CS6135 HW3 Report

109062509 熊 磊

I. How to compile and execute?

- Compilation:
  One can go to directory, `HW3/src`, and execute `make`, the executable file, named as `hw3`, will be generated in directory, `HW3/bin`.

```
[g109062509@ic53 src]$ make
```

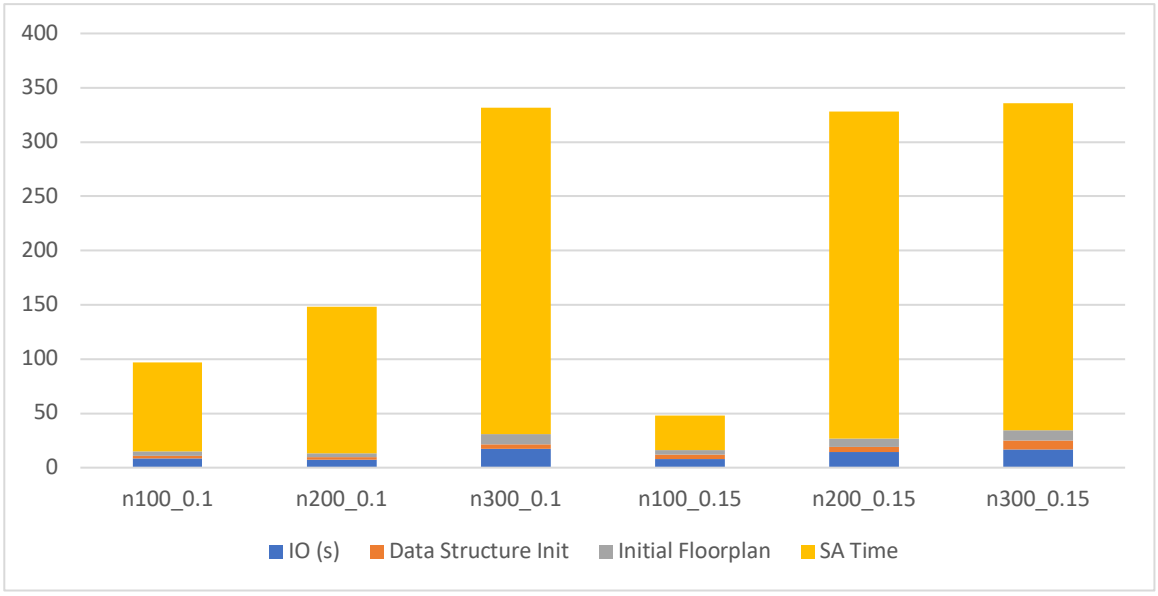
- Execution
  In directory, `HW3/bin`, execute `hw3` follow by the require files.

```
../bin/hw3 ../testcase/n100.hardblocks ../testcase/n100.nets ../testcase/n100.pl ../output/n100.floorplan 0.1
```

II. The wirelength and the runtime of each testcase with the dead space ratios 0.1 and 0.15, respectively.

Dead space	**0.1**			**0.15**		
	n100	n200	n300	n100	n200	n300
Wirelength	**212723**	**405497**	**598099**	**205567**	**388871**	**548968**
IO (s)	0.00369	0.00314	0.007691	0.003567	0.006524	0.007567
Data Structure Init (s)	0.001209	0.001281	0.00179	0.001761	0.001949	0.003549
Initial Floorplan(s)	0.001833	0.00158	0.004206	0.00183	0.003453	0.004221
SA Time(s)	82	135	301	32	301	301

Statistic



III. How small the dead space ratio could be?

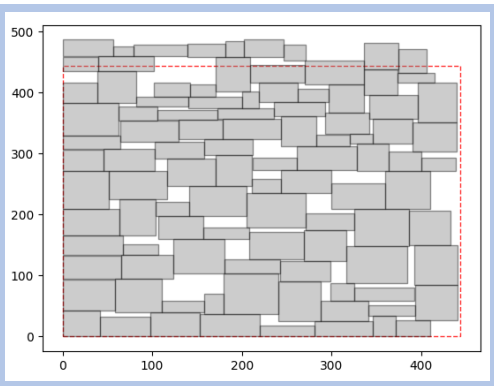
Dead space	0.1			0.15		
	n100	n200	n300	n100	n200	n300
Dead Space Ratio	0.098	0.0969	0.1	0.128	0.147	0.143

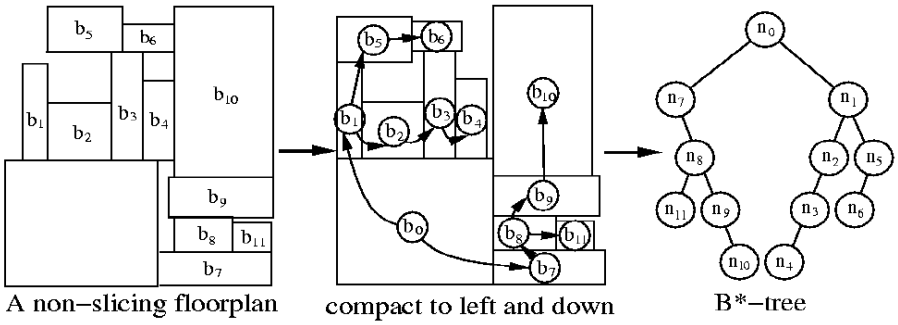
IV. Implementation details

a. The details of your algorithm.

I used b*Tree to implement.

For the initialization, also maintaining b-tree, I have to make sure the left child to attach its parent's right border, and the right child have the same y-coord as his parent.*





First, I do the initial floorplanning, the idea is very simple, I trace the hardblocks of nets, and place one by one from (0,0) as the root.

A trick I do in initialization is that I tried to horizontal each block, to minimize to initial y-bound. Then I calculate the cost and iteratively do the perturbation. When the penalty exceeds the limit, it restore the local best. *(The weight of each constraint is tuning based on the trials.)*

I implement the three kinds of perturbation: *rotate()*, *swap()*, *move()*, while, I did not implement the *flip()* shown on the slide. Each perturbation is decided by each random number. Since I give the initial seed of random, I can reproduce the result.

After each time of perturbation, I update the (x, y)-coordinate of the blocks, and count the cost, and determine whether current result satisfies requirement. Also, I would record the *local-Cost*. If the local best is better than *best-Cost*, then I save the *best-Cost*.

In the end, I restore the result having *best-Cost* and output the final answer.

b. What tricks did you do to speed up your program or to enhance your solution quality

1) First, I try to get a better floorplanning by randomly shuffle the vector of nets. Since I built the initial floorplanning depend on the nets.

2) Second, In my initialization, I tried not to exceed the outline of X.

3) Third, I initially give the seed of rand, to reproduce the similar result in my test, which is I derived for billions of random trials.

4) Last, Parallelization.

c. Please compare your results with the top 5 students' results from last year and show your advantage either in runtime or in solution quality. Are your results better than them?

	Wirelength			Runtime(s)		
Ranks	n100	n200	n300	n100	n200	n300
1	207309	367785	504903	13.97	84.54	263.33
2	209351	379674	521749	25.57	99.49	209.78
3	222513	389041	518157	42.43	282.77	1054.58
4	210220	392175	544879	37.45	105.83	486.73
5	219049	393881	537729	48.65	161.73	435.75
My	**205567**	**388871**	**548968**	**32**	**301**	**301**

Some of my wirelength is better than top-5 from last year's classmates, also, it is worth to note that I can derive the closed wirelength in shorter time.

Overall, I thought my program having high class performance.

d. What have you learned from this homework? What problem(s) have you encountered in this homework?

In the beginning I tried to implement slicing tree, however, I cannot pack it into outline. Hence, I change the method to B*-Tree. But it, as well as, is difficult to maintain. Since I have to perturb it, and during I implementing the perturbation, I have encounter tremendous segmentation error. While it is mainly causing by I have mistakes in updating the tree after the perturbation. Finally, but exceed the deadline, I figure out the problem point. And I can have a good sleep now ><…

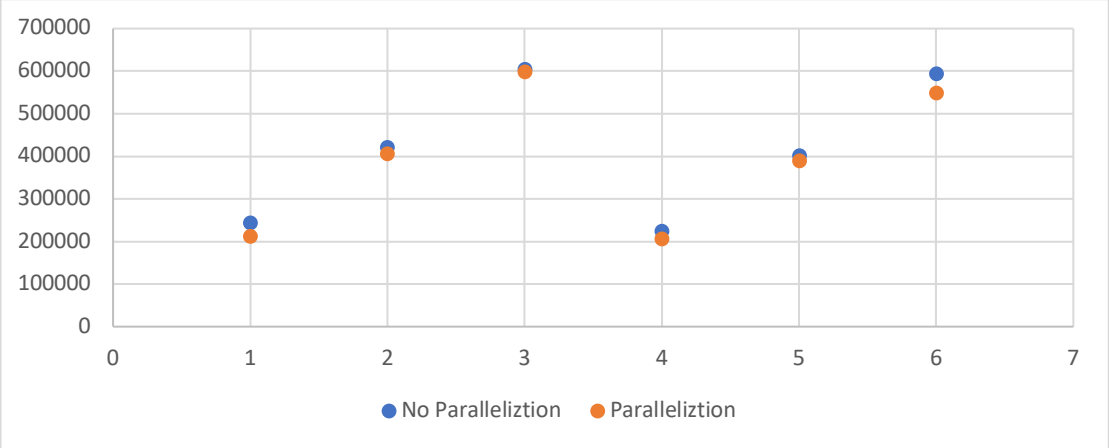
e. (Bonus) Parallelization

In this homework, I implement the parallelization for initial floorplan and perturbations. By using eight threads in initialization, I tries $5x10^5$ times to get the best seed for random. Hence, I can do the perturbation based on a better initialization.

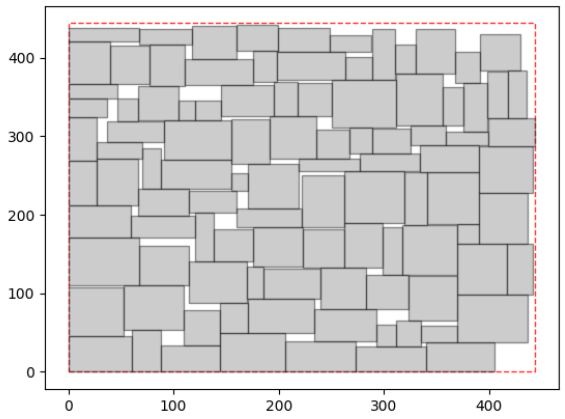
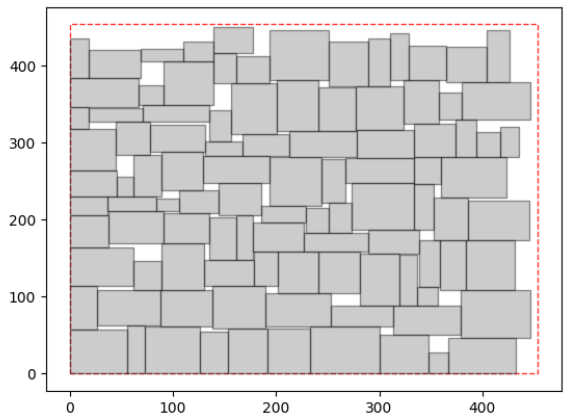
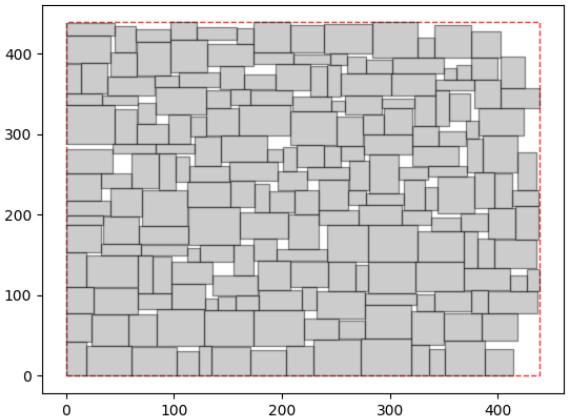
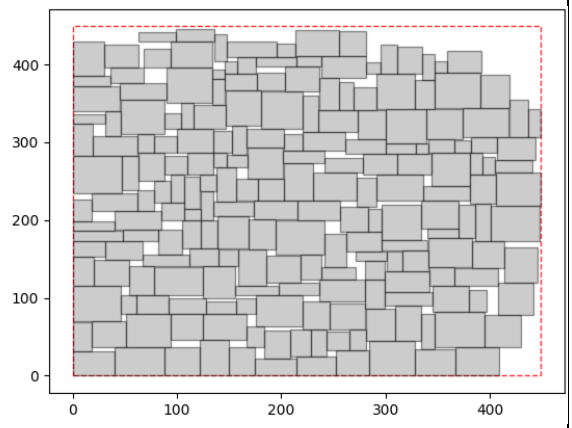
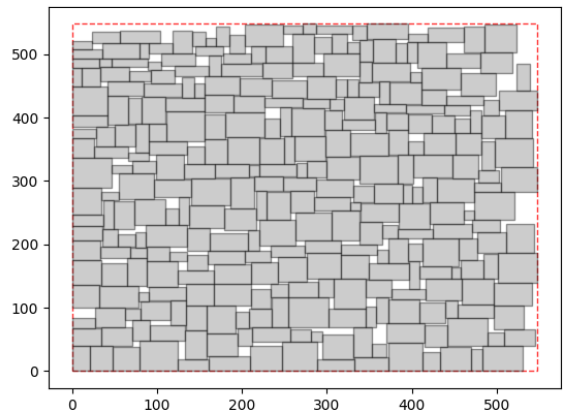
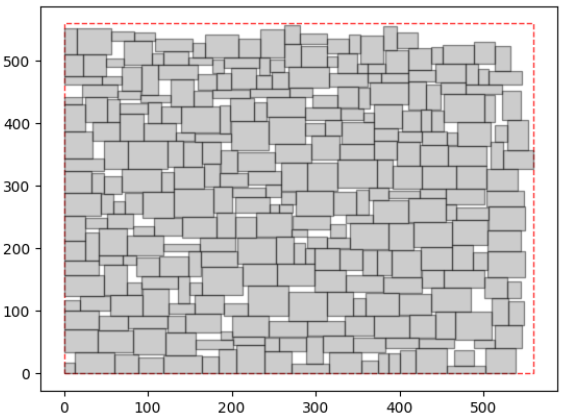
```
22      #pragma omp parallel for num_threads(num_thread)
23      for(int i=0; i<num_thread; ++i)
24      {
25          floorplans[i].initFloorplan();
26          if (i!=0) std::default_random_engine(std::chrono::system_clock::now().time_since_epoch().count());
27          floorplans[i].SA();
28
29          if (!flag_for_thread)
30          {
31              flag_for_thread = true;
32              for (int j = 0; j < num_thread; ++j) floorplans[j].setFlag();
33          }
34      }
```

Also, I record my best Wirelength (bestWL) locally, hence I would join the threads if one of the threads have attained the bestWL.

By doing the parallelization, I could try even more perturbations and it significantly improve my result quality, I show the comparison for parallelization.



f. (Bonus) Visualization of the result of my floorplaning.

Dead space = 0.1	Dead space = 0.15
 n100	 n100
 n200	 n200
 n300	 n300