Report

111061590 林學謙

1. The wirelength and the runtime of each testcase

	wirelength	runtime
Public1	143391268	491.3
Public2	21239218	492
Public3	1413148503	491.79

使用特定的 seed 來取得最好的 wirelength，至於時間的都一樣，是因為使用相同的 runtime 去找解。

2. 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).

(source: https://github.com/richard8787/Global_Placement)

我一開始想要使用這次上課內容所教的 Bindensity 與設定 LSE objective function，但按照網路上的步驟，建立起 LSE_FG 與 LSE_F 分別用在不同的 evaluate function 裡面，但是結果都不盡人意，其總是分配完位置後，都擠在一起，無法使用 legalization，後來看到別人使用 SA(上面的網址)，使用的方式相當簡單，每次只取兩個 module 進行交換，但其只能透過 randomplace()，一開始就決定好每個 module 的位置(fixed 除外)，所以它的結果無法比使用 LSE+Bindensity 來的好，下面會有圖，來解釋其 code 運作原理。

```
// get the initial solution in random
randomPlace();
// Simulated Annealing parameters
double T0 = 100000000000000000;// Initial temperature
double T = T0;// temperature
double temp_T = T0;
const double cool = 0.00000000000000000000000000000000000000000000000000000000000000000000000000000001;// cool temperature
const double fastT = 0.0001;
// const double refineT = 0.0000001;
double gamma = 0.99995;
// const double gamma_refine = 0.99995;
const double init = (_placement.boundryRight() - _placement.boundryLeft()) + (_placement.boundryTop() - _placement.boundryBottom());
double curCost, nextCost;
double delta, p, rng;
int s1, s2;
double s1x, s1y, s2x, s2y;
clock_t init_time = clock();
const int TIME_LIMIT = 500 - 10;
int runtime = 0;
```

設定 SA 基本 parameter，這邊加入與上次功課 3 一樣的條件，透過 TIME_LIMIT 來限制演算法計算時間。

```
while (runtime < TIME_LIMIT)
{
    // get the s1 to change with s2
    while(true)
    {
        s1 = rand() % _placement.numModules();
        if(_placement.module(s1).isFixed()){
            continue;
        }else break;
    }
    while(true)
    {
        s2 = rand() % _placement.numModules();
        if(_placement.module(s2).isFixed()){
            continue;
        }else break;
    }

    wrapper::Module m1 = _placement.module(s1);
    wrapper::Module m2 = _placement.module(s2);
    s1x = m1.x();
    s1y = m1.y();
    s2x = m2.x();
    s2y = m2.y();

    curCost = (netWL(m1) + netWL(m2)) / init;
    m1.setPosition(s2x, s2y);
    m2.setPosition(s1x, s1y);
    nextCost = (netWL(m1) + netWL(m2)) / init;

    delta = nextCost - curCost;
    p = exp(-delta / T);
    rng = (double)rand() / (RAND_MAX + 1.0);
```

基本流程與 SA 一樣，找其可以交換的 s1 與 s2，並且將 module 的位置互相交換，但是這邊必須先確定其是否值得交換，所以必須寫一個 HPWL 來確定其 cost(看下圖)

```cpp
double GlobalPlacer::netWL(wrapper::Module &m)
{
    double HPWL = 0;
    for (unsigned int i = 0; i < m.numPins(); i++)
    {
        // initialize the max and min value
        double maxX = -999999999;
        double maxY = -999999999;
        double minX = 999999999;
        double minY = 999999999;

        // get the net max and min value
        wrapper::Net n = _placement.net(m.pin(i).netId());
        for (unsigned int i = 0; i < n.numPins(); i++)
        {
            maxX = max(maxX, n.pin(i).x());
            maxY = max(maxY, n.pin(i).y());
            minX = min(minX, n.pin(i).x());
            minY = min(minY, n.pin(i).y());
        }

        // accumulate the HPWL
        HPWL += (maxX - minX) + (maxY - minY);
    }
    return HPWL;
}
```

藉由這樣的方式找出兩個 module 各自的 HPWL，並且相加，按理來說，根據演算法，只要交換的 delta 為負數，表示其是好的結果，則進行交換。

```cpp
if (delta <= 0); // directly accept (cost is lower)
else if (p >= rng && T < fastT); // accept with probabiltiy (cost is higher but accept)
else // reject (cost is higher and reject)
{
    m1.setPosition(s1x, s1y);
    m2.setPosition(s2x, s2y);
}
T *= gamma;//cooling down
// if the temperature is smaller than cooling temperature, put some fire to temperature and annealing again until time up.
if(T < cool)
{
    temp_T *= 0.5;
    T = temp_T;
}
runtime = (clock() - init_time)/CLOCKS_PER_SEC;
```

這邊有加入，當溫度降到冰點，因為時間尚未到達，則會將原本的溫度乘上 0.5，如同 FastSA，當溫度降到一定的程度，則回拉回一定的溫度，讓其可以解

決 local minima 的問題。

3. **Try your best to enhance your solution quality. What tricks did you do to enhance your solution quality?**

為了達到較好的結果，因為我是使用 random seed，所以每次產生的結果都會不太一樣，為了讓其達到就好的結果，我將時間調成 100 秒，並且看其每次的結果，挑出最好的結果，讓其在特定的 public 有特定的 seed。

```
size_t seed = time(NULL);
if(_placement.numModules() == 12028){
    seed = 1702623600;
}else if(_placement.numModules() == 29347){
    seed = 1702626480;
}else if(_placement.numModules() == 51382)
{
    seed = 1702628830;
}
srand(seed);
cout << "seed: " << seed;
```

除此之外，最後將運算時間拉長，讓其有更長的時間進行 module 的交換，當然我也有使用過原本的 LSE+Bindensity，但真的找不出問題在哪，每次結果，他都會全部集中在一起，我有看到一位學生，他是直接將其全部放到 boundary 的外面，讓 legalize 自己將其放進去，雖然會有結果，但是比我這個方法來的更差，所以才決定使用 SA 來解決問題。

4. **Please compare your results with the previous top 5 students' results and show your advantage in solution quality. Are your results better than theirs?**

	Wirelength		
Rank	public1	public2	public3
1	68783367	8778312	456197589
2	79542407	9155930	478967869
3	83860394	9783498	463664700
4	73804994	11105419	413161709
5	80176617	10921603	539864787

我的演算法相對於使用 LSE+Bindensity，一定有差距，因為我每次只能交換兩個 module，可能會花費大量的時間卻沒有減少 wirelength，而我自己使用過上課所教的方法，其 global placement 的結果，必須透過 alpha 與 beta 控制，只要控制的當，其結果都相對於 SA 來的更好，但是我卻沒辦法找到其可以平均放置的方法。(下圖為自己的成果)

	wirelength	runtime
Public1	143391268	491.3
Public2	21239218	492
Public3	1413148503	491.79

至於該如何增加 SA 演算法的成果，第一個當然是把時間拉得更長，或是使用 parallel 的方法，讓其各自計算，這樣一定可以得到更好的結果，但我一直不會使用 parallel 的方式進行計算，所以就沒有使用 parallel 的方式。