北京郵電大學

Beijing University of Posts and Telecommunications

数字 EDA 理论基础

Floorplan 布图算法

实验报告

姓　　名	Albresky(2024.12.30)
学　　号	
学　　院	
专　　业	
任课教师	

Floorplan 布图算法实验报告

一、 Floorplan 任务描述

1.1 任务背景

在经过芯片划分后，整个芯片被分为若干个模块（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端口（Terminal）。不同 Block 之间，Block 与 Terminal 之间会存在相互连接关系，称之为网络（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

$$W = \sum_{n_i \in N} HPWL(n_i) \tag{1}$$

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：

$$Cost = \alpha \frac{A}{A_{norm}} + (1-\alpha)\frac{W}{W_{norm}} \tag{2}$$

其中α为面积所占的权重，应在 0～1 范围内。A_{norm}、W_{norm} 分别为归一化面积与归一化线长。为简化计算，不妨令A_{norm}为所有 Block 面积之和；而W_{norm}为所有网络中的每个 Block 平均边长之和。

1.1 题目要求

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。

- **功能要求：**

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；

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；

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。

功能要求中的 a)，b)为必做内容，功能要求 c) 为选做内容。

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**。

- **输入输出说明：**
  - **输入：**

输入文件有两个，分别为.block 与.net 文件。其输入格式与说明如下：

- .block

```
Outline: <outline width, outline height>
NumBlocks: <# of blocks>
NumTerminals: <# of terminals>

<macro name> <macro width> <macro height>
… More macros

<terminal name> terminal <terminal x coordinate> <terminal y coordinate>
… More terminals
```

输入文件 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，即上图中的 macro，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

```
NumNets: <# of nets>
NetDegree: <# of terminals in this net>
<terminal name>
… More terminal names
 <macro name>
 … More macros

… More “NetDegree” and “terminal name”
```

输入文件 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：**

输出文件为.output 文件，其格式与说明如下：

- .output

<final cost>
// $Cost = \alpha A + (1-\alpha)W$
<total wirelength>
// $W = \sum_{n_i \in N} HPWL(n_i)$
<chip_area>
// area = (chip_width) * (chip_height)
<chip_width> <chip_height>
//resulting chip width and height
<program_runtime>
//report the runtime in seconds
<macro_name> <x1> <y1> <x2> <y2>
<macro_name> <x1> <y1> <x2> <y2>
// (x1, y1): lower-left corner, (x2, y2): upper-right corner
… More macros

输出包含六个部分：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），以及模块摆放信息。

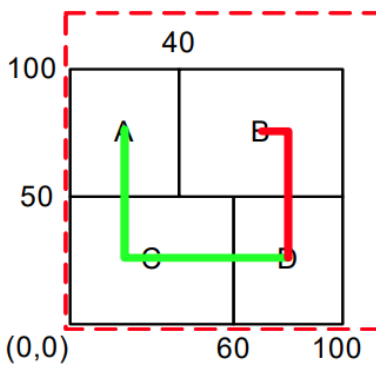
- 可视化

可视化仅为方便调试与完成报告的工具，不计入考核内容。可直接使用或参照实例程序 draw.py 完成布图结果可视化。

eg. 输入输出文件与 floorplan 结果示例：

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)

二、 问题建模

2.1 线长估计

对于双引脚线线长（Total wirelength with net weights），可以采用欧式距离，即下面的公式进行定义：

$$d_M(P_1, P_2) = |x_2 - x_1| + |y_2 - y_1| \tag{3}$$

而多引脚线长，本实验采用半周长线长进行估计，其示例如下：

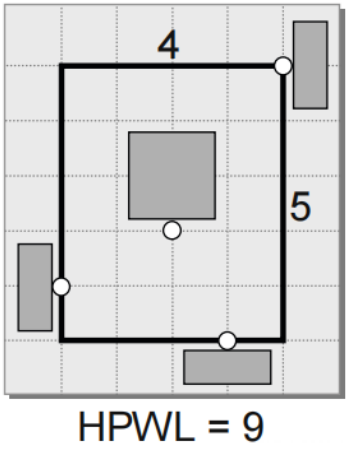


图 1：半周长线长 HPWL 示例

对于每个模块 Block 的引脚位置，本实验采用矩形面积相对较小的点进行表示。对于水平方向的 Block，取最左侧的 Block 的右侧边的中心点作为 HPWL 矩形的左侧边定位点，取最右侧 Block 的左侧边的中心点作为 HPWL 矩形的右侧边定位点；对于垂直方向的 Block，取最上侧的 Block 的下侧边的中心点作为 HPWL 矩形的上侧边定位点，取最下侧 Block 的上侧边的中心点作为 HPWL 矩形的下侧边定位点。

2.2 布图初始化策略

对于所有单元（Block，Net，Terminal），本实验对其分别定义了独立的类，并定义了如 name，x，y，width，height，rotated，placed 等属性用以标记单元，所有的单元均以 Python 列表的数据结构进行组织。

实验提供的.net、.block 等单元文件在通过手动实现 parser 构造上述的数据结构后，既可以开始布图初始化。

首先建立 outline 的直角坐标系，每个 Block 均以其左下角的顶点作为其定位的锚点。整体初始化策略为：

1）将所有 Block 按自身的面积属性递减排序。后续初始化均按排序后的

Block 顺序依次放置；

2）第 0 个 Block 以随机方向放置至坐标系原点，并置 placed 标记位属性为 True；

3）从已放置的$(i-1)$个 Blocks 中随机选取 1 个作为参考 Block，第i $(i>1)$个 Block 放置至参考 Block 的上侧边或右侧边；若从已放置的 Blocks 中无法找到合法解，则在 Outline 内部随机放置；若 i==num(Blocks)，则进行步骤 5）；

4）检查合法性，如果步骤 3）放置的 Block 超出 Outline 边界，或与其他已放置的 Blocks 产生重叠（面积有交集），则回到步骤 3）重新放置该 Block；否则置该 Block 的 placed 标记位属性为 True，然后回到步骤 3）进行下一个 Block 块的放置；

5）结束。

这种初始化策略的前提是，给定的.block 文件所描述的问题本身有解，否则初始化将无限循环。

三、 基于模拟退火的布图优化算法

3.1 算法简介

模拟退火（Simulated Annealing，SA）是一种受冶金退火过程启发的概率优化算法，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：根据冷却方法降低温度；
  - 终止：当温度足够低或达到预定迭代次数时停止。

3.2 优化策略

- **扰动：**

  随机应用以下操作之一：

  - 旋转：交换 Block 的宽度和高度（频率为 1:10）；
  - 移动：随机选择 Block，将其向左或下（靠近坐标系原点的方向）以步长 1 平移（频率为 9:10）。

由于旋转操作的变化数量只有 2 种；而移动操作在步长为 1 时，可能存在多次可行的移动。因此在随机进行扰动操作时，它们二者的操作频率比值为 1:9。

- **成本**计算：
  - 面积：Block 占用的总面积；
  - 线长：所有 Net 的半周长总和；
  - 成本函数：面积和线长的加权和。

  面积和线长的估计方法见第一章公式（1）和公式（2）。

- **接受**新解：

  1）Cost 更低，则接受新解；

2）否则，以概率 $P = e^{-\frac{\Delta Cost}{T}}$ 接受较差的解，其中，T 为模拟退火的温度参数。

为了避免陷入局部最优解，接受新解时以概率 P 接受了较差的解。

- **冷却**方法：

使用 $T = \alpha \cdot T$ 更新温度，其中冷却率 α 介于 0 到 1 之间的浮点数。

算法使用 Python 实现，未调用三方 Floorplan 库函数。其函数调用如图 2 所示。

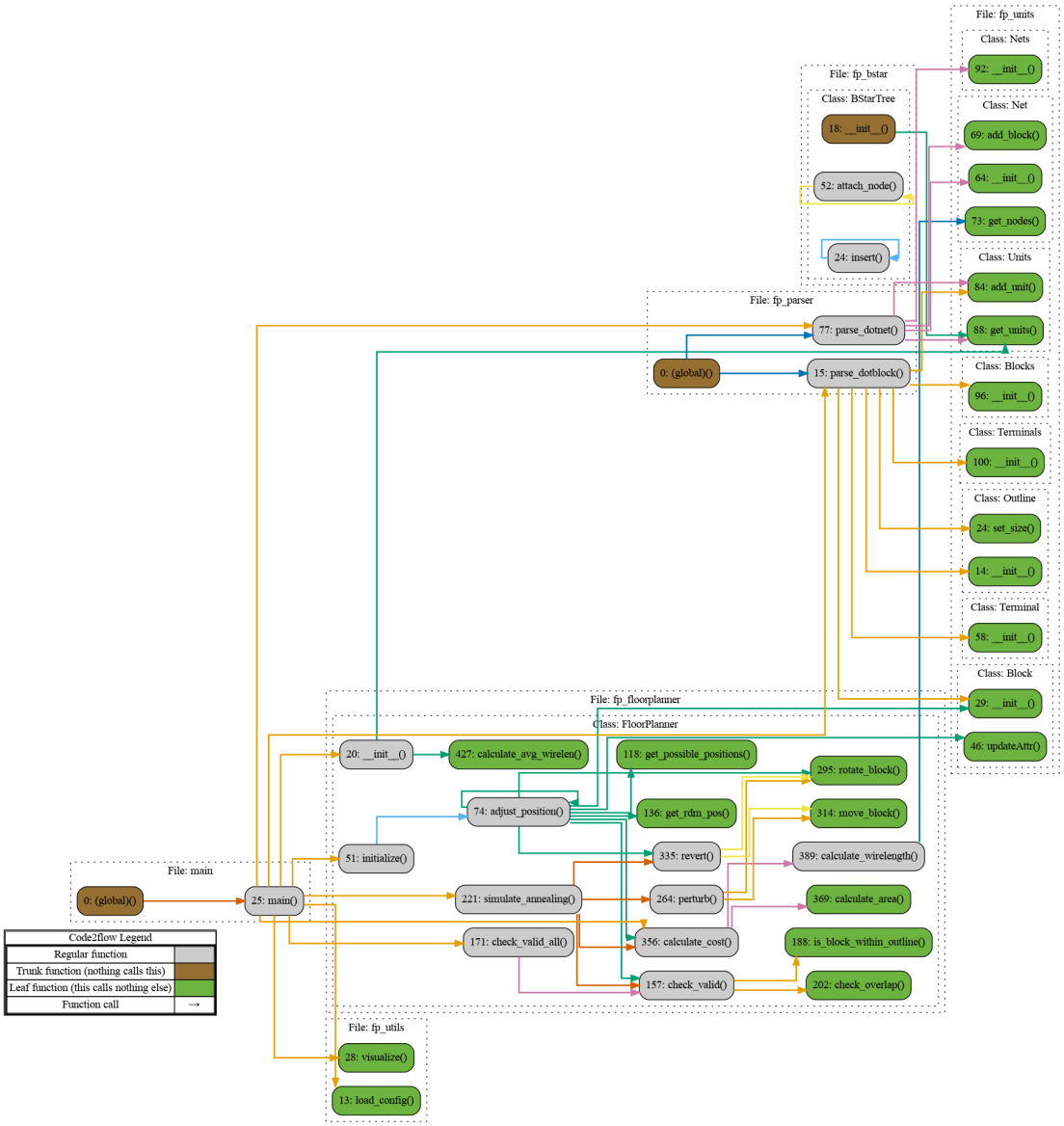


图 2：Floorplan 算法实现 Function Call Graph

四、 实验

4.1 实验设置

本实验设置成本公式的 $\alpha = 0.5$；模拟退火的冷却率 $\alpha = 0.5$，退火最大迭代次数为 10000 次。这里，当退火在连续 10 次的迭代中获得相等的成本时，认定为**收敛到最优解**。由于随机初始化会影响最终的布图结果，本报告的部分实验的结果与课上 Slides 中汇报的数据存在极小的误差，可忽略。

实验运行的系统环境为 Ubuntu 22.04.5 LTS x86_64，CPU 处理器为 Intel Xeon Platinum 8375C (128) @ 3.500GHz，RAM 大小为 504 GB，Python 解释器版本为 3.10.12。

4.1 实验结果分析

表 1：测试数据的 Floorplan 实验结果汇总

测试数据	成本	线长	面积	总宽度	总高度	运行时长 (s)
test	0.85	140	10000	100	100	0.0004
ami33	12.13	148231	1482299	1253	1183	0.453
ami49	24.14	1869938	40740560	5320	7658	7.004
xerox	40.77	1115624	24416700	6342	3850	0.030

上表汇总了本实验提供的测试数据的 Floorplan 布图结果数据，各实验的布图结果可视化后，如下图所示。

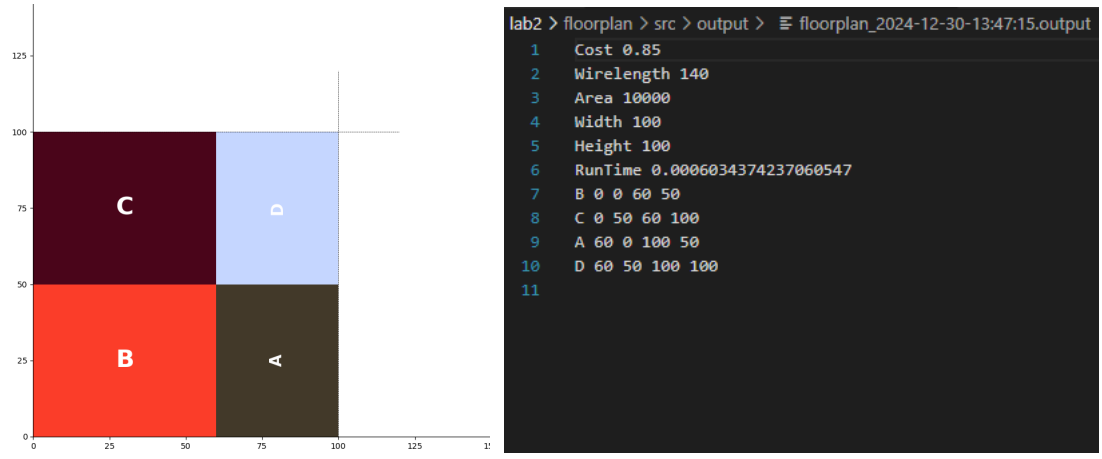
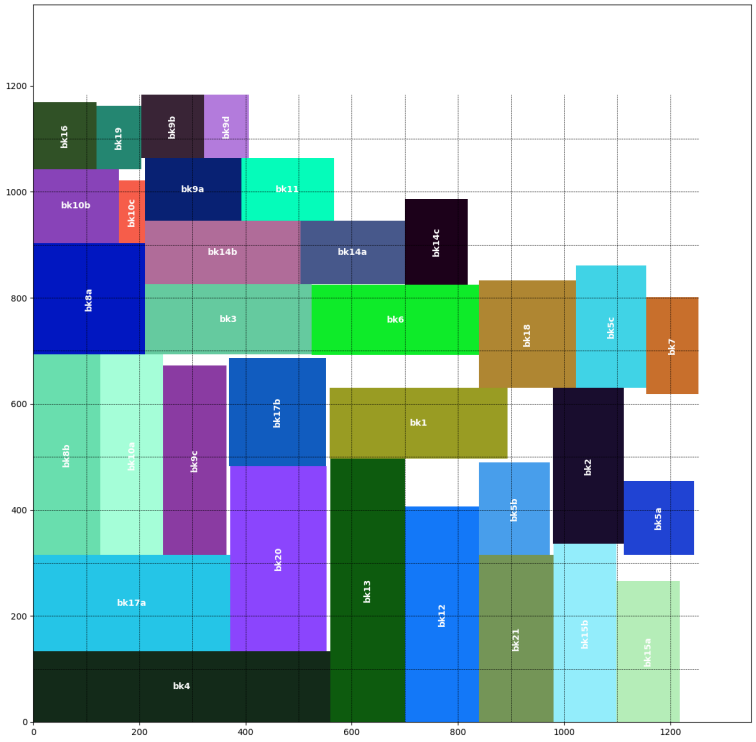


图 3：test 测试数据的布图结果

如图 2 所示，在 test 测试数据中，本实验找到了理论最优解，其总面积为 10000，线长为 140。



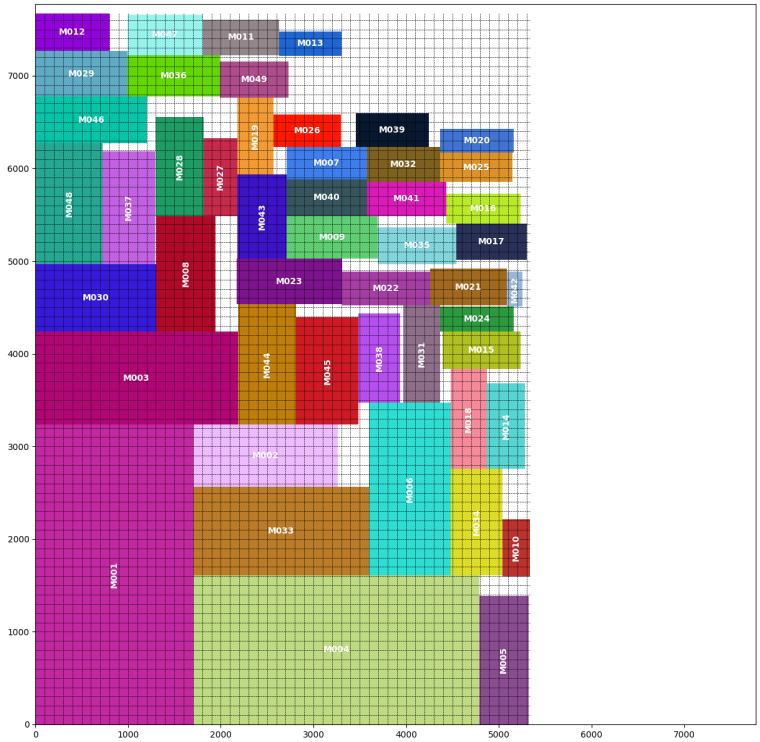
lab2 > floorplan > src > output > floorp

```
Cost 12.131659054651804
Wirelength 148231
Area 1482299
Width 1253
Height 1183
RunTime 0.45251941680908203
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 525 692 840 825
bk15b 980 0 1099 336
bk2 979 336 1112 630
bk17b 369 483 551 686
bk18 840 630 1022 833
bk14b 210 826 504 945
bk15a 1099 0 1218 266
bk5c 1022 630 1155 861
bk14a 504 826 700 945
bk5b 840 315 973 490
bk10b 0 903 161 1043
bk9a 210 945 392 1064
bk11 392 945 567 1064
bk14c 700 825 819 986
bk5a 1112 315 1245 455
bk7 1155 619 1253 801
bk16 0 1043 119 1169
bk9b 203 1064 322 1183
bk19 119 1043 203 1162
bk9d 322 1064 406 1183
bk10c 161 903 210 1022
```

```
SA[166] Cost=12.131659054651804
SA[167] Cost=12.131659054651804
SA[168] Cost=12.131659054651804
SA[169] Cost=12.131659054651804
SA[170] Cost=12.131659054651804
SA has converged at iteration 171 with cost 12.131
SA finished, 33
```

图 4：ami33 测试数据的布图结果

在 ami33 数据中，本方案迭代 171 次后找到最优解。其总成本约为 12.13，总线长为 148231，总面积为 148299，程序运行时长为 0.453 秒。




```
lab2 > floorplan > src > output > ≡ floo
  1   Cost 24.14395484331322
  2   Wirelength 1869938
  3   Area 40740560
  4   Width 5320
  5   Height 7658
  6   RunTime 7.003932237625122
  7   M001 0 0 1708 3234
  8   M004 1708 0 4788 1610
  9   M003 1708 1610 3892 2618
 10   M033 0 3234 1890 4186
 11   M006 3892 1610 4774 3472
 12   M002 1890 3178 2562 4732
 13   M030 0 4186 1302 4914
 14   M048 0 4914 1302 5642
 15   M044 2562 3178 3192 4480
 16   M008 3192 3150 3836 4396
 17   M045 2562 4480 3234 5642
 18   M005 4788 0 5320 1386
 19   M037 1708 2618 2926 3178
 20   M034 1302 4186 1862 5334
 21   M046 0 5642 504 6846
 22   M023 4788 1386 5278 2520
 23   M028 504 5642 1022 6706
 24   M029 1302 5334 1792 6328
 25   M043 2926 2618 3836 3150
 26   M009 3836 3472 4816 3934
 27   M036 1862 4732 2310 5726
 28   M038 0 6846 966 7294
 29   M018 3234 4396 4312 4788
 30   M031 3234 4788 3626 5838
 31   M014 3836 3934 4760 4340
 32   M047 1022 6328 1456 7140
 33   M022 4774 2520 5138 3472
 34   M015 1792 5726 2198 6566
 35   M040 966 7140 1834 7532
 36   M035 2310 5642 2702 6496
 37   M012 4816 3472 5222 4270
 38   M019 3626 4788 4018 5614
 39   M021 4760 4270 5152 5096
 40   M011 4312 4340 4690 5166
 41   M041 2968 5838 3822 6202
 42   M027 0 7294 840 7658
 43   M007 4284 5166 4634 6034
 44   M017 1456 6566 2212 6958
 45   M032 4690 5096 5068 5880
 46   M049 2212 6496 2954 6888
 47   M039 1834 6958 2618 7322
 48   M016 2954 6454 3276 7252
 49   M026 3276 6454 3626 7182
 50   M025 3822 5782 4144 6566
 51   M024 4018 4788 4284 5586
 52   M020 2968 6202 3766 6454
 53   M010 4634 5880 4928 6496
 54   M013 2702 5642 2968 6314
 55   M042 3626 5614 4004 5782
```

```
SA[1340] Cost=24.133014310403393
SA[1341] Cost=24.133014310403393
SA[1342] Cost=24.133014310403393
SA[1343] Cost=24.133014310403393
SA[1344] Cost=24.133014310403393
SA[1345] Cost=24.133014310403393
SA[1346] Cost=24.14395484331322
SA[1347] Cost=24.14395484331322
SA[1348] Cost=24.14395484331322
SA[1349] Cost=24.14395484331322
SA[1350] Cost=24.14395484331322
SA[1351] Cost=24.14395484331322
SA[1352] Cost=24.14395484331322
SA[1353] Cost=24.14395484331322
SA[1354] Cost=24.14395484331322
SA has converged at iteration 1355 with cost 24.1
SA finished, 49
```

图 5：ami49 测试数据的布图结果

在 ami49 数据中，本方案迭代 1355 次后找到最优解。其总成本约为 24.14，总线长为 1869938，总面积为 40740560，程序运行时长为 7.004 秒。

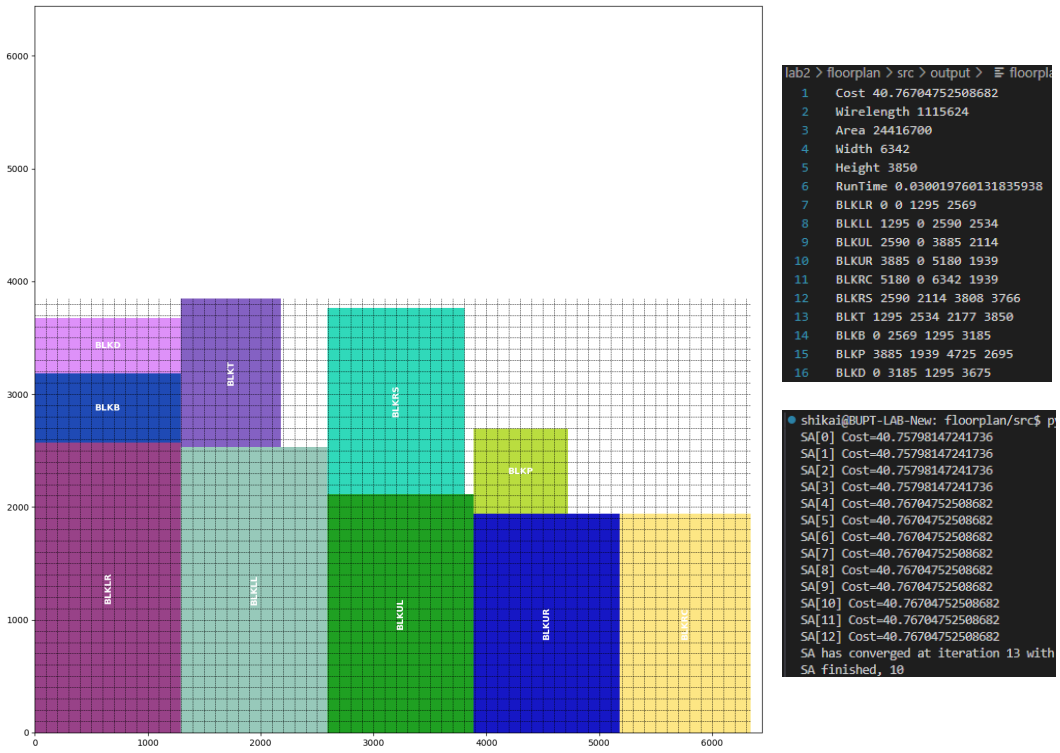


图 6：xerox 测试数据的布图结果

在 xerox 数据中，本方案迭代 19 次后找到最优解。其总成本约为 40.77，总线长为 1115624，总面积为 24416700，程序运行时长为 0.300 秒。

参考文献

[1] Adya S N, Markov I L. Fixed-outline floorplanning through better local search[C]//Proceedings 2001 IEEE International Conference on Computer Design: VLSI in Computers and Processors. ICCD 2001. IEEE, 2001: 328-334.
[2] Sherwani N A. Algorithms for VLSI physical design automation[M]. Springer Science & Business Media, 2012.
[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。
[4] kkkjfg. 第四章 Global and Detailed Placement [VLSL Physical Design 学习笔记][EB/OL]. 2023.4[2024.12]. https://blog.csdn.net/kkkjfg/article/details/128877914.