106-2 Physical Design Final Report

1. Topic:

Improve Floorplanning by Reinforcement Learning

2. Member:

r06921048 李友岐, r06921061 陳家瑄, r06943082 彭俊又

3. Introduction:

This semester, the second programming assignment [1] asks us to write a fixed-outline chip floorplanner that can deal with hard macros. Given a set of rectangular macros and a set of nets, the floorplanner places all macros within a rectangular chip without any overlaps. The objective function is a combination of area of chip bounding box and the total net wirelength. The total wirelength W of a net is computed by half-perimeter wire length (HPWL). The objective for this problem is to minimize

$$Cost = \alpha A+(1-\alpha)W$$

where A denotes the bounding-box area of the floorplan and W is the total wire length of all nets. α is a user defined parameter to balance the influence of final area and wirelength. Fig. 1 shows an example of floorplanning. To be an acceptable solution, all macros of the floorplan can not cross the fixed-outline. [1]

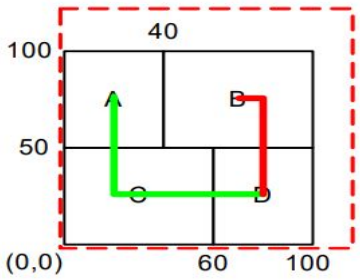


Fig. 1: A floorplanning problem and its solution

In floorplanning, we usually use Simulated Annealing (SA) to optimize the design. The algorithm of Simulated Annealing is shown in Fig. 2 and the step 6 of algorithm is to pick a random neighbor solution. Normally, we implement the floorplan by B*-Tree. For hard macro, there are three B*-Tree Perturbations, which are rotate a macro, delete & insert, and swap 2 nodes, respectively. The first one is to rotate the macro 90°, and the tree topology remains same. The second one is to delete a macro from its current location and then insert it to somewhere else. The third one is to swap the location of two nodes. Fig. 3 is an example of applying these operations to a B*-Tree. By these operations, we can obtain different floorplanning solution. After applying an operation, the resulting floorplan is a neighbor solution of previous floorplan. [2]

```
1 begin
2 Get an initial solution S;
3 Get an initial temperature T > 0;
4 while not yet "frozen" do
5     for 1 ≤ i ≤ P do
6         Pick a random neighbor S' of S;
7         Δ ← cost(S') - cost(S);
          /* downhill move */
8         if  Δ ≤ 0 then S ← S'
          /* uphill move */
9         if  Δ > 0 then S ← S' with probability e^(-Δ/T) ;
10    T ← rT;  /* reduce temperature */
11 return S;
12 end
```

Fig. 2: Generic Simulated Annealing Algorithm [2]

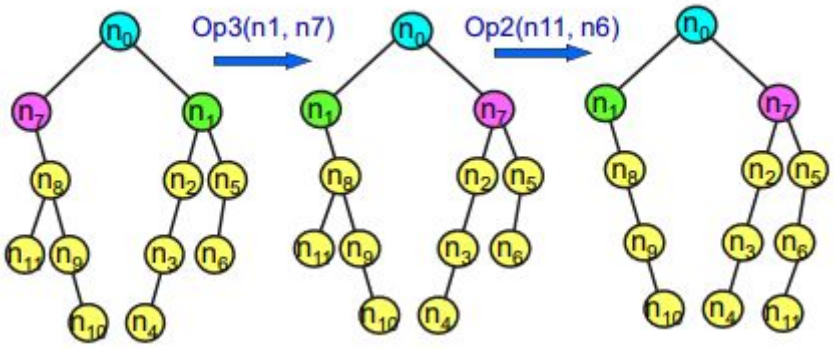


Fig. 3: An example of B*-Tree perturbation [2]

In general, we choose which operation we want to apply randomly. However, randomly searching for a better neighbourhood solution doesn't seems to be efficient nor reasonable. Although we can use a heuristic approach to obtain a better search result by changing the probability of different B*-tree operation during different stage of the Simulated Annealing. We suspect there is a better approach to optimize the floorplan. In this project, we will apply Reinforcement Learning (RL) for the optimization and compare their experiment results. Section 4 is our proposed method. Section 5 and 6 are the experient results and conclusion, respectively.

4. Proposed Method:

In SA, there’re two scopes of random choosing. First randomly choose a move and then randomly choose one or two node to perform the move. We change the random choosing heuristic in the first scope by RL Model, deciding which move to perform in given state. The input of the model is called observation, which contains cooldown progress, moves history, dead area and boundary violation information that help the RL Model to make decision. Fig. 4 shows the illustration of RL system.

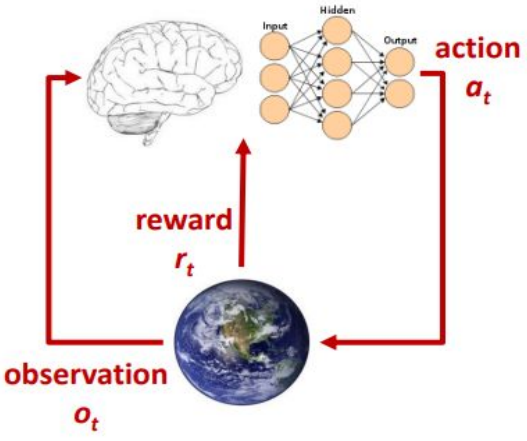


Fig. 4: The system of RL

5. Experiment Results:

Our experiment consists of two floorplanner. The first one solved the floorplanning problem by SA with pre-tuned parameter (33% for each operation), while the other one modified the SA algorithm by replacing the random-move-choose heuristic by RL algorithm and all the other parts remaining the same for fairness.

In programming assignment #2, TA provides MCNC benchmark, composed of xerox, ami33, ami49, apte and hp. Fig. 5 is the experiment result on these test cases, where SA and RL denote our first and second floorplanner. The cost function is defined as follow.

$$Cost = \alpha A+(1-\alpha)W$$

where α is equal to 0.5 here, A denotes the bounding-box area of the floorplan and W is the total wire length of all nets. Each case is run by 10 times, and we calculate the average cost of these 10 runs. We can find that our approach doesn't improve the result significantly. For ami49, apte and hp, the cost is slightly reduced. For xerox and ami33, the cost is slightly increased. Obviously, there is no big difference between the final cost of our two floorplanner.

Case	Cost of SA	Cost of RL	Rate
xerox	10685270	10711632	-0.25%
ami33	695743	701885	-0.88%
ami49	20900282	20798463	0.49%
apte	24625345	24334181	1.18%
hp	4967235	4871160	1.93%

Fig. 5: experiment result for MCNC benchmark

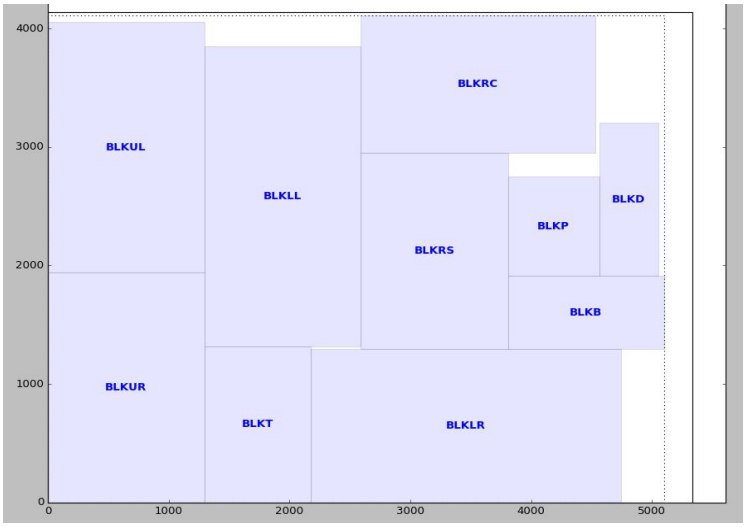
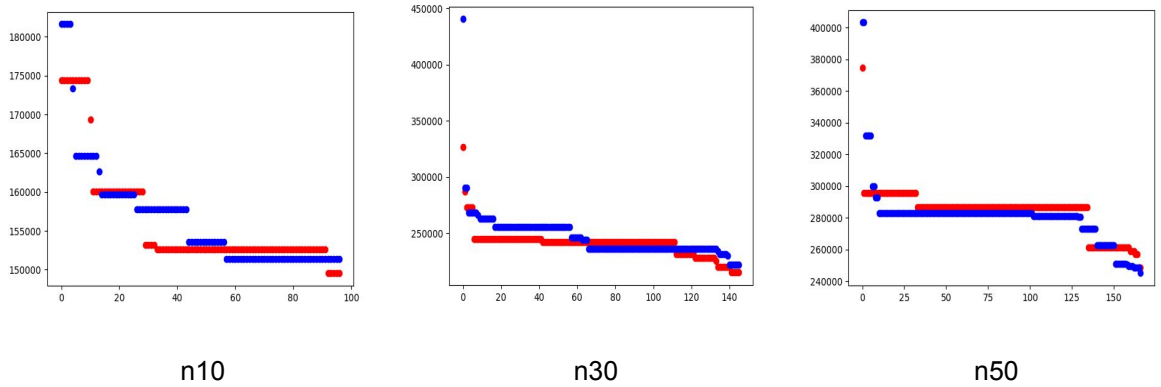
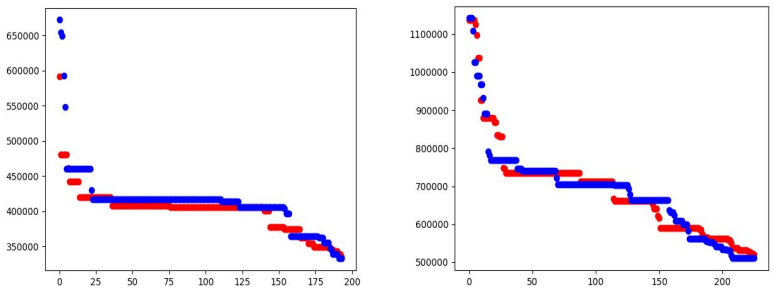


Fig. 6: floorplanning result of xerox

To figure out why the result doesn't improve significantly, we visualize the floorplan. Fig. 6 shows the final floorplan of xerox. It is obvious that the fixed-outline is too narrow, where the fixed rectangular is almost filled with all the blocks. Therefore, there is only small room for improvement.



n10 n30 n50



n100 n200

Fig. 7: The experiment result of GSRC benchmark

In addition to MCNC benchmark, we also test the GSRC benchmark, which consists of 5 cases, n10, n30, n50, n100 and n200. Where the number represents the quantity of blocks. For instance, n10 contains 10 blocks. Fig. 7 is the experiment result of GSRC benchmark, where the blue line and red line represent our first and second floorplanner, respectively. The X-coordinate is the number of iteration while the Y-coordinate is the cost of floorplan. According to the result, we can find that the floorplanner with RL model lead to smaller cost initially. However, the cost of two floorplanner converges as the number of iteration grows. This phenomenon indicates that our RL model help us to find a better floorplan at the beginning, but the final cost of two floorplanner would end up close eventually.

6. Conclusion:

In this project, we try to improve the floorplan cost by Reinforcement Learning. There are three operations of B*-Tree preturbation. For each iteration, we need to pick which move to apply. Usually, we do this randomly but it seems inefficient. As a result, we use RL model to decide which move to apply. However, the resulting cost doesn't improve significantly. This is possibly due to the fact that the fixed-outline limit the floorplan result to some degree. Hence, it doesn't have much room to improve. Besides, maybe the key is to pick the right macro to move rather than pick the right operation to apply. Therefore, building a floorplanner where we pick which macro to move based on RL instead of randomly choosing can be our future work.

7. Reference:

[1] Chang, "Programming Assignment #2: Fixed-outline Floorplanning of Physical Design for Nanometer ICs," 2018.

[2] Chang, "Lecture note unit 4 of Physical Design for Nanometer ICs," 2018.

[3] Wong & Liu, "A new algorithm for floorplan design," DAC, 1986.

[4] Kirkpatrick, Gelatt, and Vecchi, "Optimization by simulated annealing," Science, 1983.

[5] Abadi, "TensorFlow: A system for large-scale machine learning," OSDI, 2016.

[6] Kahng, "Machine Learning Applications in Physical Design: Recent Results and Directions," ISPD, 2018.