PD PA2 Report

R10943185 葛艾

A. My implementation

1. Flow chart

Parser -> Initial b star tree -> Simulated annealing -> output

2. Parser

I divide my parser into two parts :
void parse_blk(input_blk), void parse_net(input_net).
Below is my data structure to store the input information :
vector<Node*> node_list : b star tree node.
vector<Block*> blk_list : block information.
vector<Terminal*> tml_list : terminal information.
vector<Net*> net_list : all nets and each net has its cell list (terminals and blocks).

3. Initial b star tree

I choose the left skew floor plan to be my initial b star tree settings.

void coordinate(Node* node) :
Use this function to compute y-coordinates, during SA, I repeat calling this function.
(1)Horizontal contour
list<contour_node> contour_line : store nodes on the horizontal contour line.
(2)Amortized O(1) time
Find all blocks that is under the current block, and return the max y coordinate. Update the contour line. We only have to check a few blocks under the current block. -> Linear time after all.

4. Simulated annealing

(1)neighborhood structure:
rotate and module/delete and insert one node/swap two nodes
(2)reject rate:
I add a reject rate into my code, if there are too many uphill move in one iteration (rejectRate > 0.96), I stop the session to accelerate my implementation.

Below is my pseudo code:

```
Begin
reject <— 0;
T <— 100000;
while rejectRate < 0.96 or T > 0.000001 do
        reject <— 0;
        for 0 <= i < 1000 do
                Pick a random neighbor s' of s;
                Delta <— cost(s')-cost(s);
                if delta <= 0 then
                        s <— s';
                if delta > 0 then
                        s <— s' with probability exp(- delta/T);
                        reject++;
        T*=0.6;
        rejectRate <— reject/1000;
        if fit fixed outline then
                return;
End
```

B. My results

ami49:

ami49	alpha=0.25	alpha=0.5	alpha=0.75
Cost	11061288	20504168	29898916
Wirelength	1341984	1803438	1380036
Area	40219200	39204900	39405212
Chip width/outline width	5320/5336	5334/5336	5222/5336
Chip height/outline height	7560/7673	7350/7673	7546/7673
Runtime (s)	4.750560	1.219900	3.509907

ami33:

ami33	alpha=0.25	alpha=0.5	alpha=0.75
Cost	401058.625	677335.250	948674.875
Wirelength	105243	109090.5	93092.5
Area	1288504	1245580	1233869
Chip width/outline width	1211/1326	1148/1326	1183/1326
Chip height/outline height	1064/1205	1085/1205	1043/1205
Runtime (s)	1.796990	2.134688	3.509907

C. Discussion

1. How to reduce cost?

What I do:

(1) Adjust the parameter in simulated annealing:

T = 100000

frozen T = 0.000001

r = 0.6

(2) Adjust cast function:

When all modules don't fit the fixed outline, my cost function is:

(x_error + y_error)*10

x_error : Width outside the fixed outline

y_error : Height outside the fixed outline

When all modules fit the fixed outline, my cost function is:

- (chip_w*chip_h)*5 / (alpha*area+(1-alpha)*hpwl)

By doing this I can quickly converge to optimal solution and get a better solution.

What to do :

(1) Define some other useful neighborhood structure because different neighborhood structure may lead to different simulated annealing process and effect solution directly (I implement this discussion in my handwriting homework2 DIY problem).

(2) Adjust my initial b* tree. In this homework I use the left skew tree as my initial b* tree, if I use a better initial tree (more balanced), it will reduce the cost.

2. How to improve runtime?

What I do:

(1) Add a reject rate into my code(mentioned above).

(2) Implement Amortized O(1) time(mentioned above).

(3) Adjust cast function(mentioned above).

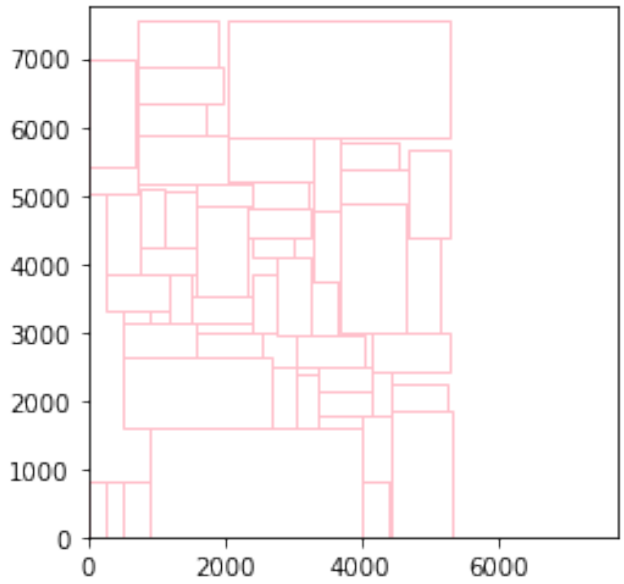
What to do :

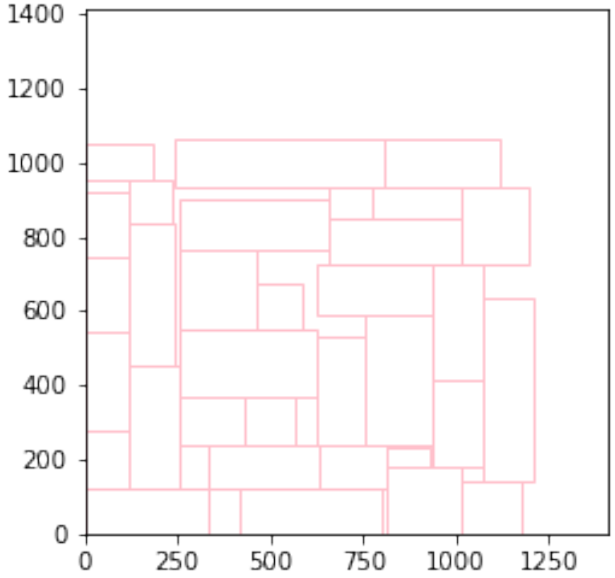
(1) When doing deletion and insertion or swapping, I recreate the whole b* tree, but in fact it will be faster if I only adjust the subtree instead of the whole tree.

(2) Adjust my initial b* tree. In this homework I use the left skew tree as my initial b* tree, if I use a better initial tree (more balanced), it will reduce the time to reach my optimal solution.

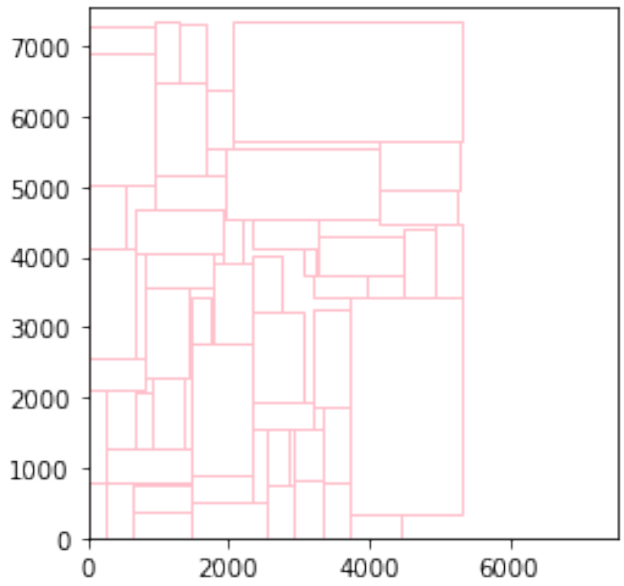
D. Visualize my results

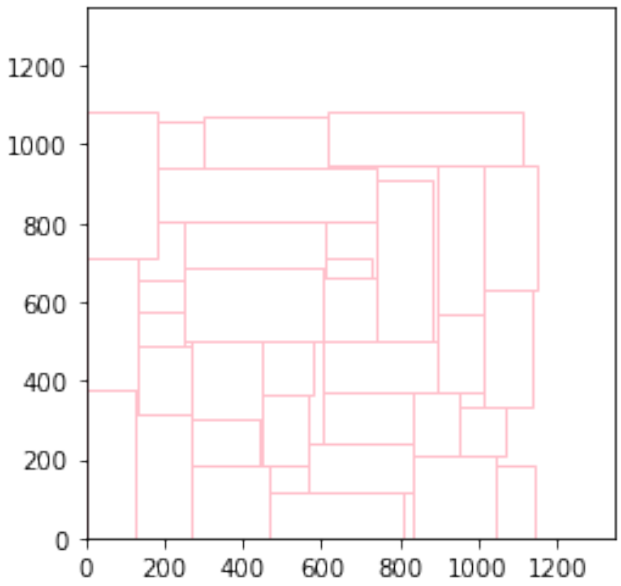
ami49	ami33
alpha=0.25	alpha=0.25





ami49	ami33
alpha=0.5	alpha=0.5





ami49	ami33
alpha=0.75	alpha=0.75