B07901026 吳禎軒 PA3 Report

$Data Structure and how I designed the code:

1. Analytical method:

$$\min\{\mathrm{W}(\mathrm{x},\mathrm{y}) + \lambda \sum [D(x,y) - Mb]^2\}$$

The gradient solver takes only wirelength model into consideration for the first iteration, while both wirelength model and density model are evaluated for the rest of the iterations. I pursue the minimum wirelength first and gradually spread the modules to reduce density.

2. Wirelength model

(a)LSE model: Although we learned that WA model has better performance on the lecture, I choose LSE because it is easier to differentiate.

(b)parameter:

#_gamma(for exponential terms)- (ChipWidth+ChipHeight)/1200

(c)data structure:

_LSEcontainer- a N*4(N = model number) vector storing the four exponential terms in the formula

#_tF- Traverse each pin in every net and save the value for objective function

#gradient computation- Note that the values of _tF are components for gradient computation, so we can omit the necessity of recalculation

3. Density model

(a)Sigmoid smoothing: I select sigmoid smoothing because it doesn't require case classification like Bell shape smoothing.

(b)parameter:

#_binNum- 225, close to the setting of legalization program

#_alpha(for sigmoid function)- range from 10^{-7} ~ 10^{-4}

#_Mb(for formula)- N/BinNumber

#_lambda(for formula)- proportional to the chip area and increases every iteration

(c)data structure:

#_xd, _yd, _density- Traverse each module in every bin and store N values in these arrays.

4.Initialization: The first method is to randomly distribute the modules in the chip; the second method is to divide the chip into N bins and place each module in a bin.

5.Manual outline legalization: After each iteration, manually push the illegal modules into the boundary.

$My finding

1. Balance between wirelength model and density model

The gradient solver sometimes becomes stagnant and has identical gradient values in each step, which not only reduces the effectiveness of density model but increases considerable running time. Besides, the gradient solver may continue reducing the value of wirelength though the value of density is much greater. I infer that these phenomena are due to the unbalanced relationship between wirelength model and density model. When the gradient value or the objective function value for both models differ significantly, the solver fails to find a better solution or searches for the solution that we don't desire.

2.Density model normalization

To reduce the difference of the gradient value or the objective function value for both models, I decide to normalize the density term. The following is my current method:

(a)Modify lambda- As mentioned previously, lambda is proportional to the chip area and increases every iteration.

(b)divided by the initial density value- The objective function value for density is restricted to 0~1.

I also tried another method but failed: In the beginning of each iteration, multiply the gradient value and the objective function value for density by a dynamic variable so that they are very close to those for wirelength.

3. Parameter setting may be pivotal

My program is very sensitive to the parameter setting. The performance of it is just average, and I guess it may be improved if I have more time to find a better combination of parameters. The following is my experience:

(a)_alpha: Theoretically, a value close to 0 is suitable for it. However, its appropriate value varies for different cases in practical. A large value causes inf or nan for gradient computation, while a small value pulls all modules outside the boundary and leads to legalization failure.

(b)_Mb: My current setting is the average number of module in each bin. I've tried the average area of all modules in each bin, which is not successful.

(c)step size/ number of steps/ number of iteration: These parameters really differ for different cases, so I have to adjust them one by one.

(d)random seed for initialization: The result may change by up to 5%.

4. Speed up the program

The calculation of sigmoid function in density model takes considerable running time: 2 sec for a step in gradient solving. I increase the speed by only examining the bins that cover the module instead of all bins. However, there is

a tradeoff between running time and accuracy, which indicates that the process of spreading the crowded modules may be less effective and that legalization failure is more likely to occur.