Floorplaning algorithm based on simulated annealing

2024/12/27

1 题目介绍

1.1 问题描述

1.2 题目要求

1.3 输入输出

1.1 问题描述

在经过芯片划分后，整个芯片被分为若干个Block。在简单情况下，这些Blocks均可视为宽高比固定的可移动矩形。芯片上同样摆放着若干Terminal。不同Block之间，Block与Terminal之间会存在相互连接关系，称之为Net。

衡量芯片Floorplan的质量优劣往往采用面积（Area）与线长（Wirelength）两个指标。

芯片的面积（A）为所有Block接后的图形的上、下、左、右边界围成的面积，即能够包裹住Floorplan后所有Block的最小矩形面积。

而芯片的线长（W）为所有Net的半周长线长之和，即：

$$W = \sum_{n_i \in N} HPWL(n_i)$$

其中，N为所有网络的集合，n_i为N中的一个网络。

最终以加权求和的形式来计算芯片的总Cost

$$Cost = \alpha \frac{A}{A_{norm}} + (1-\alpha)\frac{W}{W_{norm}}$$

1.2 题目要求

1.遵循给定的输入输出格式，使用C/C++、python或matlab中一或多种语言编程实现一个简易Floorplanner，算法不限。

2.功能要求：

a)根据输入文件.block中的所有macro信息，将所有block均摆放在给定的Outline范围内，且不允许重叠；

b)在满足功能要求a）的基础上，根据输入文件.block与.net，计算芯片的Cost，并使得芯片Cost最小；

c)在满足功能要求b）的基础上，设计一种方法能够尽可能使得每个网络中相邻的blocks更多，且相邻的边长更长。

3.功能要求中的a)，b)为必做内容，功能要求c) 为选做内容。

4.在报告中需说明程序运行方法与项目目录结构，在文件读写与脚本文件中使用相对路径，不要使用绝对路径。

1.3 输入输出

输入：.block与.net文件

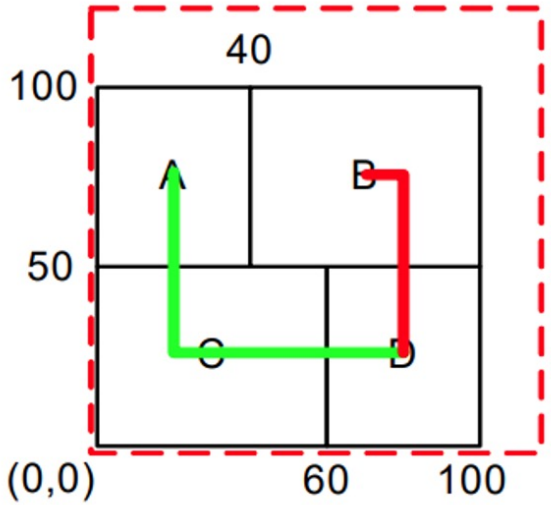
- Block包含有芯片尺寸要求与所有Block与Terminal的输入信息
  - Outline表示芯片的边界信息，最终floorplan后的结果不得超出Outline范围
  - Block 输入信息包含其名称、宽度与高度
  - Terminal 的输入信息包括其名称与坐标
- Net包含有芯片内所有互连关系的要求。一个Net中可能包含若干个Blocks和Terminals。

输出：.output文件

- 最终的Cost
- 总半周长线长HPWL
- 芯片面积
- 芯片的宽与高
- 程序运行时间（秒）
- 模块摆放信息

Input files (input.block):

```
Outline: 120 120
NumBlocks: 4
NumTerminals: 0
A    40   50
B    60   50
C    60   50
D    40   50
```



(input .nets)

```
NumNets: 2
NetDegree: 3
A
C
D
NetDegree: 2
B
D
```

```
5085
170
10000
100  100
0.24
A    0    50   40   100
B    40   50   100  100
C    0    0    60   50
D    60   0    100  50
```

Output files (output.rpt)

2 模拟退火

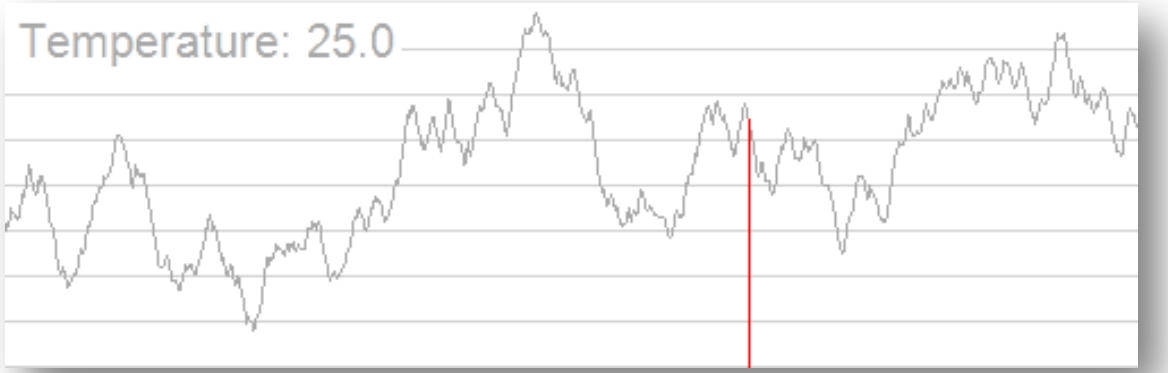
模拟退火（Simulated Annealing，SA）是一种受冶金退火过程启发的概率优化技术，常用于在大型搜索空间中找到全局最小值。

关键概念：

- 温度：控制接受较差解的概率。
- 冷却计划：逐渐降低温度以细化搜索。
- 扰动：应用随机变化以探索解决方案空间。

SA 算法步骤：

- 初始化：以初始解和高温度开始。
  - 迭代：在每个温度下：
  - 应用扰动生成新解。
  - 计算成本差异。
  - 根据接受概率决定是否接受新解。
- 冷却：根据冷却计划降低温度。
- 终止：当温度足够低或达到预定迭代次数时停止。



3 方法实现

3.1 问题建模

3.2 整体流程

3.3 初始构建

3.4模拟退火优化

3.1 问题建模

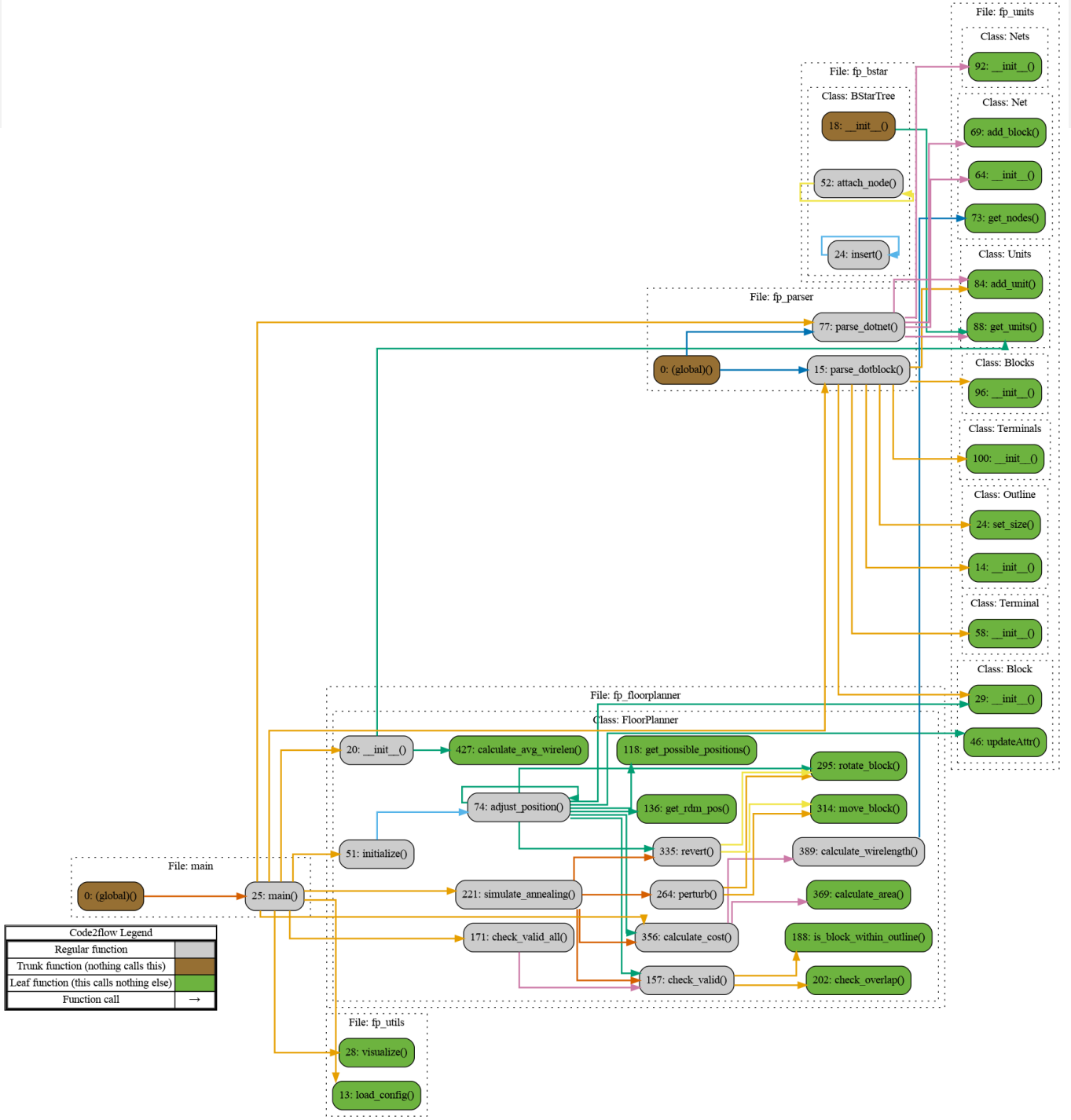
```python
class Block:
    def __init__(self, name=None, width=None, height=None, x=0, y=0) -> None:
        self.name = name
        self.width = width
        self.height = height
        self.x = x
        self.y = y

        # 标记模块是否旋转
        self.rotated = False
        self.rotate_point = 0 # 0: 左下角, 1: 左上角, 2: 右上角, 3: 右下角
        self.placed = False

        # B*-树相关指针
        self.parent = None
        self.left = None
        self.right = None

    def updateAttr(self, block) -> None:
        if isinstance(block, Block):
            self.name = block.name
            self.width = block.width
            self.height = block.height
            self.x = block.x
            self.y = block.y
            self.rotated = block.rotated
            self.rotate_point = block.rotate_point
            self.placed = block.placed
```

```python
class Terminal:
    def __init__(self, name, x, y) -> None:
        self.name = name
        self.x = x
        self.y = y
class Net:
    def __init__(self, name:str, degree:int=0) -> None:
        self.name = name
        self.nodes = []
        self.degree = degree

    def add_block(self, block) -> None:
        self.nodes.append(block)
        self.degree += 1

    def get_nodes(self) -> list:
        return self.nodes
class Outline:
    def __init__(self, width:int=0, height:int=0) -> None:
        self.w = width
        self.h = height

    def set_height(self, height:int) -> None:
        self.h = height

    def set_width(self, width:int) -> None:
        self.w = width

    def set_size(self, width:int, height:int) -> None:
        self.w = width
        self.h = height
```

- Block
- Terminal
- Net
- Outline

3.2 整体流程

1. 输入解析：从输入文件读取块和网络列表信息。
2. 数据结构初始化：为块、终端、网络和外形轮廓创建对象。
3. 初始 Floorplan 构建：构建初始初始可行解。
4. 模拟退火优化：使用 SA 优化 Floorplan。
5. 结果输出：将最终 Floorplan 写入文件并进行可视化。



3.3 初始构建

1. 优先放置大模块，根据每个块的面积（宽度 * 高度）对块进行排序，更好地利用空间并避免后续放置时出现空间不足的问题
2. 第一个从坐标原点开始放置；
3. 后续从已经放置的右边或上边放置
   get_possible_positions(block)
4. 若右边上边冲突，则
   get_rdm_pos

3.4 模拟退火优化

扰动操作（perturb）：

- 随机应用以下操作之一：
  - 旋转：交换块的宽度和高度。
  - 移动：将一个块移动到树中的不同位置(向下/向左)。

成本计算（calculate_cost）：

- 面积：块占用的总面积。
- 线长：所有网络的半周长总和。
- 成本函数：面积和线长的加权和（总cost公式）。

接受准则：

接受具有更低成本的新解。

以概率 P = exp(-ΔCost / Temperature) 接受较差的解。

冷却计划：

使用 Temperature = α * Temperature 更新温度，其中 α 为 0 到 1 之间的冷却率。

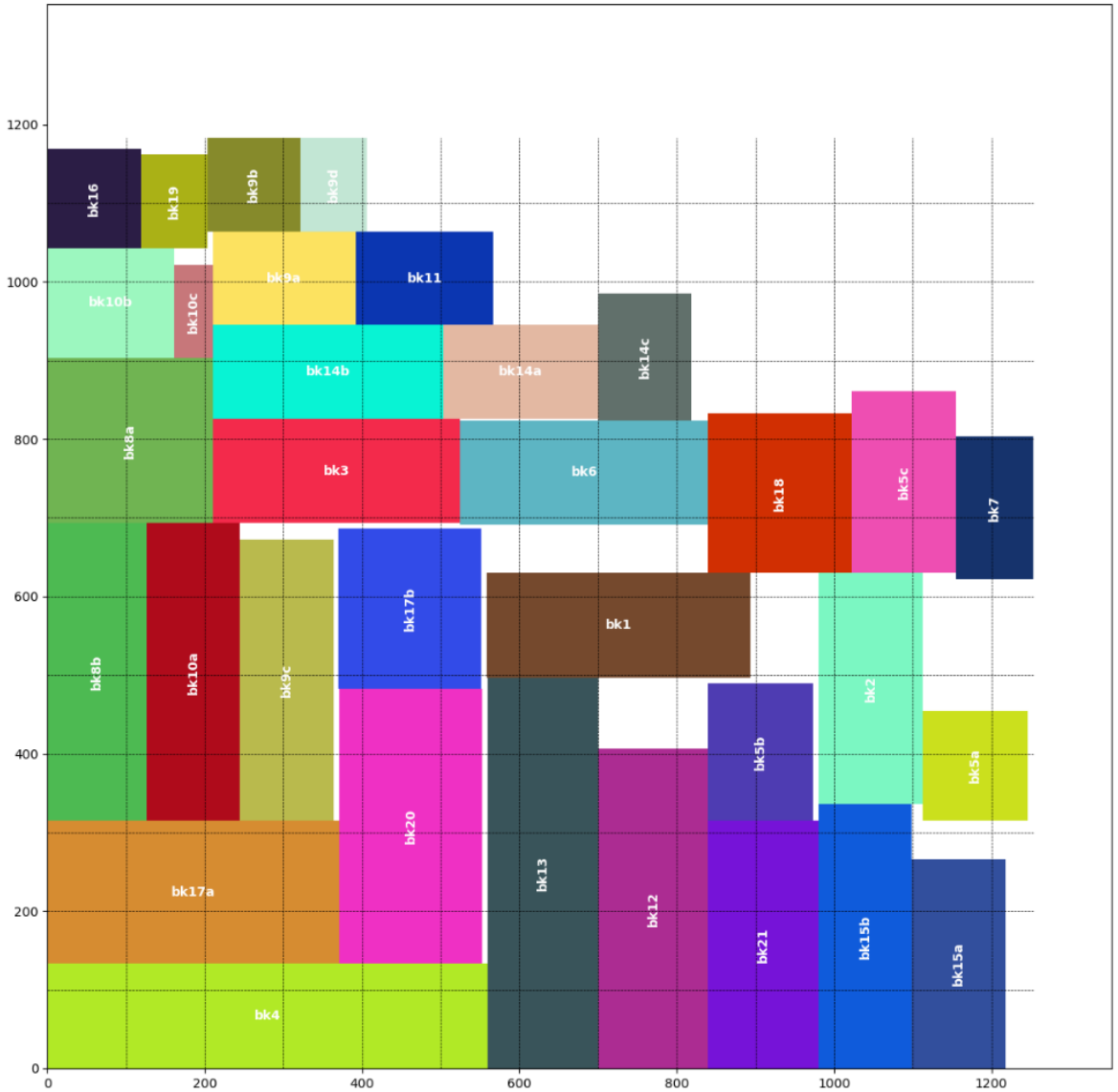
4 实验设定与结果

- 总 Cost 中 α 的取值为 0.5
- 布图优化迭代次数 $iteration$ 的取值为 100000
- 模拟退火中 temperature 取 1000

表：执行结果汇总

Name	Cost	Wirelength	Area	Width	Height	RunTime(s)
ami33	12.13	148237	1482299	1253	1183	0.402
ami49	24.13	1869070	40740560	5320	7658	12.284
test	0.85	140	10000	100	100	0.0004
xerox	40.77	1115624	24416700	6342	3850	0.041

4.1 布图结果 | ami33



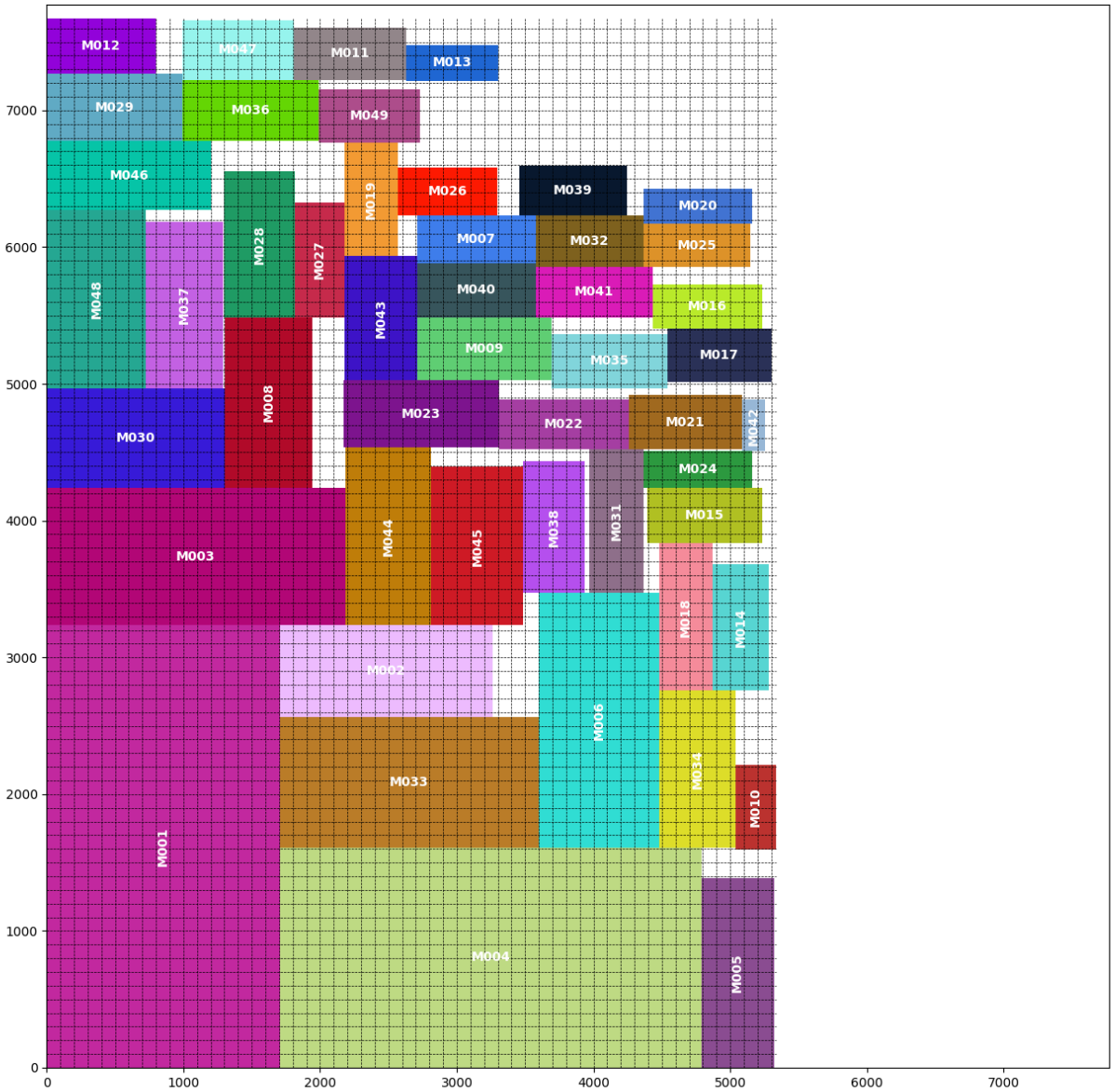
可视化结果

src > output > floorplan_2024-12-26-20:26:41.output

```
Cost 12.132124170930872
Wirelength 148237
Area 1482299
Width 1253
Height 1183
RunTime 0.40212035179138184
bk4 0 0 560 133
bk13 560 0 700 497
bk17a 0 133 371 315
bk20 371 133 553 483
bk12 700 0 840 406
bk8b 0 315 126 693
bk10a 126 315 245 693
bk1 558 497 894 630
bk21 840 0 980 315
bk8a 0 693 210 903
bk9c 245 315 364 672
bk3 210 693 525 826
bk6 525 691 840 824
bk15b 980 0 1099 336
bk2 980 336 1113 630
bk17b 370 483 552 686
```

.output 结果

4.2 布图结果 | ami49



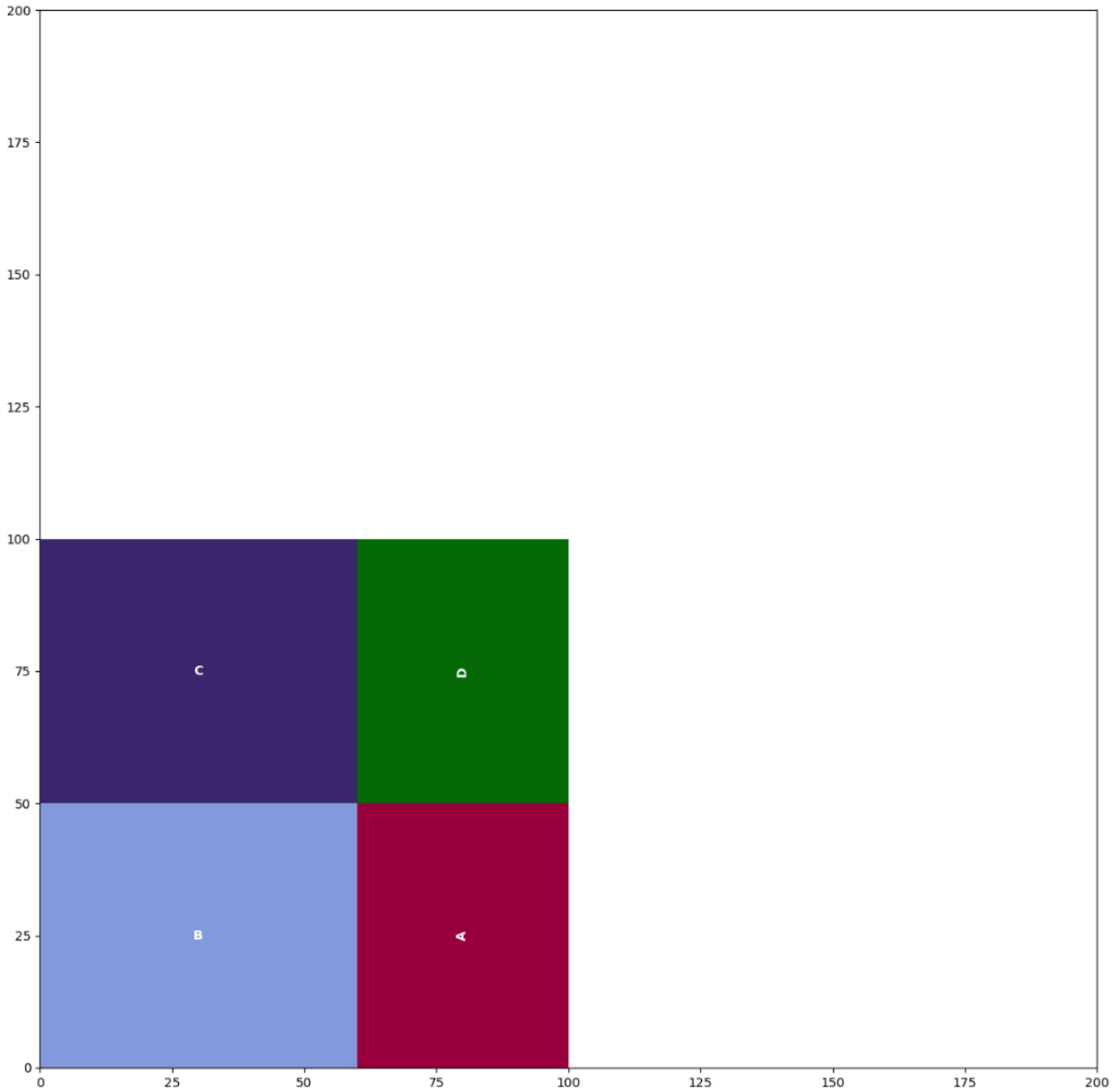
可视化结果

src > output > floorplan_2024-12-26-20:24:47.output

```
Cost 24.133014310403393
Wirelength 1869070
Area 40740560
Width 5320
Height 7658
RunTime 12.283810377120972
M001 0 0 1708 3234
M004 1708 0 4788 1610
M003 1708 1610 3892 2618
M033 0 3234 1890 4186
M006 3892 1610 4774 3472
M002 1890 3178 2562 4732
M030 0 4186 1302 4914
M048 0 4914 1302 5642
M044 2562 3178 3192 4480
M008 3192 3150 3836 4396
M045 2562 4480 3234 5642
M005 4788 0 5320 1386
M037 1708 2618 2926 3178
M034 1302 4186 1862 5334
M046 0 5642 504 6846
M023 4788 1386 5278 2520
M028 504 5642 1022 6706
M029 1302 5334 1792 6328
M043 2926 2618 3836 3150
M009 3836 3472 4816 3934
M036 1862 4732 2310 5726
M038 0 6846 966 7294
```

.output 结果

4.3 布图结果 | test

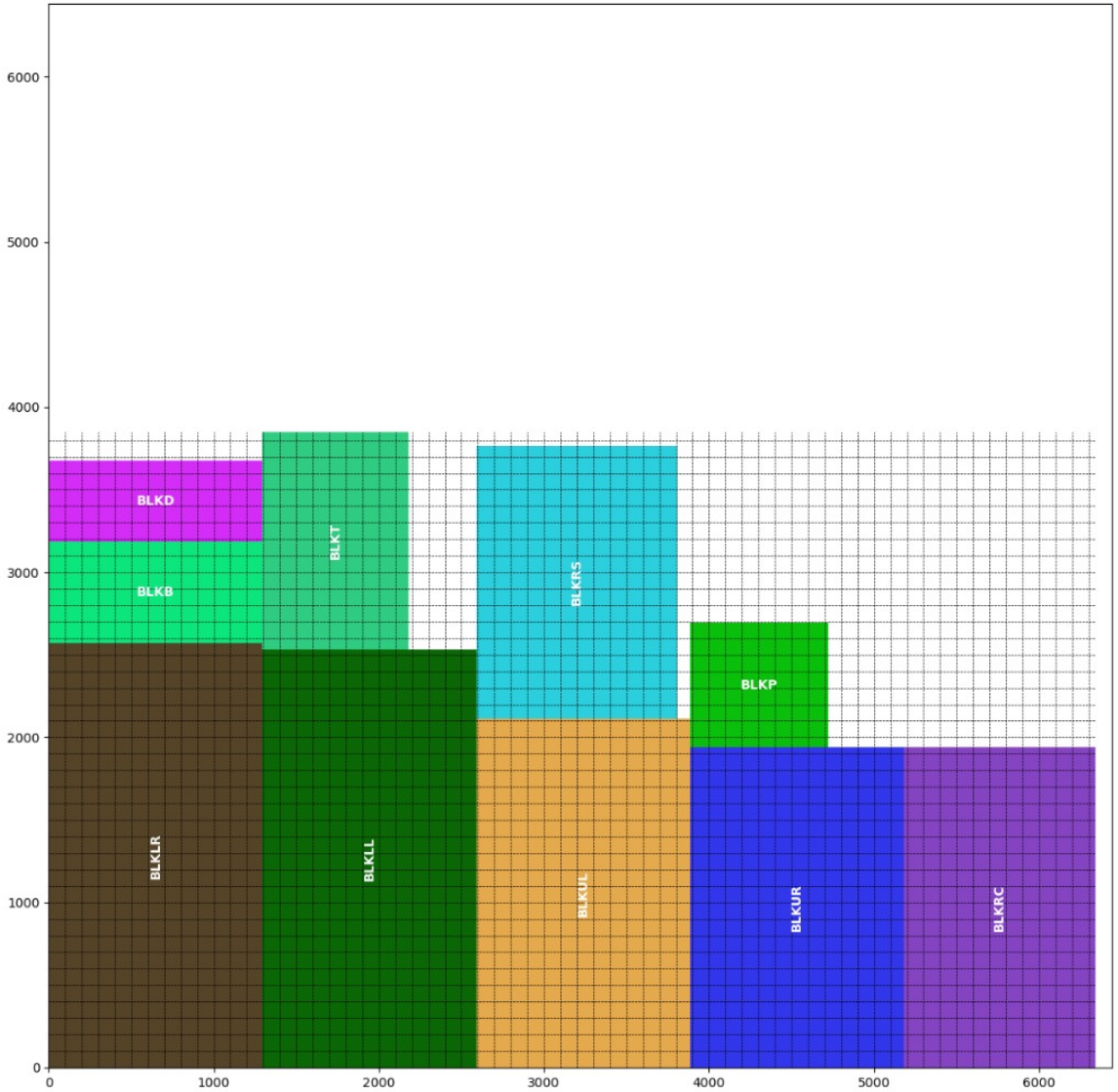



```
Cost 0.85
Wirelength 140
Area 10000
Width 100
Height 100
RunTime 0.00048279762268066406
B 0 0 60 50
C 0 50 60 100
A 60 0 100 50
D 60 50 100 100
```

可视化结果

.output 结果

4.4 布图结果 | xerox



可视化结果

src > output > floorplan_2024-12-26-20:30:32.output

```
Cost 40.76704752508682
Wirelength 1115624
Area 24416700
Width 6342
Height 3850
RunTime 0.040709733963012695
BLKLR 0 0 1295 2569
BLKLL 1295 0 2590 2534
BLKUL 2590 0 3885 2114
BLKUR 3885 0 5180 1939
BLKRC 5180 0 6342 1939
BLKRS 2590 2114 3808 3766
BLKT 1295 2534 2177 3850
BLKB 0 2569 1295 3185
BLKP 3885 1939 4725 2695
BLKD 0 3185 1295 3675
```

.output 结果

5 总结

收获

- 熟悉了 Floorplan 的任务目标和基础流程。
- 学会了使用朴素数据结构和优化算法处理 Floorplan 任务
- 可视化 Floorplan 有助于调试并验证算法的正确性
- 积累了自己的编码经验

参考

[1] S. N. Adya and I. L. Markov, "Fixed-outline Floorplanning through Better Local Search," International Conference on Computer Design (ICCD), 2001.
[2] N. Sherwani, Algorithms for VLSI Physical Design Automation, Springer, 2002.
[3] 丛京生，萨拉费扎德，"芯片布局设计与优化"，IEEE Design & Test of Computers，第14卷，第2期，页码12-25，1997年。