集成电路EDA设计精英挑战赛2020

作品设计报告

赛题三：智能MPW拼接

参赛队ID：EDA20315

二〇二〇年十一月

一、赛题成果

1.1 成果汇总

① 作品设计报告，设计文档详见六、附录（一）；

② 10 个测试算例的可视化结果，详见七、附录（二）；

③ 服务器代码路径：/home/eda20315/SmartMPW

1.2 赛题介绍

本赛题智能 MPW 拼接描述为：输入 N 个不同形状和尺寸的多边形（相当于版图边界外形)，多边形为矩形或边均为正交方向的多边形（L 字形、T 字形、凹字形），要求输出各个多边形的最终摆放位置。目标函数为最小化拼接形成的包络矩形的面积，在面积相同的情况下，长宽比越接近 1：1 的矩形更优。拼接规则为：1）不允许重叠；2）允许做基本的几何旋转；3）拼接成的包络矩形满足 50um≤宽≤300um，50um≤长≤400um。

本次比赛官方共提供了 10 个测试算例，我们统计了各个算例的规模和形状分布情况，得到下表 1.1。

表 1.1 算例情况统计

算例名	矩形数目	L 字形数目	T 字形数目	总数	总面积
polygon_area_etc_input_1	20	10	5	35	1929
polygon_area_etc_input_2	22	10	6	38	2243
polygon_area_etc_input_3	47	29	9	85	4997
polygon_area_etc_input_4	106	50	26	182	8828
polygon_area_etc_input_5	169	98	34	301	16959
polygon_area_etc_input_6	93	54	18	165	27578
polygon_area_etc_input_7	44	22	11	77	4630
polygon_area_etc_input_8	38	21	6	65	7575
polygon_area_etc_input_9*	103	40	16*	159	9812
polygon_area_etc_input_10	39	21	8	68	3606

* polygon_area_etc_input_9 所包含的 16 个 T 字形中有 15 个凸 T 字形和 1 个凹字形。

*所有算例文件都经过预处理，去除了重复和共线的坐标点。

从表 1.1 可知，polygon_area_etc_input_5 的规模最大，polygon_area_etc_input_1 的规模最小，polygon_area_etc_input_9 包含一个凹字形；所有的算例都是矩形数目最多，L 字形次多，T 字形最少。以上这些分析对后续的算法设计具有指导意义。

1.3 计算结果

经过测试，我们的算法得到以下结果，见表 1.2。

表 1.2 计算结果

算例名	L	W	Area(L×W)	利用率(%)	$\frac{max\{L,W\}}{min\{L,W\}}$
polygon_area_etc_input_1*	59	35	2065	93.414	1.686
polygon_area_etc_input_2*	50	48	2400	93.458	1.042
polygon_area_etc_input_3	73	71	5183	96.411	1.028
polygon_area_etc_input_4	78	116	9048	97.569	1.487
polygon_area_etc_input_5	122	142	17324	97.893	1.164
polygon_area_etc_input_6	159	181	28779	95.827	1.138
polygon_area_etc_input_7	62	78	4836	95.74	1.258
polygon_area_etc_input_8	80	99	7920	95.644	1.238
polygon_area_etc_input_9	130	78	10140	96.765	1.667
polygon_area_etc_input_10	60	63	3780	95.397	1.05

* L：包络矩形长度（水平方向），W：包络矩形宽度（垂直方向），Area：包络矩形面积，利用率 = $\frac{多边形总面积}{Area}\times 100\%$。

* polygon_area_etc_input_1 和 polygon_area_etc_input_2 规模较小，拼接形成的包络矩形面积不足 50×50。

从上表 1.2 可以看出，我们的算法成功求解了比赛官方提供的 10 个算例，除了 polygon_area_etc_input_1 和 polygon_area_etc_input_2 算例规模较小，改进空间有限，其余 8 个算例均求得利用率为 95%以上，其中最大算例 polygon_area_etc_input_5 求得利用率为 97.893%，polygon_area_etc_input_9（包含一个凹字形）求得利用率为 96.765%，而且包络矩形的长宽比基本限制在 1.5 左右。

此外，我们对拼接结果作了可视化输出，详见七、附录（二）。

二、主要技术路线及实现方式

2.1 拼接问题研究现状综述

根据比赛官方给出的参考资料，我们自己也搜集并阅读了相关文献，发现 MPW 拼接问题与矩形背包、布局规划、俄罗斯方块等经典学术问题有相似之处，如：1) 2D-Bin-Packing[1][2]、2) Floor Planning[3][4]、3) Chip Placement[3]、4) Tetris，它们的目标函数都是最小化包络矩形面积，即最大化利用率。此外，与芯片设计相关的问题可能还会涉及线长、温度等优化目标[3]。在这些问题中，俄罗斯方块针对的是像素化的多边形（也被称作 Polyominoes 问题），而其他大多数问题的放置对象都是矩形。

本次比赛的多边形增加了 L 字形、T 字形、凹字形，其中 T 字形也并非左右对称等宽的规则 T 字形，这增加了问题难度。因此，如何合理有效地给这类形状找到最佳放置位置，是本次比赛问题的重点难点所在。

赛题解析中官方为我们提供了两种方案：1）以早期 Floorplan 中 Stockmeyer 算法为代表的动态规划方法[4]，2）谷歌基于强化学习的 Macro 布局思路[5]。我们通过调研发现启发式算法也是解决拼接问题的常用方法之一，常用的思路有两种：

① 直接应用经典的元启发式算法，如模拟退火（Simulated Annealing）、局部搜索（Local Search）、遗传算法（Genetic Algorithm）等，这种思路的关键点在于给布局方案构造一个合理的解的表达形式，比如相关文献中针对 Floorplan 提出的 Sequence Pair、O-tree[6]和 B*-tree[7]等。一个合理的表达形式不仅能在满足约束的前提下准确地映射布局方案，而且能应用上述元启发式算法高效地探索解空间，在一定时间内找到局部甚至全局最优解；

② 固定包络矩形的长度将问题转化为 2DSP（2D strip packing）[7]，或者同时固定长度和宽度转为 2DRP（2D rectangle packing）[8]。2DSP 和 2DRP 是 cutting and packing 领域研究已久的经典问题，相关成熟的技术有很多。在这种思路下，对解的表达形式要求并不严格，而是要求在放置过程中能维护当前的放置状态（packing pattern），如角、边等信息，skyline 就是一种著名的 packing pattern。

在本次比赛中，我们主要使用了②，即基于 2DSP+skyline 的思路实现算法。

2.2 自适应选择搜索框架（ASA）

我们参考了论文《An adaptive selection approach for the 2D rectangle packing area minimization problem》[1]中提出的 Adaptive selection approach，采用固定长度的思路，结合赛题具体要求对其进行改造形成了算法最终的搜索框架，如图 2.1 所示。

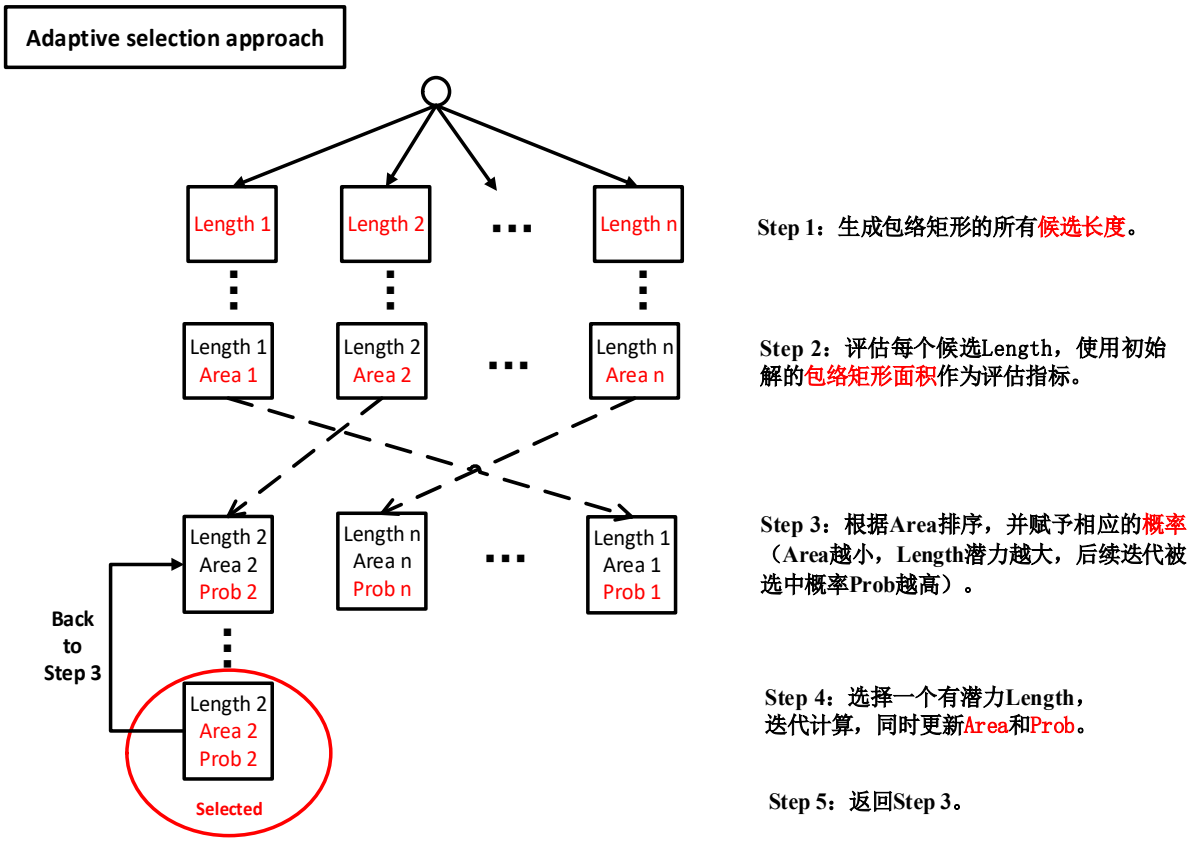

图 2.1 改造后的 Adaptive selection approach 搜索框架

赛题要求最终拼接形成的包络矩形满足 50um≤宽≤300um，50um≤长≤400um，我们以 1um 为单位间隔等距地产生候选长度，则候选长度集合为$\{50, 51, \ldots, 400\}$。考虑到候选长度集合过大会导致分支数目过多（50um~400um 共 351 个分支），我们定义：

$$min\_length = \sqrt{ins\_area} \times 0.5 \qquad max\_length = \sqrt{ins\_area} \times 2.0$$

则候选长度集合为$\{max\{50, min_length\}, \ldots, min\{400, max_length\}\}$，如此便大大减少了分支数目，而且也能优化长宽比（式中ins_area表示算例所有多边形面积和）。

由于我们仅固定了包络矩形的长度，没有限制宽度，有可能导致某些很短的长度求得的包络矩形宽度会超出 300um。当检测到这种解不合法的情况时，我们对该长度施加惩罚，具体做法是将其面积 Area 设置为无穷大，如此意味着该长度潜力很差，则在后续的迭代中该长度被再次选中的概率 Prob 就会很小，从而淘汰潜力差的候选长度。

2.3 随机局部搜索（RLS）

我们使用的随机局部搜索（Random local search）参考了论文《An efficient intelligent search algorithm for the two-dimensional rectangular strip packing problem》[2]。局部搜索算法包括初始解构造阶段和迭代改进阶段。

2.3.1 初始解构造阶段

在初始解构造阶段，我们使用了一种贪心放置策略。针对当前放置状态（packing pattern）找到最下最左（bottom-left）角，尝试将所有还未放置的多边形放置在该角，由此可以产生多种新的 packing pattern，对这多种 packing pattern 进行打分评估，选择得分最高的一个多边形放置。然后找到新形成的 bottom-left 角放置新的多边形，直到所有的多边形都放置完毕即得到一个完整的布局方案。在整个放置过程中，使用 skyline 维护边、角等信息。针对不同的多边形形状，我们设计了相应的打分策略。

矩形的打分策略如图 2.2 所示：

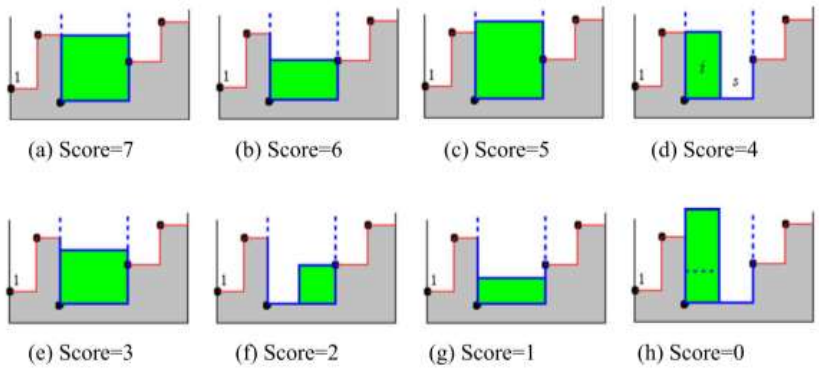

图 2.2 矩形打分策略

L 字形的打分策略如图 2.3 所示：

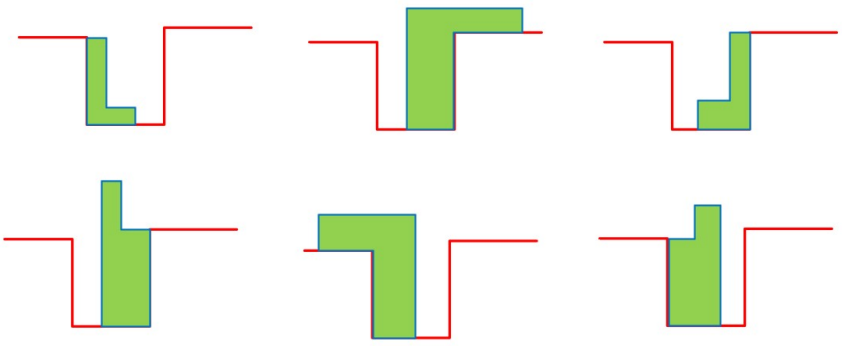

图 2.3 L 字形打分策略（Score=10）

T 字形的打分策略如图 2.4 所示：

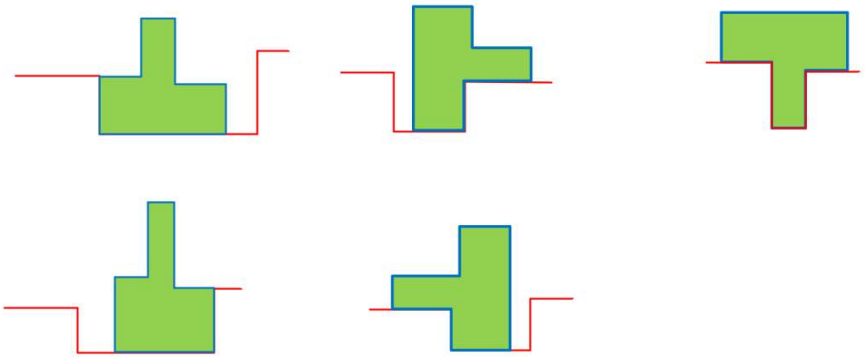

图 2.4 T 字形打分策略（Score=20）

凹字形的打分策略如图 2.5 所示：

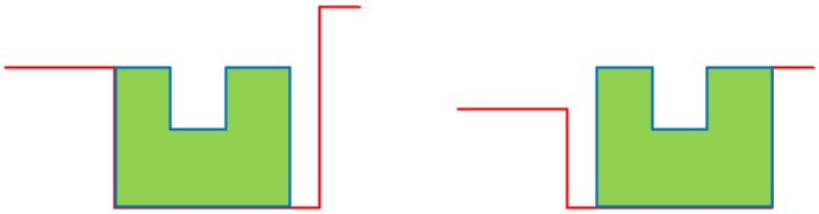

图 2.5 凹字形打分策略（Score=20）

我们的原则是能从 Score 值反映不同形状的放置优先级，即 T 字形/凹字形>L 字形>矩形。矩形放置相对容易，也只有一种旋转方式，因此我们参考论文中细化的矩形打分策略，针对矩形与当前 packing pattern 的不同位置关系设计了 8 个 Score 值奖励；由于 L、T、凹字形的放置难度大于矩形，在放置时我们会评估所有旋转角度（0°/90°/180°/270°，凹字形不考虑旋转），选择能使得当前 packing pattern 的边、角数目减少最多的旋转角度作为最佳放置方案，并给予相应的 Score 值奖励。

2.3.2 迭代改进阶段

迭代改进阶段我们使用了随机局部搜索（RLS），其算法思想与 ASA 类似，ASA 的目的在于找到一个有潜力的包络矩形长度（Length），而 RLS 是为了找到一个有潜力的多边形放置顺序（Rule），具体的算法流程如图 2.6 所示。

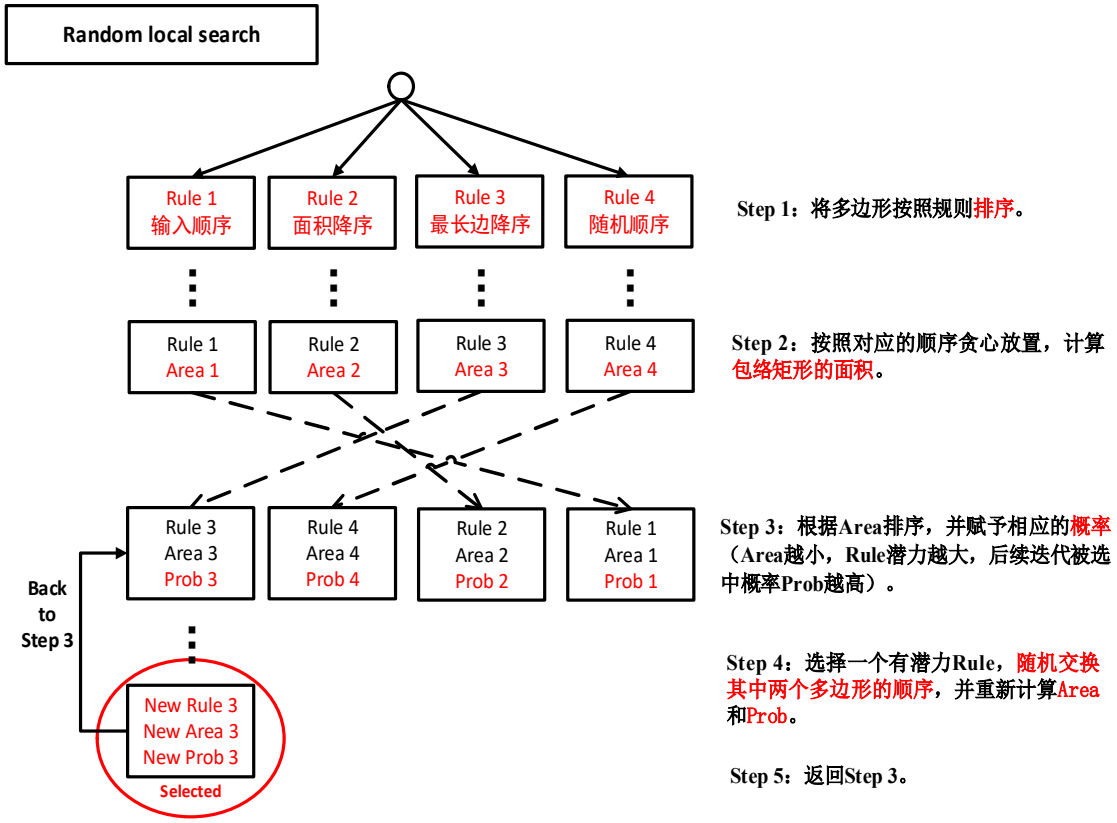

图 2.6 Random local search 算法流程

RLS 涉及随机交换当前序列中两个多边形的顺序（图 2.6 中 Step 4），如果基于新序列的解不比上一个解差，则接受该解并使用新序列替换旧序列。此外，基于概率的自适应选择策略能较好的平衡疏散性和集中性，不停迭代优化放置顺序。

三、主要创新点

3.1 基于 ASA 改造的搜索框架

原论文[1]提出的 ASA 搜索框架采用的生成候选长度的办法是使用排列组合的方法。需要计算所有的矩形长度、宽度的组合，生成的候选长度集合具有一定的局限性（因为这样隐含要求最底一层必须放满），并不适用于放置多边形包含 L、T、凹字形的情况；而且计算排列组合复杂度太高，会大大延长程序耗时。改造后的 ASA 分支策略是等距地生成候选长度，生成完整候选长度集合的时间复杂度仅为 O(n)，且结合 *min_length* 和 *max_length* 能使分支数目远小于 351（[50,400]），具体分支情况统计如表 3.1 所示。

表 3.1 分支情况统计

算例名	min_length	max_length	分支数目
polygon_area_etc_input_1	50	88	39
polygon_area_etc_input_2	50	95	46
polygon_area_etc_input_3	50	142	93
polygon_area_etc_input_4	50	188	139
polygon_area_etc_input_5	65	261	197
polygon_area_etc_input_6	92	333	242
polygon_area_etc_input_7	50	137	88
polygon_area_etc_input_8	50	175	126
polygon_area_etc_input_9	50	199	150
polygon_area_etc_input_10	50	121	72

此外，原论文[1]解决的是 2DSP，即固定长度不限制宽度的矩形 packing 问题。结合赛题要求，我们对于超出宽度约束的放置结果施加惩罚，使得其潜力变差，从而在后续迭代中削弱其被选中的概率。

3.2 针对不规则多边形的打分策略

原论文[2]的放置对象只有矩形，因此他们只设计了矩形打分策略。赛题包含 L、T、凹字形，我们分别针对上述多边形设计了相应的打分策略，并使用不同的 Score 值反映其放置优先级。详见 **2.3.1 初始解构造阶段**。

四、遇到的主要问题及解决方式

表 4.1 遇到的主要问题及解决方式

遇到的问题	解决方式
测试算例文件中有部分共线、重复的坐标点	将算例文件可视化输出， 删除共线、重复的坐标点
polygon_area_etc_input_9 中包含一个凹字形	为凹字形单独设计打分策略
输出结果文件如何进行重叠检测？	将结果文件可视化输出，详见七、附录（二）
算法停止条件如何设置？	使用超时时间和迭代步数双重控制， 防止运算耗时过长
服务器上编译出现动态链接库.so 不匹配的问题，报错如下： /usr/lib64/libstdc++.so.6: version `GLIBCXX_3.4.18` not found	使用静态链接编译选项：-static-libstdc++

五、参考文献

[1] WEI L, ZHU W, LIM A等. An adaptive selection approach for the 2D rectangle packing area minimization problem[J]. Omega (United Kingdom), Elsevier Ltd, 2018, 80: 22–30.

[2] WEI L, QIN H, CHEANG B等. An efficient intelligent search algorithm for the two-dimensional rectangular strip packing problem[J]. International Transactions in Operational Research, 2016, 23(1–2): 65–92.

[3] CHEN J, LIU Y, ZHU Z等. An adaptive hybrid memetic algorithm for thermal-aware non-slicing VLSI floorplanning[J]. Integration, the VLSI Journal, Elsevier B.V., 2017, 58(March): 245–252.

[4] STOCKMEYER L. Optimal orientations of cells in slicing floorplan designs[J]. Information and Control, 1983, 57(2–3): 91–101.

[5] MIRHOSEINI A, GOLDIE A, YAZGAN M等. Chip Placement with Deep Reinforcement Learning[J]. 2020.

[6] GUO P N, CHENG C K, YOSHIMURA T. O-tree representation of non-slicing floorplan and its applications[J]. Proceedings - Design Automation Conference, 1999: 268–273.

[7] CHANG Y, CHANG Y, WU G等. B * -Trees : A New Representation for Non-Slicing Floorplans[J]. 2000(c): 458–463.

[8] HE K, JI P, LI C. Dynamic reduction heuristics for the rectangle packing area minimization problem[J]. European Journal of Operational Research, Elsevier Ltd., 2015, 241(3): 674–685.

六、附录（一）设计文档

6.1 软件架构

本算法是一个清晰的二层搜索框架，上层 ASA 的目标在于找到一个有潜力的包络矩形长度，下层的 RLS 则是为了找到一个合理的放置顺序。为了平衡算法的疏散性和集中性，我们在两层中都引入了基于概率的自适应选择策略，使潜力大的候选成员能得到较好的发展，又不至于完全抛弃潜力差的成员。

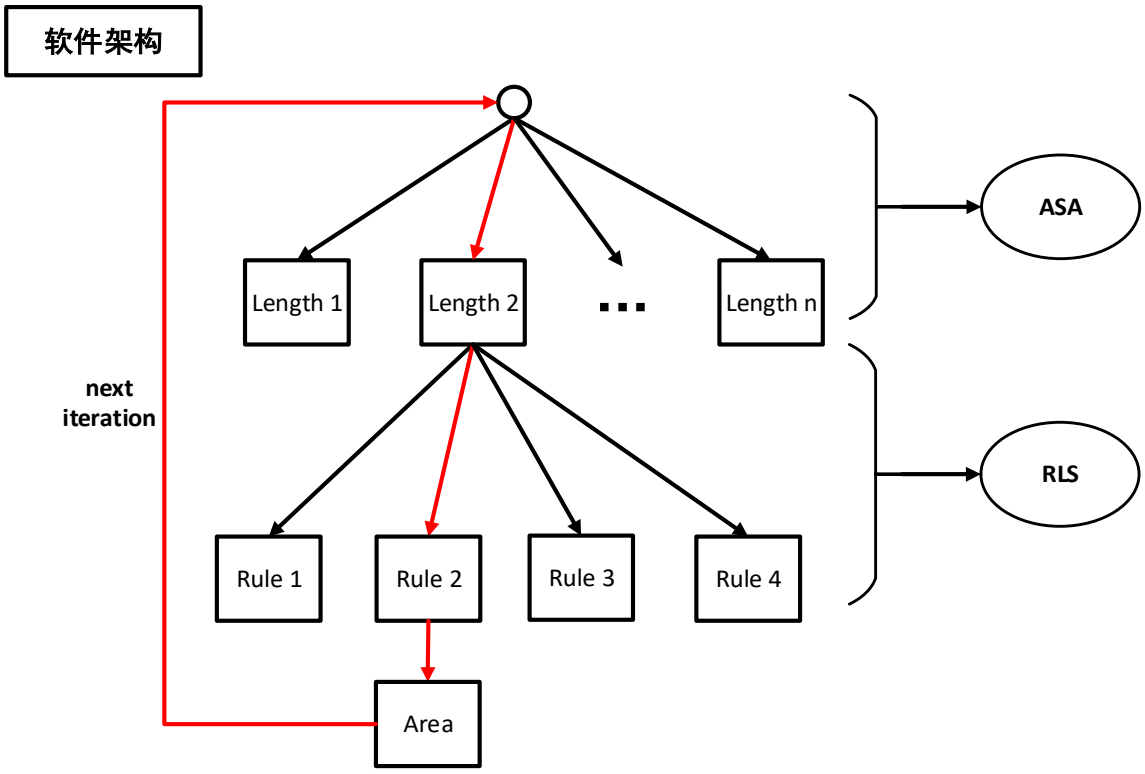

图 6.1 软件架构图

6.2 主要模块说明

6.2.1 ASA 模块

ASA 模块图示详见 **2.2 自适应选择搜索框架（ASA）**。

下面提供 ASA 的伪代码：

Algorithm 1 Adaptive selection approach for the RPAMP.

AdaptiveSelection(R)

Calculate the set of candidate widths W
$totalArea$ = total area of rectangles in R
for each width $W_i \in W$
    H_i = RandomLS $(R, W_i, 1)$
    $fr_i = \frac{totalArea}{W_i \times H_i}$
    $iter_i = 1$
Sort each width in W by the increasing filling ratio
while time limit is not exceeded
    Select the kth width from W randomly with the probability $\frac{2k}{|W| \times (|W|+1)}$
    $iter_k = \min(2 \times iter_k, n)$
    H = RandomLS $(R, W_k, iter_k)$
    Update H_k and fr_k if H is smaller than H_k
    Update W to make sure it is sorted by the increasing filling ratio
return best found solution

6.2.2 RLS 模块

RLS 模块图示详见 2.3.2 迭代改进阶段。

下面提供 RLS 的伪代码：

Algorithm 2 RandomLS Procedure.

RandomLS($R, W_k, iter$)

Let sr_k be the set of sorting rules on W_k
if it is the first time to call RandomLS on width W_k
    for each sorting rule
        Sort R using the sorting rule
        h = HeuristicPacking(R, W_k)
        Add this sorting rule into sr_k
    Sort the sorting rules in sr_k in decreasing order of the height returned by HeuristicPacking
Select the ith sorting rule from sr_k randomly with probability $\frac{2i}{|sr_k| \times (|sr_k|+1)}$
Sort R using the selected sorting rule
h = HeuristicPacking(R, W_k)
for $i = 1$ to $iter$
    Generate sequence R' from R by randomly swapping two rectangles
    h' = HeuristicPacking(R', W_k)
    if $h' \leq h$
        $h = h'$; $R = R'$
Update sr_k to make sure it is sorted in decreasing order of the height
return minimum found height

七、附录（二）结果可视化

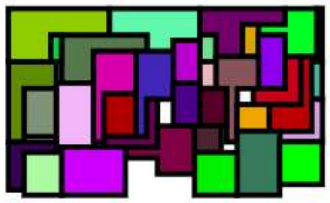

图 7.1 polygon_area_etc_input_1 结果

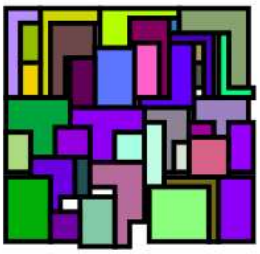

图 7.2 polygon_area_etc_input_2 结果

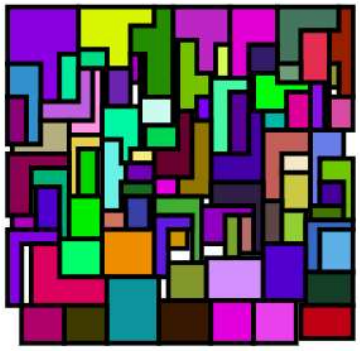

图 7.3 polygon_area_etc_input_3 结果

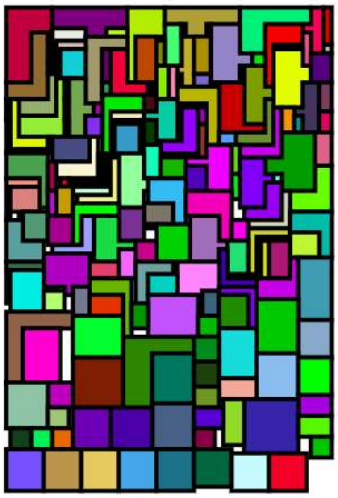

图 7.4 polygon_area_etc_input_4 结果

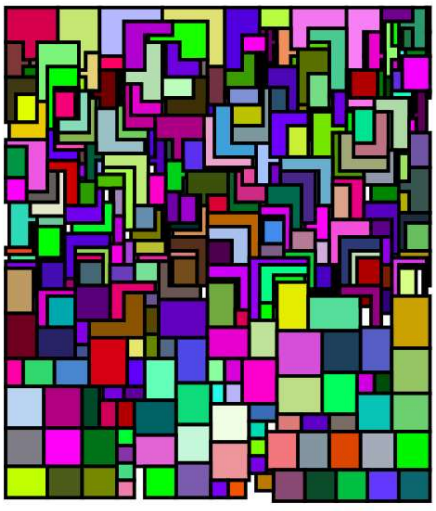

图 7.5 polygon_area_etc_input_5 结果

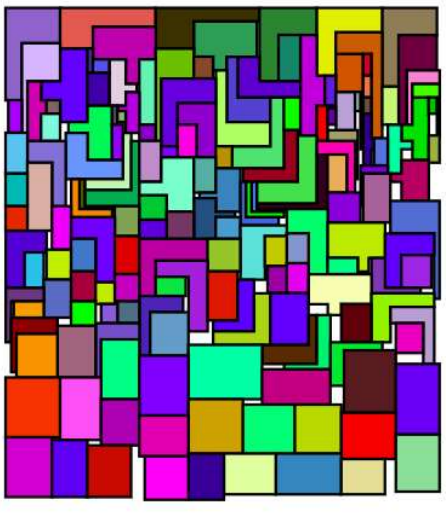

图 7.6 polygon_area_etc_input_6 结果

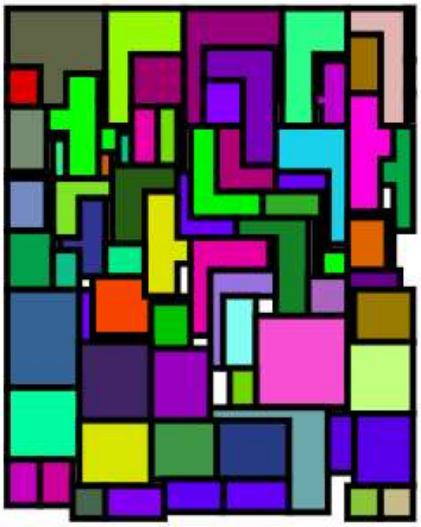

图 7.7 polygon_area_etc_input_7 结果

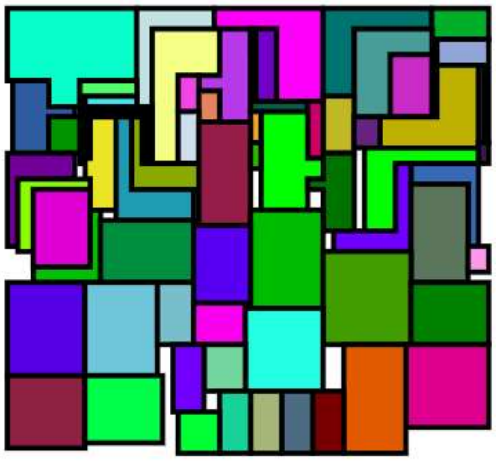

图 7.8 polygon_area_etc_input_8 结果

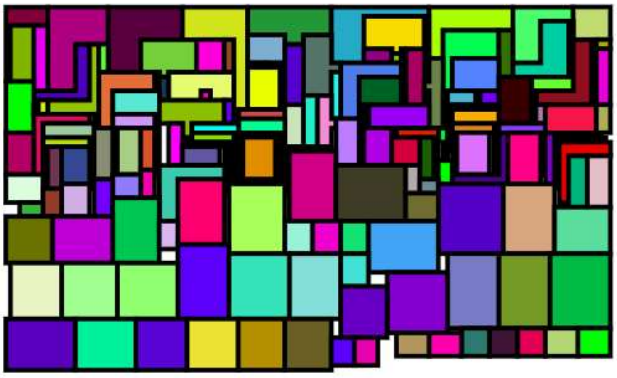

图 7.9 polygon_area_etc_input_9 结果

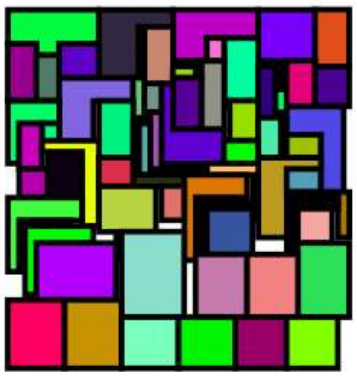

图 7.10 polygon_area_etc_input_10 结果