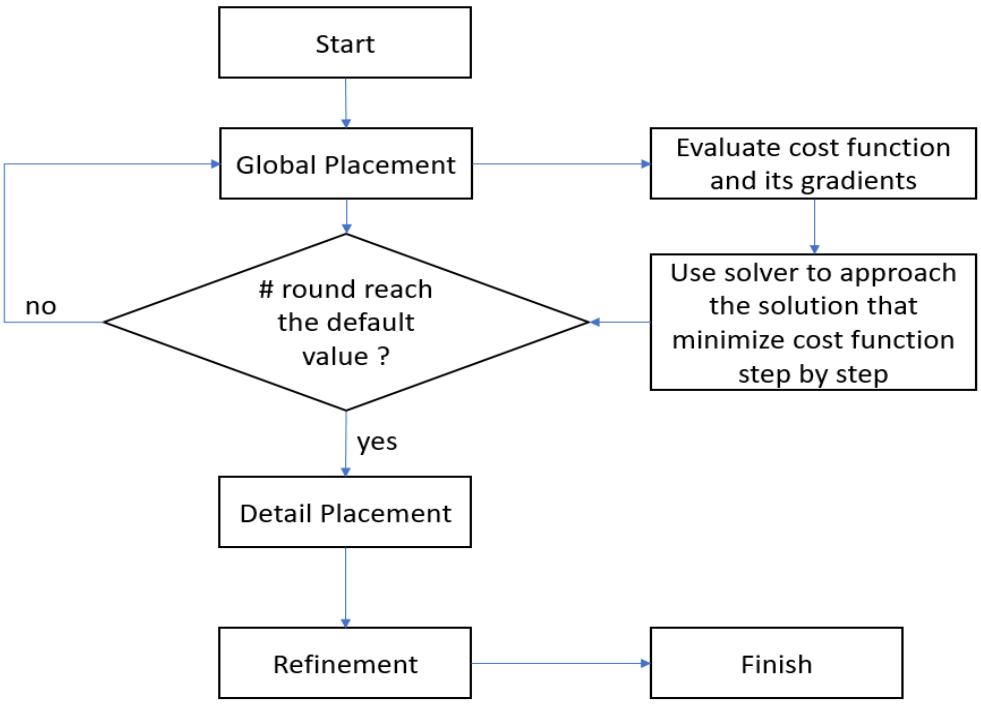
實作 (Implementation)

(1) 演算法流程 (Algorithm Flow)



⇧演算法流程圖

詳細之流程說明(只說明實作部分 global placement)：

1. 在 global placement 執行前會先使用亂數的方式決定所有 modules 的座標，作為將來使用 LSE 估線長時的初始解。

2. 接著設定 global placement 所要執行的次數後開始執行 global placement 直到設定輪數到達。

3. 每次 global placement 都使用 solver 來使的 cost function 之值最小，cost function 由 wirelength 和 bin density 來估算，其中 wirelength 使用 LSE 來估算(因其能微分)；bin density 則是將可適用的面積切成 15*15 的 bins，並使用上課講義中計算覆蓋率的公式來運算，其權重隨著輪數的增加，如此所有的 module 才不會過於集中導致 legalization 失敗。

p.s. 第一輪不考慮 bin density，只先讓線長盡量變小。

(2) 虛擬碼 (Pseudocode) (global placement only**)**

```
Input: placement info
Output: coordinate of each module
begin
    Set number of iteration and randomly determine each module's coordinate
    While(default number of iteration has not reached){
        Set weight of bin density and parameters for solver
        Set each module's coordinate according to the result return by solver
        Check if any module beyond the boundary, if yes, make some change on their
        coordinate to fit the constraint.
    }
end
```

(3) 資料結構 (Data Structure)

在 EampleFunction.h 中新增一些變數

```
/*****************************
******* variable for LSE ******
*****************************/
double reduce; //縮小 exp(x)中的 x，否則無法進行運算
vector <double> exponential; //紀錄個 module 的±exp(x)及±exp(y)

/*******************************
******variable for BinDensity *****
*******************************/
int rounds; // bin density 之權重(講義上的 beta)
int nbinperside; // number of bin per side
double targetdensity; //理想 density = 全部 module 的面積總和/可用面積
double binWidth; //每個 bin 的寬
double binHeight; //每個 bin 的高
```

vector <double> BDE; // 每個 bin 的 density

vector <double> g_BDE;

(4) Compare to the top 3 results of last year

My result

	ibm01	ibm05
HPWL	85214684	13596626
RUN TIME	99 sec	154 sec

Ibm01 較去年前三較差，可能原因為參數設定較不好或是初始解較差。

Ibm05 比去年第三名稍佳，仍輸給前兩名。