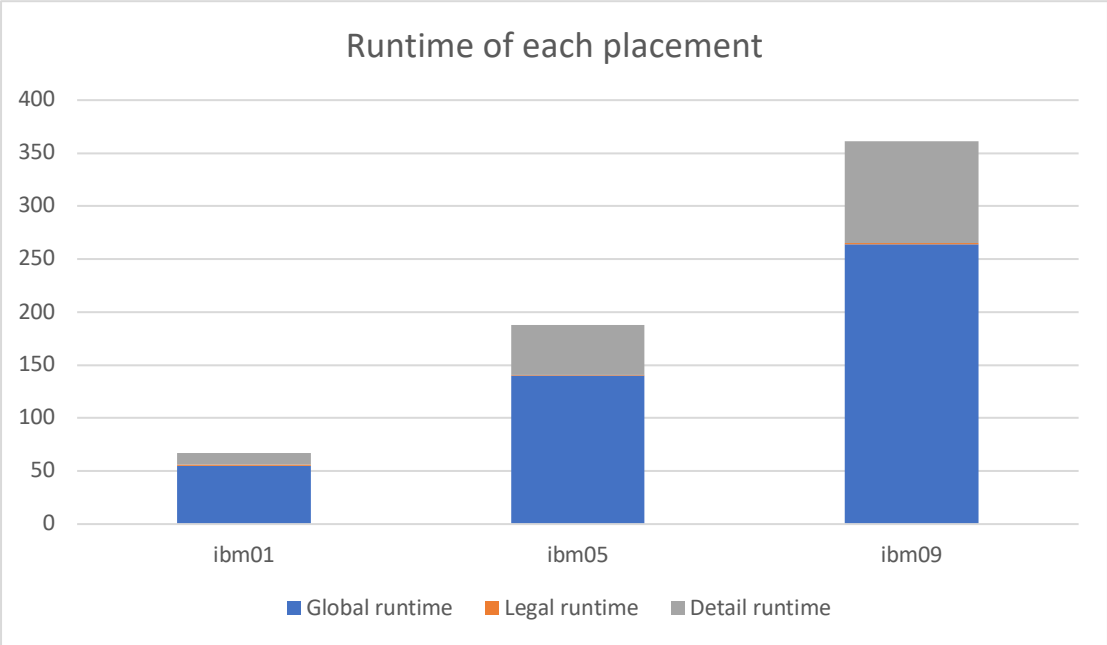
Homework 3 109065520 林元泰

1. The wirelength and the runtime of each testcase.

	Imb01	Imb05	Imb09
Global HPWL	**374556291**	**32623375**	**2932985103**
Legal HPWL	**609311362**	**53813525**	**4936342724**
Detail HPWL	**374671295**	**34752212**	**3304408419**
runtime	**66.0 sec**	**187.0 sec**	**361.0 sec**




```
////////////////////
Benchmark: ibm01-cu85

Global HPWL: 374556291   Time:   55.0 sec (0.9 min)
 Legal HPWL: 609311362   Time:    0.0 sec (0.0 min)
Detail HPWL: 374671395   Time:   11.0 sec (0.2 min)
 ======================================================================
       HPWL: 374671395   Time:   66.0 sec (1.1 min)
```

```
////////////////////
Benchmark: ibm05

Global HPWL: 32623375   Time:  140.0 sec (2.3 min)
 Legal HPWL: 53813525   Time:    0.0 sec (0.0 min)
Detail HPWL: 34752212   Time:   47.0 sec (0.8 min)
 ======================================================================
       HPWL: 34752212   Time:  187.0 sec (3.1 min)
```

```
////////////////////
Benchmark: ibm09-cu90

Global HPWL: 2932985103   Time:  264.0 sec (4.4 min)
 Legal HPWL: 4936342724   Time:    1.0 sec (0.0 min)
Detail HPWL: 3304408419   Time:   96.0 sec (1.6 min)
 ===================================================================
       HPWL: 3304408419   Time:  361.0 sec (6.0 min)
```

3. The detail of my algorithm.

Part 0 : Initialize the ExampleFunction

下圖是 **ExampleFunction class** 中所有的元素，各自的功能都寫在註解中。

```
double Width; // the width of the placement
double Height; // the heigh of the placement
double Area; // the total area of the placement
int module_num; // number of modules in the placement
int net_num; // number of net in the placement

//*********************************************************
double alpha; // Smoothing parameter
double *exp_arr; // 4 exp results per module

// parameters of LSE
double x_pos;
double x_neg;
double y_pos;
double y_neg;

//*********************************************************
double beta; // for density
int bin_num; // number of bins in the edge
double density_r; // density ratio
double *bin_density;
double target_density;

double bin_width; // width of a bin
double bin_height; // height of a bin

double bin_center_x; // x coor of center of bin
double bin_center_y; // y coor of center of bin

double dx; // dx = |xi - xb|
double dy; // dy = |yi - yb|

// parameters for overlap function
double x_a;
double x_b;
double y_a;
double y_b;

// results of overlap function
double overlap_x;
double overlap_y;
double *overlap_g; // 2 grad (x , y) for a module
```

當呼叫一個 ExampleFunction 時，我們會初始化一些元素。

```
ExampleFunction::ExampleFunction(Placement &placement)
    :_placement(placement)
{

    Width =_placement.boundryRight() - _placement.boundryLeft();
    Height = _placement.boundryTop() - _placement.boundryBottom();
    Area = Width * Height;
    alpha = Width / 600;
    beta = 0;
    module_num = _placement.numModules();
    net_num = _placement.numNets();


    exp_arr = new double[module_num * 4]();
    x_pos = 0;
    x_neg = 0;
    y_pos = 0;
    y_neg = 0;

    bin_num = 10;
    bin_width = Width / bin_num;
    bin_height = Height / bin_num;
    bin_density = new double[bin_num * bin_num]();
    x_b = 0;
    y_b = 0;
    overlap_g = new double[module_num * 2]();
    target_density = 0;
    for(int i = 0 ; i < module_num ; i++){
        target_density += _placement.module(i).width() * _placement.module(i).height();
    }
    target_density = target_density / Area;

}
```

Part 1 : Wirelength function : LSE

- **Calculate parameters :**

  參數的計算我參考了講義中的算式

```
// calculate LSE
for(int i = 0 ; i < module_num ; i++){
    Module cur = _placement.module(i);

    // calculate the parameters of LSE function
    exp_arr[4 * i] = exp( x[2 * i] / alpha );
    exp_arr[4 * i + 1] = exp( (-1) * ( x[2 * i] / alpha ) );
    exp_arr[4 * i + 2] = exp( x[2 * i + 1] / alpha );
    exp_arr[4 * i + 3] = exp( (-1) * (x[2 * i + 1] / alpha ) );

    //sum of those arameters
    x_pos += exp_arr[4 * i];
    x_neg += exp_arr[4 * i + 1];
    y_pos += exp_arr[4 * i + 2];
    y_neg += exp_arr[4 * i + 3];

}
```

- **Calculate g :**

  對每個不是 **fixed** 的 **module** 計算他們的 **x,y** 的 **g**，**G** 的計算式我參考了在 **github** 上找到的相關程式碼

```
// calculate gradient
for(int i = 0 ; i < net_num ; i++){

    int pin_num = _placement.net(i).numPins();
    for(int j = 0 ; j < pin_num ; j++){
        Module cur =  _placement.module(i);
        int cur_ID = _placement.net(i).pin(j).moduleId();

        // if module is fixed, g = 0
        if( cur.isFixed() == 1 ){…
        // if module is not fixed, calculate g with LSE parameters
        else{…

    }
    // implement the f1
    f1 += alpha *  ( log(x_pos) + log(x_neg) + log(y_pos) + log(y_neg) );  // LSE wirelength solution
}
```

Part 2 : Bin density function : Bell-shaped

- 我會在每個 **bin** 中去測試所有的還沒有 **fixed** 的 **module**，並計算出該 **module** 在這個 **bin** 中的 **overlap** 的參數，最後決定這個 **module** 是否要放進這個 **bin**。

```
// smoothing by Bell-shaped function
// total num of bins : bin_num * bin_num
// testing which bin should each module to locate
for(int i = 0 ; i < bin_num ; i++){

    for(int j = 0 ; j < bin_num ; j++){

        // calculate the coor of center of bin ( x_b , y_b )
        x_b = ( (i + 0.5) * bin_width ) + _placement.boundryLeft();
        y_b = ( (j + 0.5) * bin_height ) + _placement.boundryBottom();

        // try every module in bin[i][j]
        for(int k = 0 ; k < module_num ; k++){

            Module cur = _placement.module(k);
            double cur_width = cur.width(); // x coor of current module
            double cur_height = cur.height(); // y coor of current module

            // if not fixed, calculate the bin density and g of overlap
            if( cur.isFixed() == 0){…

            // calculate bin density for bin[i][j]
            bin_density[ j * bin_num + i ] += density_r * overlap_x * overlap_y;

        }

        // calculate f2
        f2 += beta * pow(( bin_density[j * bin_num + i] - target_density) , 2);

        // calculate true grad
        for(int k = 0 ; k < module_num ; k++){
            g[2 * k] += beta * 2 * ( bin_density[j * bin_num + i] - target_density) * overlap_g[2 * k];
            g[2 * k + 1] += beta * 2 * ( bin_density[j * bin_num + i] - target_density) * overlap_g[2 * k + 1];
        }
    }
}
```

- 當 **module** 不是 **fixed** 時，程式會執行上塗紅匡內的程式

  X 軸與 **Y** 軸的 **A** 和 **B** 我是參考講義上的算式。下方 **bell-shaped** 的範圍設定是參考自網路上的相關程式碼

```
// calculate a , b for x and y
x_a = 4 / ( (cur_width + bin_width) * ( 2 * cur_width + bin_width) );
x_b = 4 / ( cur_width * ( 2 * cur_width + bin_width) );
y_a = 4 / ( (cur_height + bin_height) * ( 2 * cur_height + bin_height) );
y_b = 4 / ( cur_height * ( 2 * cur_height + bin_height) );

// calculate the dx = | xi - xb | and dy = | yi - yb |
double xi = cur.centerX(); // xi = x[2 * k]
double yi = cur.centerY(); // yi = x[2 * k + 1]
dx = abs( xi - x_b );
dy = abs( yi - y_b );

double cond1_x = bin_width  + (cur_width / 2);
double cond2_x = cond1_x + bin_width;
double cond1_y = bin_height  + (cur_height / 2);
double cond2_y = cond1_y + bin_height;
```

Overlap function 的計算參考自講義的算式

```
// calculate overlap function for x coor
if ( dx >= 0  && dx <= cond1_x ){

    overlap_x = 1 - ( x_a * pow( dx , 2 ) );

}
else if ( dx >= cond1_x && dx <= cond2_x){

    overlap_x = x_b * pow( ( dx - bin_width - ( cur_width / 2 ) ) , 2);

}
else if ( dx >= cond2_x ){

    overlap_x = 0;

}
else{
    cout<<"[error] Wrong dx in overlap function."<<endl;
}
```

Overlap 的 **G** 的計算參考自網路上的相關程式碼

```
// calculate g of overlap function for x coor
if (dx <= cond1_x ){

    overlap_g[2 * k] = density_r * (-2) * x_a * dx * overlap_y;

}
else if( dx >= cond1_x && dx <= cond2_x ){

    overlap_g[2 * k] = density_r * 2 * x_b * (dx - ( bin_width + cur_width / 2 ) ) * overlap_y;

}
else{
    overlap_g[2 * k] = 0;
}
```

Part 3 : Global placement arrangement

- **Initialize placement 需要的資料**

```
//initial example function
ExampleFunction ef(_placement);
        class std::__1::vector<double>
//******  Include other forward declarations here  ********************************************************//
vector<double> solution_v(ef.dimension());  // create the solution vector
srand(1234567);
randomPlace(solution_v);  //generate the value of solution vectors for modules

//*************************************************************************************************************//
```

- **用 for loop 來執行 optimizer，並在每一輪開始時修改參數。並在執行玩 optimizer 後檢查所有的 module 有沒有在 boundary 中（紅匡處）。**

```
NumericalOptimizer no(ef);

int right = _placement.boundryRight();
int left = _placement.boundryLeft();
int top = _placement.boundryTop();
int bottom = _placement.boundryBottom();
int Width = right - left;
int Height = top - bottom;

for(int i = 0 ; i < 2 ; i++){
    cout<<i<<" th round"<<endl<<endl;

    ef.beta += 2000; // beta increase 2000 every round
    no.setX(solution_v);
    no.setNumIteration( 40 ); // user-specified parameter
    //no.setNumIteration( _placement.numModules() / 500); // user-specified parameter

    no.setStepSizeBound( max(Width , Height) * 7); // user-specified parameter
    no.solve();             // Conjugate Gradient solver

    // check all of modules are in the boundary
    int module_num = _placement.numModules();
    for(int j = 0 ; j < module_num ; j++){⋯

}
```

- **檢查後如果超出 boundary，會將 module 靠著最近的 boundary 擺放，並將結果存回 module 和 solution vector。**

```
int flag = 0;

// if x coor is out of boundary
if( module_x + module_width > right){
    module_x = right - module_width;
    flag = 1;
}
else if (module_x - module_width > left){
    module_x = left + module_width;
    flag = 1;
}
```

```
// store the changed coor back to solution vector

if( flag == 1){
    _placement.module(j).setCenterPosition(module_x , module_y);
    solution_v[2 * j] = module_x;
    solution_v[2 * j + 1] = module_y;
}
```

4. What trick did you do to speed up your program or enhance your solution quality?

- **Number of iteration:**

  經過幾次測試後，我發現如果我把 **iteration** 次數設太大的話，後面的 **iteration** 往往都是重複的數字，跳不出更好的結果，所以我將 **iteration** 的次數勁量調整到合適的大小，既可以減少執行時間，又可以保留好的 **solution quality**。

- **Step size boundary**

  以 **max(width , height)**當作單位，我測試了用不同的倍數執行的結果，如果倍數太小會跑很多 **iteration** 還沒有到達極限，太大的話會卡住跳不動。

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

我的結果比之前的同學的結果差。原因大概可以分成兩大類：

- 可能先前的同學使用的 **example function** 得出的Ｆ和Ｇ比較好，連帶著 **iteration** 的結果會更好。
- 可能先前的同學在 **iteration** 上的參數設置很精準，這樣 **iteration** 出來的結果可以更小。