learned from this homework? What problem(s) have you encountered in this homework?

- SA 的初始溫度調越高，雖然能跑越久但不保證能找到更好的解
- 設 timer 阻止程式超時
  - 雖然一直看 wall time 要 system call 也很浪費時間
- SA perturb 時需要記住擾動前狀態，如果擾動後發現不合法才能回朔

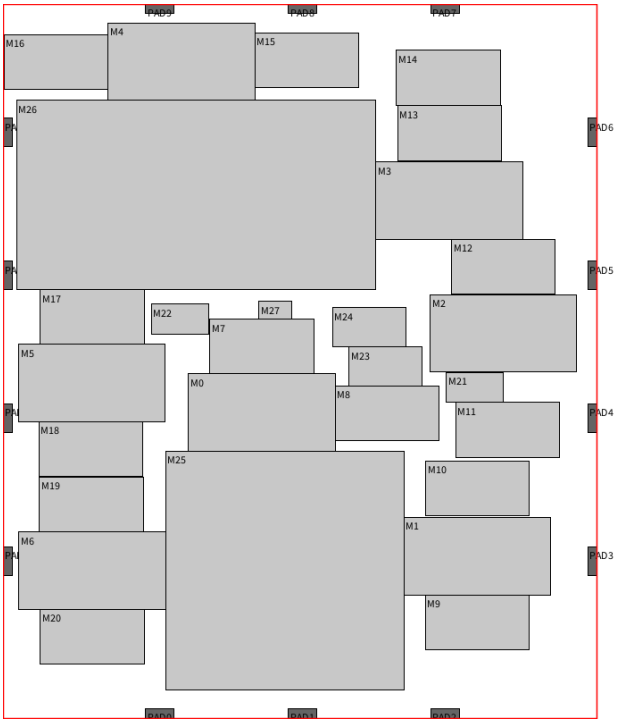
G. Floorplan result visialization

- Public1

- Public2



- Public3



- Public4