Physical Design Automation Lab03

313510229 蔡宗儒

1. Pseudocode

a. Legalize()

```
ReadFile(.lg)
Build Cell Map
Insert Cell In Every PlacementRow
Sort Cell In Every PlacementRow
While(ReadFile(.opt))
    While(Get Banking Cell)
        Remove(Banking Cell)
    Insert(New Cell)
```

b. Remove(Banking Cell)

```
Get Row = StartRow
Get EndRow
While(Row <= EndRow)
    For Every Cell in PlacementRow[Row]
        If(Cell Name == Banking Cell’s Name)
            Erase This Cell In PlacementRow[Row]
    Row++
Erase This Cell In Cell Map
```

c. Insert(New Cell)

```
If(Target Space Is Empty)
    Insert New Cell In Cell Map
    Refresh & Sort Cell In Every PlacementRow Relating to this New Cell
Else
    For Row in PlacementRow’s row Size
        While(True)
            For Size in New Cell’s Size
                Get Interval’s Most Left & Most Right point
            If Right - Left >= Cell’s Size
                Calculate Cost
                If(Cost < BestCost)
                    Refresh Candidate Cell
            If(Find All Interval)
                Break
    Insert Candidate Cell In Cell Map
    Refresh & Sort Candidate Cell In Every PlacementRow Relating to this Candidate Cell
```

2. Time Analysis

N is # of Cell
M is # of Placement Row
K is # of New Cell
H is heights of Cell

Remove:	remove cell in map -> O(1)
	remove cell in placement row -> O(H * lg(N / M))
Insert:	O(H * N)
Legalize:	O(K * H * N)

3. Special Features of my program

a. Easy to implement: This method is quite simple: I first check if the target position of the cell can be placed. If it can, I place it directly. If not, I traverse the entire layout starting from the bottom-left corner, find every interval that can fit the cell, calculate the cost, and store the position if its cost is smaller than the current best cost. This ensures that the cost is minimized as much as possible.

b. Structure PlacementRow: I created a structure called PlacementRow. Each PlacementRow contains a vector that stores the cells within it. By sorting this vector, the order of the cells can be determined, making it easy to calculate the size of each interval. Additionally, a variable istIdx is used to keep track of the current cell being processed.

4. Feedback

In this lab, I read through a paper from the University of Hong Kong and developed a preliminary prototype. However, the paper itself wasn't written in great detail. I speculated that it might require using the Corner Stitching method to establish relationships between cells and then define a window and identify the local region. Considering the significant amount of time this would take, I ultimately chose to adopt a direct placement strategy instead.

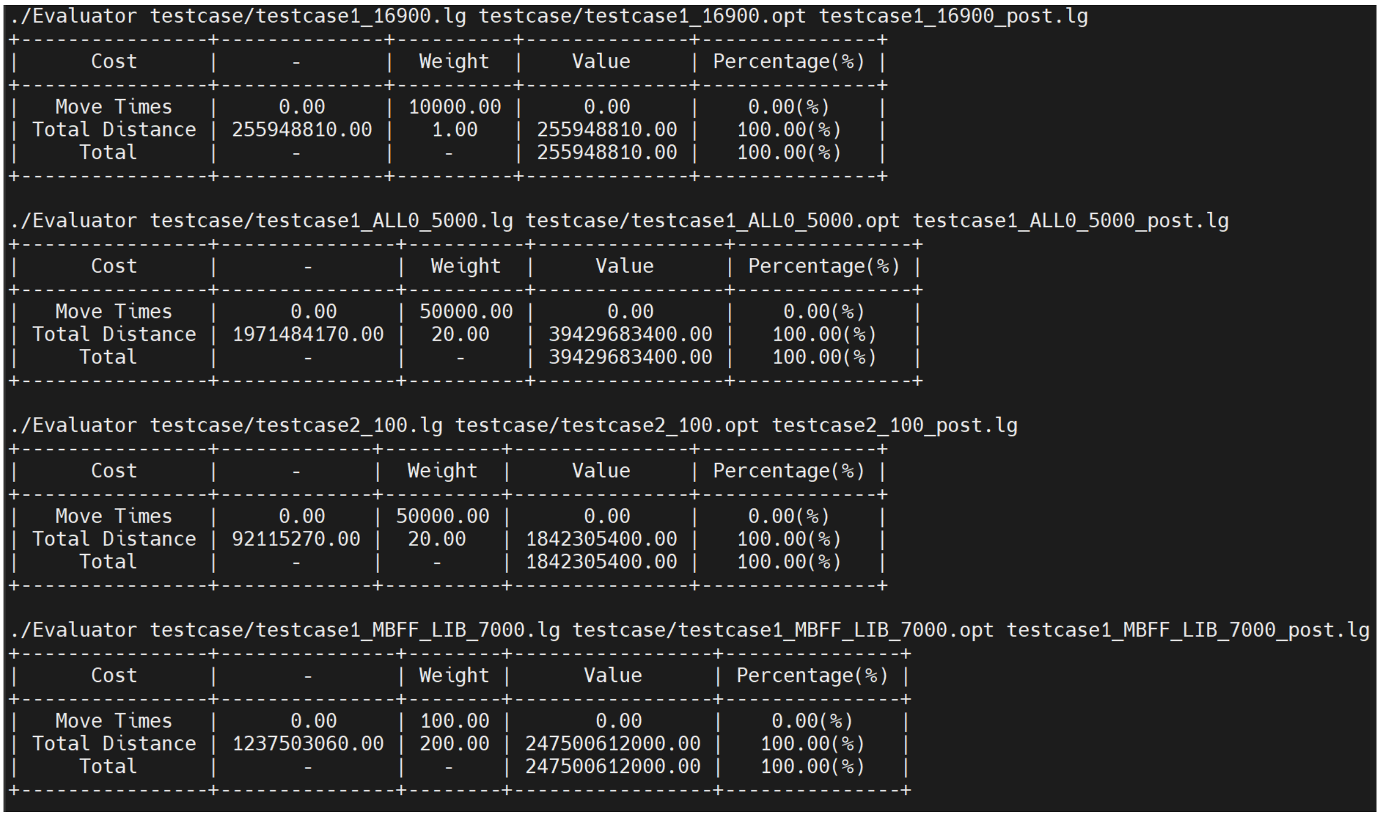
I'm fairly satisfied with my approach because it required very little time to implement, allowing me to maintain a good work-life balance. Compared to other peers who also place cells directly, I incorporated a greedy algorithm to minimize costs as much as possible. While I don't expect my performance to rank among the best, I believe it's not too bad either. In my opinion, this approach offers a very high cost-performance ratio.

5. Conclusion

I implemented a simple method to place cell directly and incorporated a greedy approach to minimize the cost as much as possible. Although I couldn't replicate the more advanced methods from the paper, this approach still allowed me to achieve decent performance while passing all the test cases, including the hellish case.

6. Result

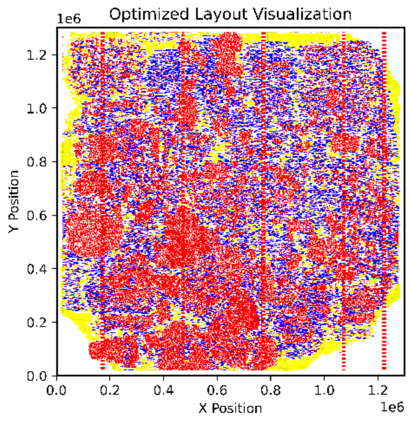
The following are the results:



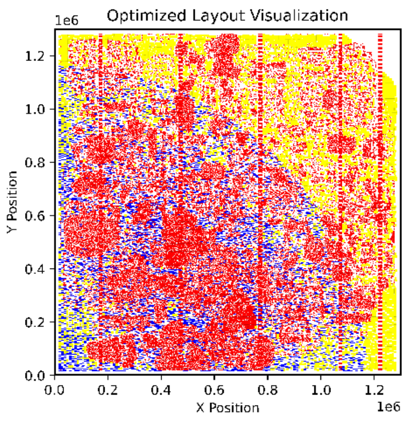
```
./Evaluator testcase/testcase1_16900.lg testcase/testcase1_16900.opt testcase1_16900_post.lg
+----------------+--------------+----------+--------------+---------------+
|      Cost      |      -       |  Weight  |    Value     | Percentage(%) |
+----------------+--------------+----------+--------------+---------------+
|   Move Times   |     0.00     | 10000.00 |     0.00     |    0.00(%)    |
| Total Distance | 255948810.00 |   1.00   | 255948810.00 |   100.00(%)   |
|      Total     |      -       |    -     | 255948810.00 |   100.00(%)   |
+----------------+--------------+----------+--------------+---------------+

./Evaluator testcase/testcase1_ALL0_5000.lg testcase/testcase1_ALL0_5000.opt testcase1_ALL0_5000_post.lg
+----------------+---------------+----------+----------------+---------------+
|      Cost      |       -       |  Weight  |     Value      | Percentage(%) |
+----------------+---------------+----------+----------------+---------------+
|   Move Times   |      0.00     | 50000.00 |      0.00      |    0.00(%)    |
| Total Distance | 1971484170.00 |  20.00   | 39429683400.00 |   100.00(%)   |
|      Total     |       -       |    -     | 39429683400.00 |   100.00(%)   |
+----------------+---------------+----------+----------------+---------------+

./Evaluator testcase/testcase2_100.lg testcase/testcase2_100.opt testcase2_100_post.lg
+----------------+-------------+----------+---------------+---------------+
|      Cost      |      -      |  Weight  |     Value     | Percentage(%) |
+----------------+-------------+----------+---------------+---------------+
|   Move Times   |     0.00    | 50000.00 |      0.00     |    0.00(%)    |
| Total Distance | 92115270.00 |  20.00   | 1842305400.00 |   100.00(%)   |
|      Total     |      -      |    -     | 1842305400.00 |   100.00(%)   |
+----------------+-------------+----------+---------------+---------------+

./Evaluator testcase/testcase1_MBFF_LIB_7000.lg testcase/testcase1_MBFF_LIB_7000.opt testcase1_MBFF_LIB_7000_post.lg
+----------------+---------------+--------+-----------------+---------------+
|      Cost      |       -       | Weight |      Value      | Percentage(%) |
+----------------+---------------+--------+-----------------+---------------+
|   Move Times   |      0.00     | 100.00 |       0.00      |    0.00(%)    |
| Total Distance | 1237503060.00 | 200.00 | 247500612000.00 |   100.00(%)   |
|      Total     |       -       |   -    | 247500612000.00 |   100.00(%)   |
+----------------+---------------+--------+-----------------+---------------+
```

The following chart is plotted using Python, red represents fixed cells, blue represents non-fixed cells, and yellow represents new cells.

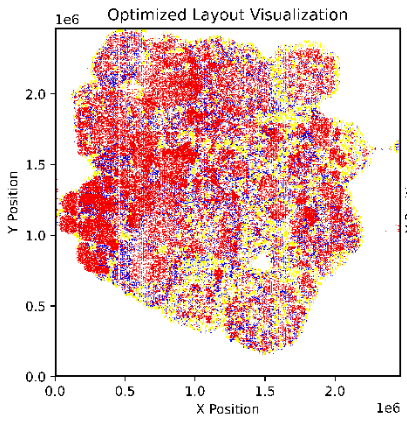
testcase1_16900



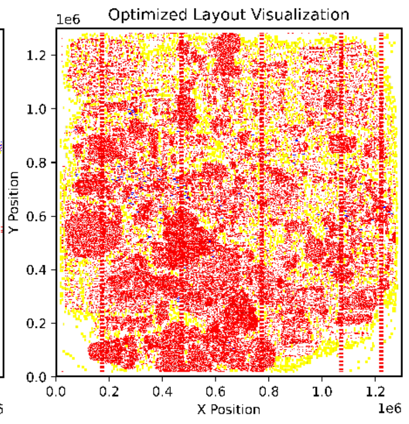
testcase1_ALL0_5000



testcase2_100



testcase1_MBFF_LIB_7000


Optimized Layout Visualization
1e6
Y Position
X Position
1e6