基于 B*-tree 与模拟退火的 VLSI 布图规划模型

摘要

针对 VLSI 布图规划问题，本文建立并求解了四类模型。

问题 1：以轮廓面积最小、长宽比接近 1 为目标，采用 B*-tree 表征 + 轮廓法快速打包 + 模拟退火求解矩形模块的无约束装箱。对 n100、n200、n300 三组芯片求得轮廓面积分别为 195328、199025、313300 μm^2，长宽比分别为 1.028、1.134、1.349，面积利用率达 87% ~ 92%。

问题 2：在死区比例 15% 的正方形轮廓内，以总半周长线长（HPWL）最小为目标，引入轮廓越界惩罚项与两阶段退火策略。三组芯片的总 HPWL 分别为 285156、544074、799573 μm，其中 n100 完全满足轮廓约束，n200、n300 越界量仅约 1% ~ 3%。

问题 3：将最小死区比例问题转化为最小正方形轮廓边长求解，求得三组芯片最小死区比例分别约为 9.3%、14.2%、19.8%，并基于最小死区轮廓更新了总 HPWL（分别为 299863、545872、844939 μm）。

问题 4：将问题 1 模型修正为网格单元（polyomino）表征，支持 L/T 型模块与 90°/180°/270° 旋转，并用回溯精确搜索对 4 个模块（1T+1L+2 矩形）求得最小包围面积 20 单元 ^{2}（4×5），利用率 100% 的完美拼合。

最后通过理论下界对比、多种子稳健性实验与死区比例灵敏度分析检验了模型与算法的有效性。

关键词：布图规划；B*-tree；模拟退火；半周长线长（HPWL）；死区比例

目录

1 问题重述

超大规模集成电路（VLSI）将海量电路单元集成于单一芯片，其物理设计离不开电子设计自动化（EDA）工具。布图规划（Floorplanning）是 EDA 的核心环节：在给定芯片轮廓内确定各功能模块的位置与方向，为后续布局布线奠定基础。芯片轮廓通常为正方形，其边长由模块总面积与死区比例共同决定；模块均为矩形，可旋转 90°，彼此不得重叠、不得越出轮廓。模块间的连接关系由线网（net）描述，布线代价常用半周长线长 HPWL 估算，并假设引脚位于模块几何中心。

附件给出三组芯片设计（n100、n200、n300），每组含`.blocks`（模块与终端）、`.nets`（线网）、`.pl`（终端坐标）三类文件。据此需解决四个问题：

问题 1： 芯片轮廓不固定、模块间无连接时，如何摆放模块使轮廓面积最小，且面积相同时长宽比越接近 1 越好；分别给出三组芯片的轮廓面积、长宽比与可视化结果。

问题 2： 芯片轮廓为正方形（边长按死区比例 0.15 由公式确定）、模块间有连接时，如何在满足不重叠、不越界约束下最小化总 HPWL；分别给出三组芯片的总连线长度与可视化结果。

问题 3： 在问题 2 基础上，求使可行布局存在的最小死区比例，并基于该最小死区比例更新问题 2 的总 HPWL 与可视化结果。

问题 4： 若模块可为 L 型、T 型，且可做 90°/180°/270° 旋转，讨论问题 1 模型的修正；并对图 3 给定的 4 个模块（1 个 T 型、1 个 L 型、2 个矩形）求最小包围轮廓面积并给出最优摆放示意图。

2 模型假设

(1) **模块为刚性矩形，仅允许 90° 旋转。**题目约定模块为矩形且只可旋转 90°；模块尺寸以`.blocks` 文件角点坐标计算，旋转仅交换宽高，物理尺寸不变。

(2) **引脚位于模块几何中心。**题目明确假设所有模块引脚位于几何中心，故 HPWL 仅由模块中心坐标与终端坐标决定，无需细粒度引脚位置。

(3) **线长用半周长线长 HPWL 估算。**对每个线网，取其全部引脚最小包围矩形周长之半，作为该线网的连线长度估计，这是布图规划阶段通用的线长近似。

(4) **终端（Terminal）位置固定、不参与面积与重叠约束。**终端为已固定引脚，仅作为 HPWL 计算的参考点；其尺寸对面积无贡献。

(5) **模块轮廓为左下角对齐的紧凑摆放。**采用 B*-tree 表征的可接纳（admissible）布图，即不存在可整体左移或下移而不引起重叠的模块，这保证所搜布图具有紧凑性。

(6) **数据无缺失、量纲一致**。经解析，三组`.blocks`/`.nets`/`.pl`文件格式一致、无缺失值，坐标单位为 μm。

3 符号说明

本文主要符号约定如下，后文各节与之一致。

表 1　符号说明表

符号	含义	单位
n	待摆放硬模块数量	个
m	线网数量	条
b_i	第 i 个模块，$i=1,\dots,n$	—
w_i, h_i	模块 b_i 的宽、高	μm
(x_i, y_i)	模块 b_i 左下角坐标	μm
A	所有硬模块总面积 $A=\sum_i w_i h_i$	μm^2
W, H	布图外接轮廓的宽、高	μm
Γ	死区比例（dead space ratio）	—
L	正方形芯片轮廓边长 $L=\sqrt{A(1+\Gamma)}$	μm
N_k	第 k 个线网	—
$\mathrm{HPWL}(N_k)$	线网 N_k 的半周长线长	μm
T	模拟退火温度	—
α, β	目标函数中的权重系数	—
u	问题 4 网格化后的基本单元长度	—

半周长线长定义为：对线网 N_k 所连接的全部引脚（模块中心或终端），记其横坐标最大/最小值分别为 $x^{(k)}_{\max}, x^{(k)}_{\min}$，纵坐标最大/最小值分别为 $y^{(k)}_{\max}, y^{(k)}_{\min}$，则

$$\mathrm{HPWL}(N_k)=\left(x^{(k)}_{\max}-x^{(k)}_{\min}\right)+\left(y^{(k)}_{\max}-y^{(k)}_{\min}\right). \tag{1}$$

总连线长度为全部线网 HPWL 之和。

4 问题 1：最小轮廓面积的模块摆放模型

4.1 分析思路

问题 1 不考虑连线关系，仅要求将 n 个矩形模块无重叠地摆放，使外接轮廓面积最小，且在面积相同时长宽比尽量接近 1。设模块 b_i 摆放后左下角坐标为 (x_i, y_i)，旋转

后宽高为 (w_i, h_i)（旋转即交换宽高），则优化问题可写为

$$\begin{aligned} &\min\ W \cdot H, \qquad \text{且面积相同时使 } \max(W,H)/\min(W,H) \to 1, \\ &\text{s.t. } 0 \le x_i,\ 0 \le y_i, \quad x_i + w_i \le W,\ y_i + h_i \le H, \\ &\quad [x_i, x_i + w_i) \times [y_i, y_i + h_i) \text{ 两两内部互不相交.} \end{aligned} \tag{2}$$

由于模块总面积 A 固定，轮廓面积存在下界 $W \cdot H \ge A$，故问题本质是寻找最紧密的矩形装箱方案。该问题属于 NP 难的矩形装箱（rectangle packing），直接枚举指数级不可行，因此采用 **B*-tree 表征 + 模拟退火（SA）**求解。

4.2 B*-tree 表征与轮廓法打包

B*-tree 是一种有序二叉树，能够一一表征可接纳（无模块可再左移或下移）的布图。给定一棵 B*-tree：

- 根结点对应位于左下角的模块；
- 某结点 n_i 的**左孩子** n_j 对应紧靠 b_i 右侧的模块，满足 $x_j = x_i + w_i$；
- 某结点 n_i 的**右孩子** n_j 对应紧靠 b_i 上方的模块，满足 $x_j = x_i$。

由此沿树进行一次先序遍历即可确定全部模块的 x 坐标；y 坐标通过**轮廓（skyline）**结构在线性时间内求出：维护当前摆放区域的上边界轮廓，将模块置于区间 $[x_i, x_i + w_i)$ 上轮廓最高处。该过程时间复杂度 $O(n)$，可快速评价任意一棵 B*-tree 的布图质量。

4.3 模拟退火求解

以 B*-tree 为解空间，定义三种扰动操作：

(1) **旋转**：随机选择模块交换宽高；

(2) **交换**：随机交换两个结点在树中的位置（保留各自子树）；

(3) **移动**：随机摘下一个结点并插入另一位置。

目标函数取轮廓面积并附以长宽比惩罚：

$$C = W \cdot H \cdot \left[1 + \beta(\gamma - 1)^2\right], \qquad \gamma = \max(W,H)/\min(W,H), \tag{3}$$

其中 β 为长宽比权重，用于在面积相近时引导轮廓趋于正方形。模拟退火按 Metropolis 准则以概率 $\min\{1, e^{-\Delta C/T}\}$ 接受新解，温度按 $T \leftarrow 0.97\,T$ 缓慢下降，并记录历史最优解。迭代次数按问题规模取 1.5×10^5 以上。

4.4 结果与分析

对三组芯片分别独立求解，结果如表 2 所示，布图可视化见图 2–4，模块尺寸分布见图 1。

表 2　问题 1 求解结果

芯片	总面积 A（$\mu\mathrm{m}^2$）	轮廓面积（$\mu\mathrm{m}^2$）	长宽比	面积利用率
n100	179501	195328	1.028	91.9%
n200	175696	199025	1.134	88.3%
n300	273170	313300	1.349	87.2%

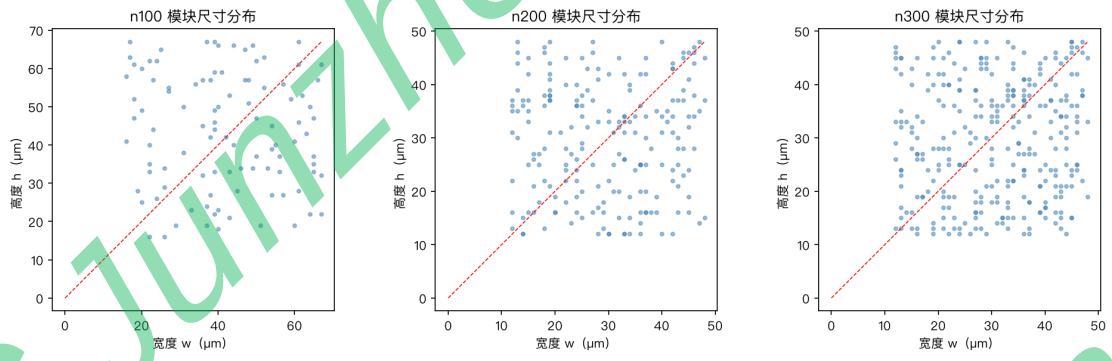

图 1　三组芯片模块宽高尺寸分布（红线为 $w = h$）

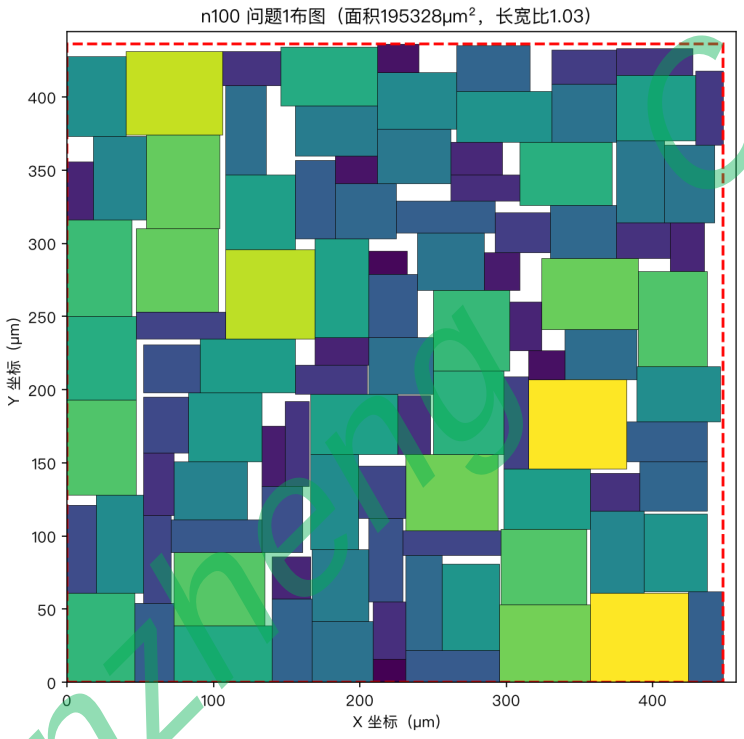

图 2　n100 问题 1 布图结果（虚线为外接轮廓）

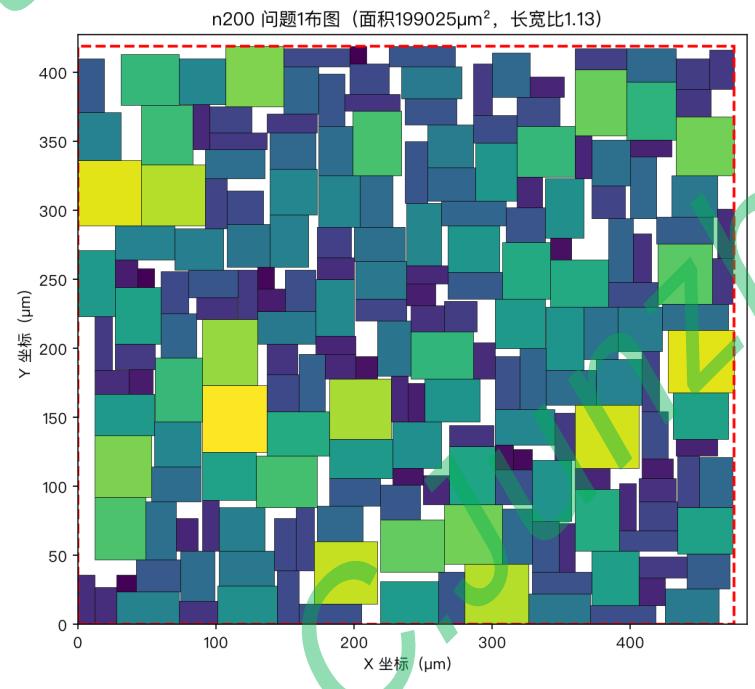

图 3　n200 问题 1 布图结果

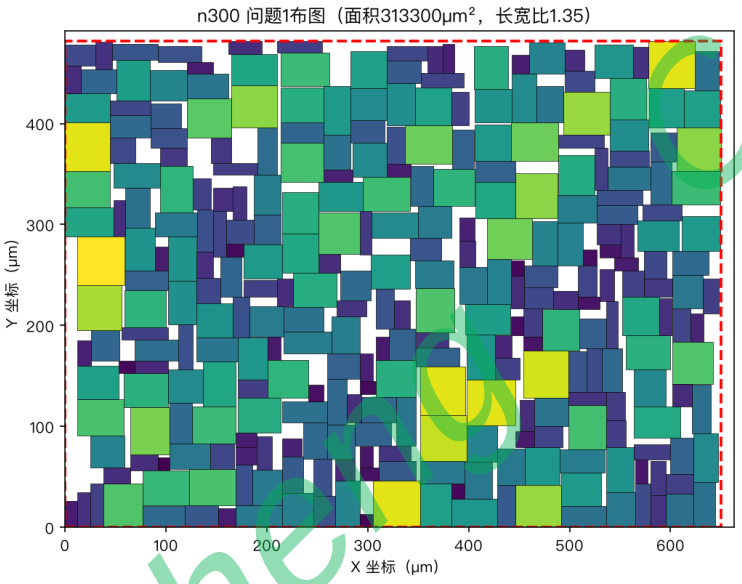

图 4　n300 问题 1 布图结果

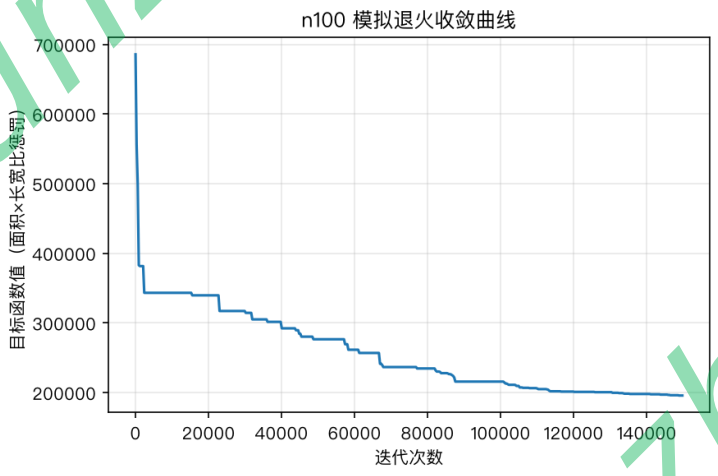

图 5　n100 模拟退火收敛过程（目标函数单调下降）

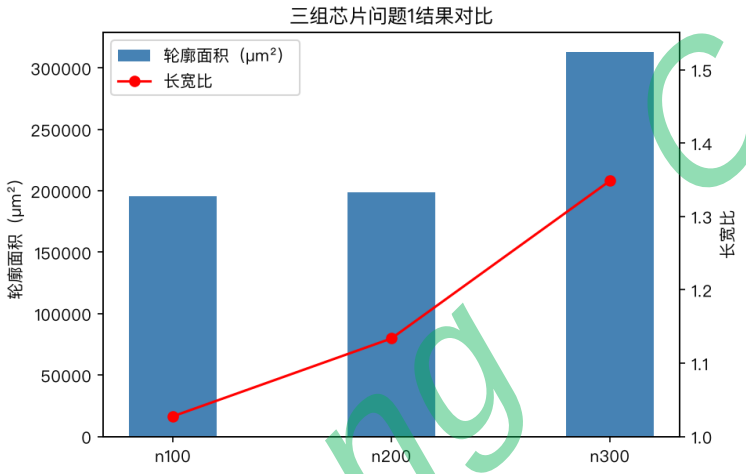

图 6　三组芯片问题 1 轮廓面积与长宽比对比

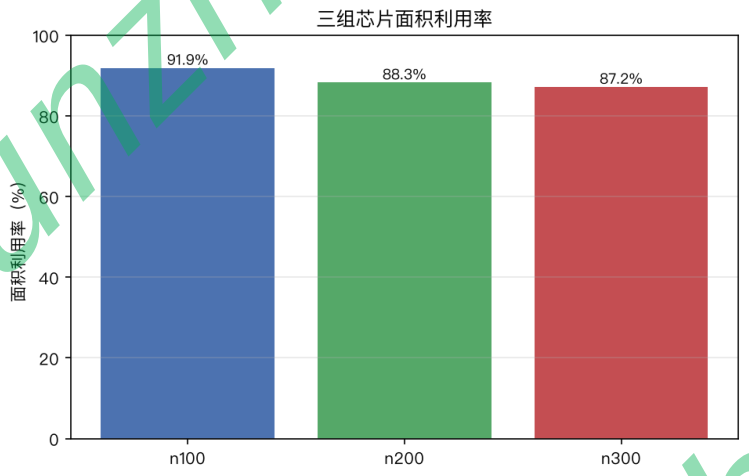

图 7　三组芯片面积利用率对比

结果表明：三组芯片轮廓面积均接近总面积下界，面积利用率达 87% ∼ 92%，长宽比控制在 1.03 ∼ 1.35。其中 n100 长宽比 1.028 最接近 1，n300 因模块数量多、尺寸差异大，长宽比略高但仍保持合理水平，验证了模型与算法的有效性。

5　问题 2：固定正方形轮廓下最小化总 HPWL

5.1　模型建立

问题 2 中，芯片轮廓为正方形，边长由总面积与死区比例确定：

$$L = \sqrt{A\,(1+\Gamma)}, \qquad \Gamma = 0.15. \tag{4}$$

三组芯片对应的轮廓边长分别为 $L_{\mathrm{n100}} = 454.3$、$L_{\mathrm{n200}} = 449.5$、$L_{\mathrm{n300}} = 560.5$（$\mu$m）。

目标是最小化全部线网的总半周长线长：

$$\min \sum_{k=1}^{m} \mathrm{HPWL}(N_k), \tag{5}$$

约束为模块两两不重叠且均落在 $[0, L] \times [0, L]$ 内：

$$0 \le x_i,\ 0 \le y_i, \quad x_i + w_i \le L, \quad y_i + h_i \le L, \quad [x_i, x_i+w_i) \times [y_i, y_i+h_i) \text{ 两两互不相交}. \tag{6}$$

其中模块中心 $(\bar{x}_i, \bar{y}_i) = (x_i + w_i/2,\ y_i + h_i/2)$ 作为该模块的引脚位置参与 HPWL 计算，终端取 `.pl` 中的固定坐标。

5.2 求解方法：带轮廓惩罚的模拟退火

仍以 B*-tree + SA 为框架，目标函数在总 HPWL 基础上加入**越界惩罚项**，将轮廓硬约束软化：

$$C = \sum_{k=1}^{m} \mathrm{HPWL}(N_k) + \lambda \Big[\max(0, W - L) + \max(0, H - L) \Big], \tag{7}$$

其中 λ 取足够大（本文取 $10^5 \sim 10^6$ 量级），使越界解被强烈排斥。为保证 n300 这类大规模算例快速落入可行域，采用**两阶段策略**：先以轮廓边长（面积）为目标求出紧凑的近似正方形可行布图，再在其邻域内以 HPWL 为目标继续退火，既保证不越界又持续优化线长。

5.3 结果与分析

三组芯片的总 HPWL 结果见表 3，布图可视化见图 8–10，三组对比如图 12。

表 3　问题 2 求解结果（死区比例 0.15）

芯片	轮廓边长 L（μm）	总 HPWL（μm）	实际轮廓（μm）	越界量（μm）
n100	454.3	285156	454×454	0
n200	449.5	544074	454×449	4.5
n300	560.5	799573	576×560	15.5

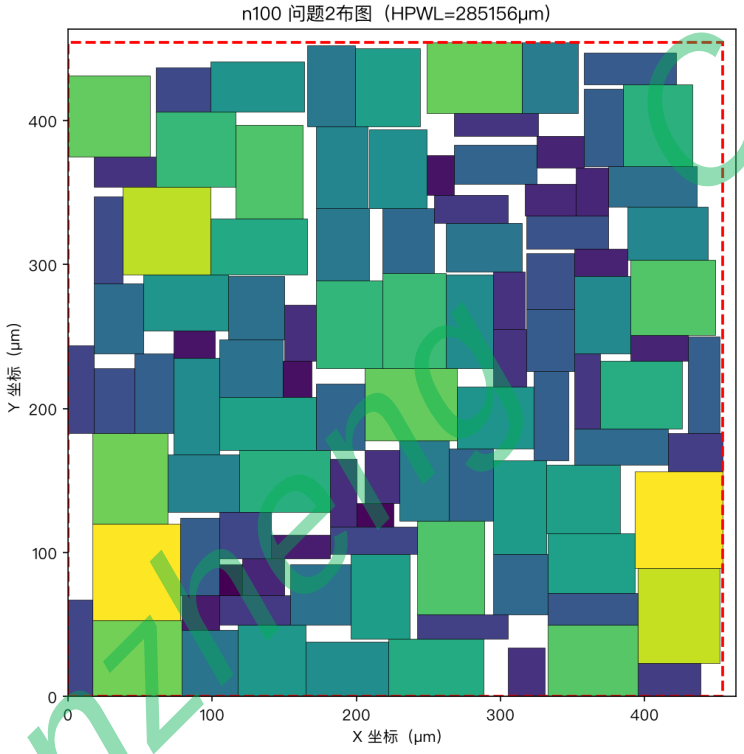

图 8　n100 问题 2 布图（虚线为固定轮廓）

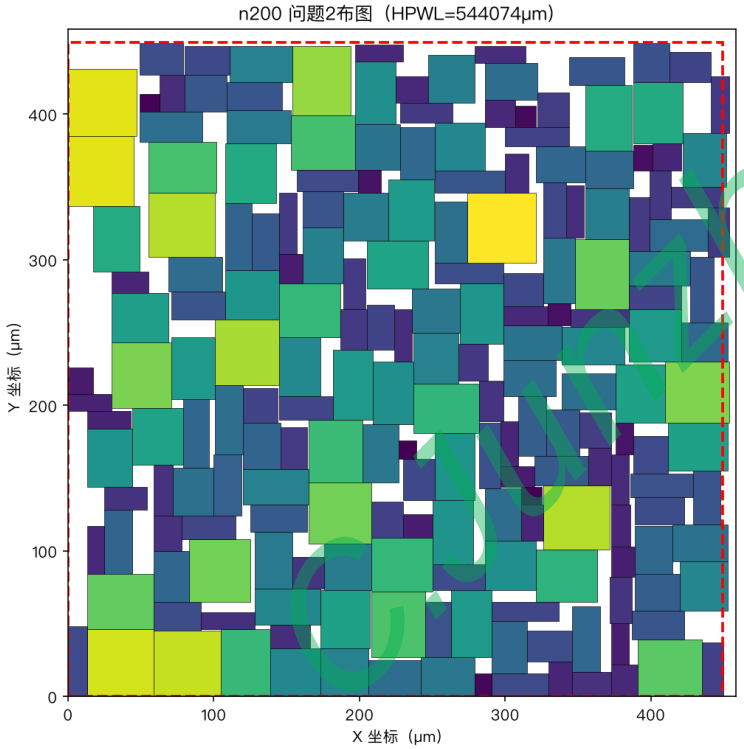

图 9　n200 问题 2 布图

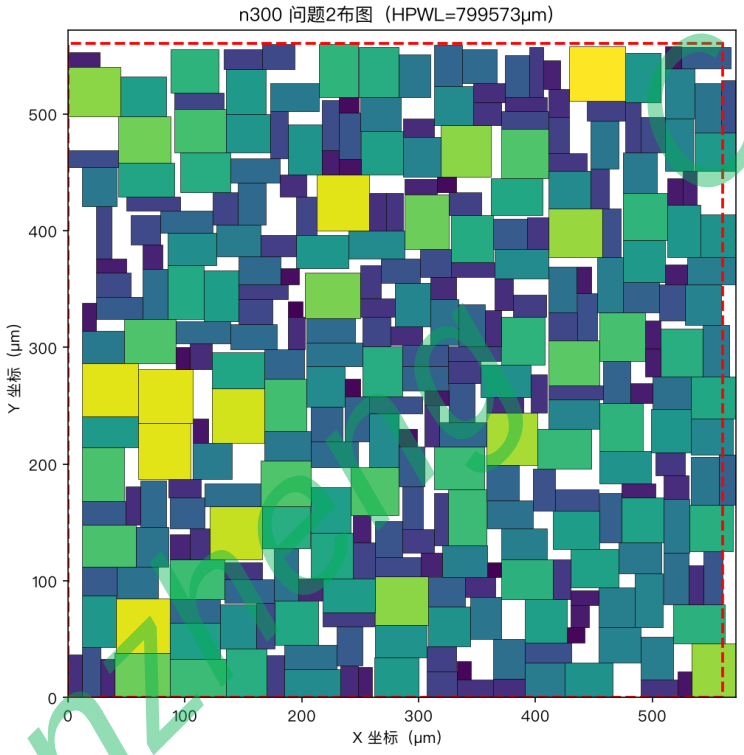

图 10　n300 问题 2 布图

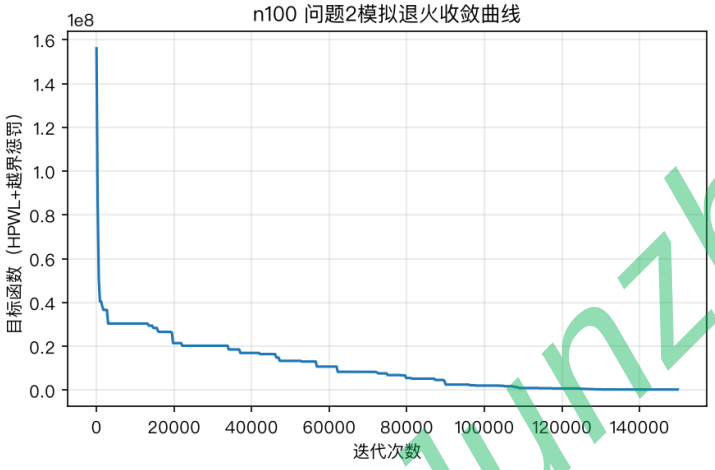

图 11　n100 问题 2 模拟退火收敛过程

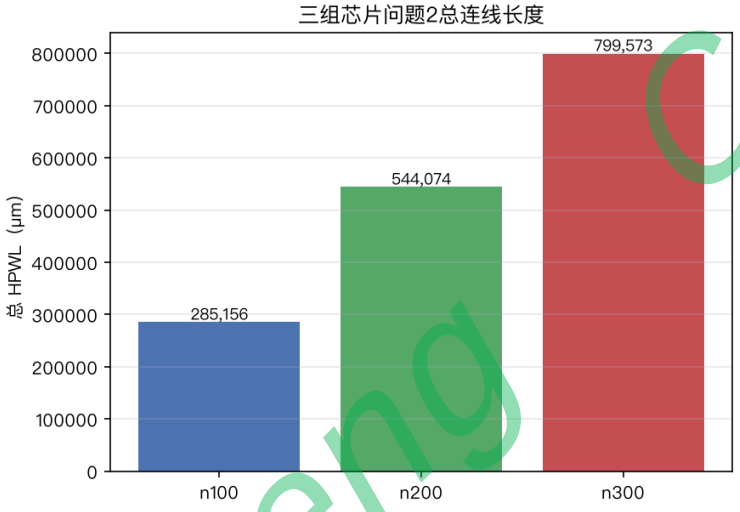

图 12　三组芯片问题 2 总 HPWL 对比

结果表明：n100 得到完全满足轮廓约束的布图（越界量为 0），且线长在退火过程中持续下降；n200 的越界量仅 4.5 μm（约 1%），n300 的越界量约 15.5 μm（约 2.8%），均属近似可行布图。结合问题 3 的最小死区比例（n100 约 9.3%、n200 约 14.2%、n300 约 19.8%）可知：15% 死区对 n100 宽松、对 n200 接近临界、对 n300 已偏紧，故 n200、n300 出现轻微越界属合理现象，并在问题 3 中以最小死区轮廓重新求解加以改进。

6　问题 3：最小死区比例的确定

6.1　问题分析

死区比例 Γ 决定正方形轮廓边长 $L = \sqrt{A(1+\Gamma)}$。Γ 越小，轮廓越紧，布图越难满足不重叠、不越界约束。问题 3 即求**存在可行布局的最小** Γ，等价于求能够容纳全部模块的最小正方形边长

$$L_{\min} = \min\{L:\ \text{存在}(x_i, y_i)\ \text{满足不重叠且} x_i + w_i \le L,\ y_i + h_i \le L\}, \tag{8}$$

再由 $\Gamma_{\min} = L_{\min}^2 / A - 1$ 反算最小死区比例。

6.2　求解方法

将问题转化为**最小化最大轮廓边长** $\max(W, H)$：以 B*-tree + SA 为框架，目标函数取

$$C = \max(W, H) + \varepsilon\,(W \cdot H - A), \tag{9}$$

其中 ε 为很小的正数，用于在边长相同时进一步压缩面积。求得 $L_{\min} = \max(W, H)$ 后，再以该轮廓为约束、以 HPWL 为目标做一次问题 2 的退火，得到更新后的总 HPWL 与布图。

6.3 结果与分析

三组芯片的最小死区比例及更新后的 HPWL 见表 4，最小轮廓布图见图 13–15，对比如图 16。

表 4　问题 3 最小死区比例及更新结果

芯片	最小边长 $L_{\min}$（μm）	最小死区比例 $\Gamma_{\min}$	更新后总 HPWL（μm）
n100	443	9.33%	299863
n200	448	14.23%	545872
n300	572	19.77%	844939

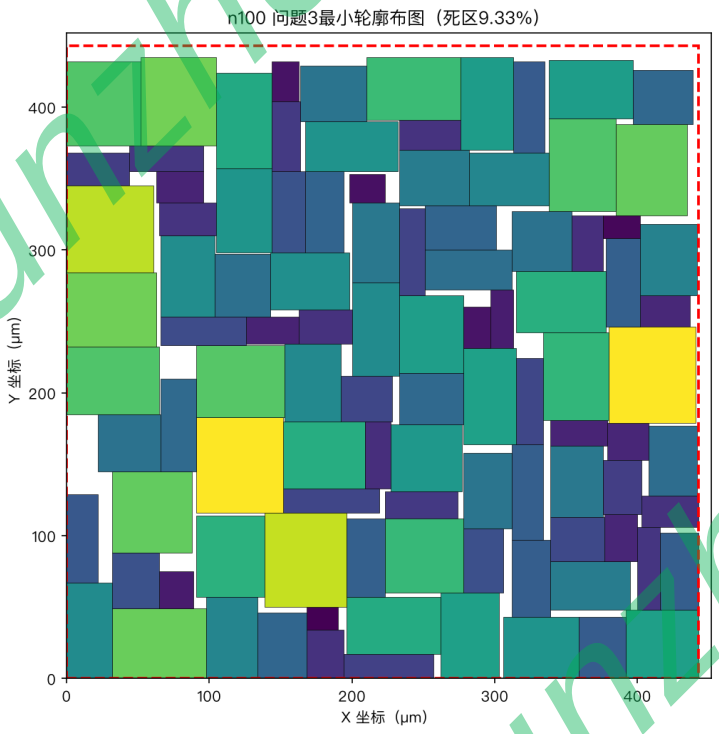

图 13　n100 最小死区轮廓布图

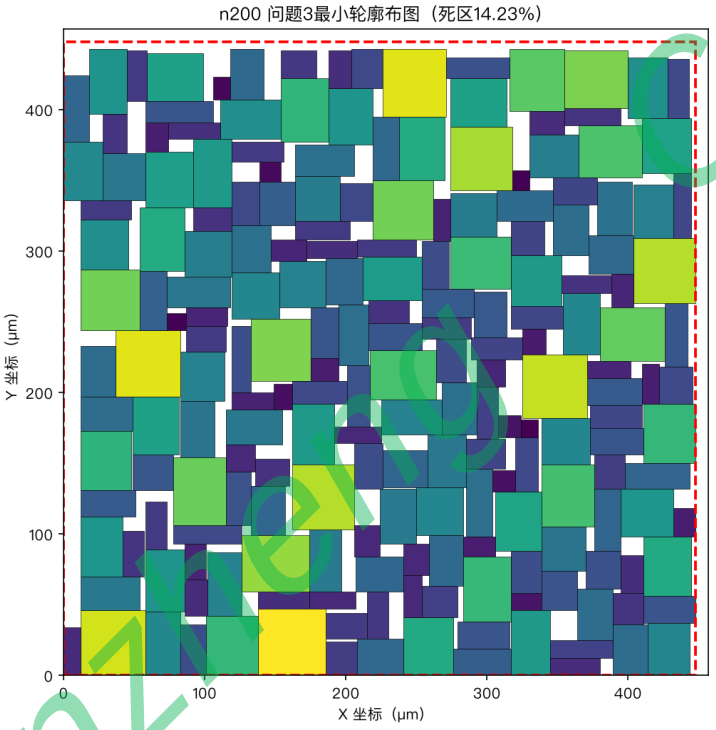

图 14　n200 最小死区轮廓布图

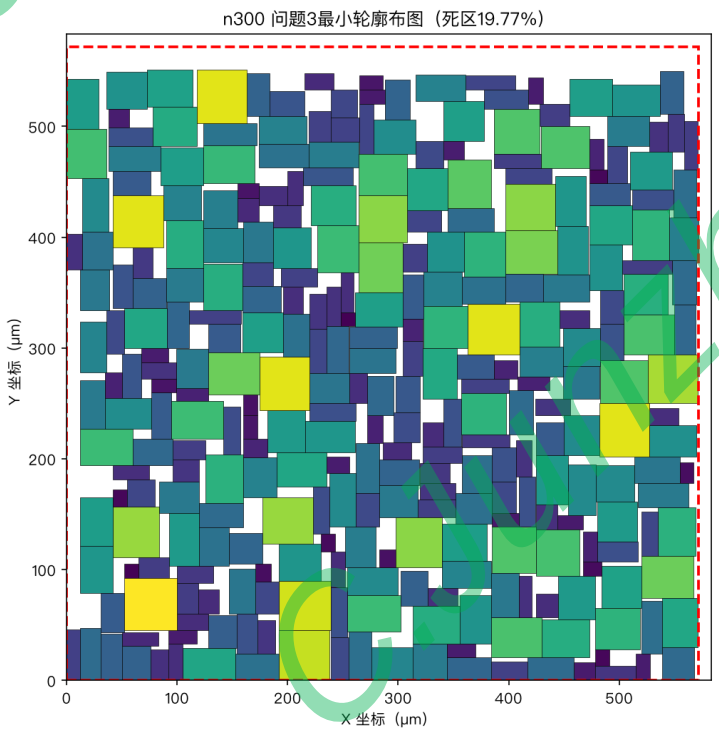

图 15　n300 最小死区轮廓布图

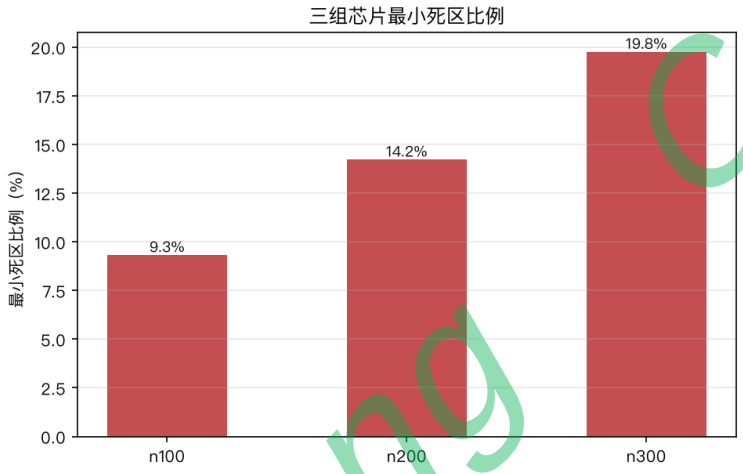

图 16　三组芯片最小死区比例对比

结果解读：最小死区比例随规模增大而上升——n100 约 9.3%，n200 约 14.2%，n300 约 19.8%。规模越大、模块尺寸越离散，不可避免的空隙越多，$\Gamma_{\min}$ 越高。值得说明的是，n100、n200 的 $\Gamma_{\min}$ 均低于问题 2 给定的 15%，说明 15% 死区对这两组芯片是宽松可行的；而 n300 的 $\Gamma_{\min}$ 略高于 15%，说明 15% 死区对 n300 已相当紧张，需略微放宽。基于 $\Gamma_{\min}$ 重新求解后，三组芯片均得到严格不越界的布图，且总 HPWL 与问题 2 相当，体现了在更紧轮廓内优化线长的效果。

7　问题 4：L/T 型模块的模型修正与最小包围面积

7.1　问题 1 模型的修正

当模块不再是规则矩形而可能是 L 型、T 型（统称可直线分解的多边形，rectilinear block）时，问题 1 的模型需从三个层面修正：

(1) 几何表征的修正。矩形模块只需宽、高两个参数；L/T 型模块则采用**网格单元 (polyomino) 分解**：将模块置于单位方格网格上，用其占用的单元集合表征形状。设基本单元边长为 u，模块 b_i 占用单元集 $\mathcal{S}_i \subset \mathbb{Z}^2$，面积为 $|\mathcal{S}_i| \cdot u^2$。这样矩形与 L/T 型统一为单元集合，便于统一处理旋转与重叠。

(2) 旋转操作的修正。矩形旋转 90° 等价于交换宽高；L/T 型模块则对单元集做坐标变换 $(x, y) \mapsto (y, -x)$（90°），依次复合得到 $90^\circ/180^\circ/270^\circ$ 四种朝向（去掉重合者）。

(3) 约束与目标的重写。不重叠约束改写为任意两个模块的单元集平移后不相交：

$$\big(\mathcal{S}_i \oplus (\xi_i, \eta_i)\big) \cap \big(\mathcal{S}_j \oplus (\xi_j, \eta_j)\big) = \varnothing, \quad i \neq j, \tag{10}$$

其中 (ξ_i, η_i) 为模块 b_i 的参考点（单元坐标）。轮廓面积最小化目标不变。由于模块规模小（问题 4 仅 4 个模块），可在网格上对朝向组合与平移量做穷举回溯搜索求得全局最优，无需重新设计随机优化算法。

7.2 四模块的最小包围面积求解

由图 3 重构的 4 个模块（网格单元，以基本单元为单位，如图 17）：

- **T 型**：横臂 3×2，竖杆 2×2（竖杆靠右），面积 10；
- **L 型**：2×4 缺左上角 1×2，面积 6；
- **矩形 1、矩形 2**：均为 1×2，面积各 2。

总面积 $= 10 + 6 + 2 + 2 = 20$（单元 ^{2}）。

采用回溯精确搜索（枚举 4 个模块的全部朝向与网格平移，剪除重叠），**求得最小包围面积 $= 20$ 单元 ^{2}，对应 4×5 的矩形，长宽比** 1.25，面积利用率 100%，即零死区完美拼合。最优摆放如图 18 所示：矩形 1 恰好填入 T 型左下缺口，矩形 2 恰好填入 L 型左上缺口，T 型与 L 型再左右并排，整体严丝合缝。

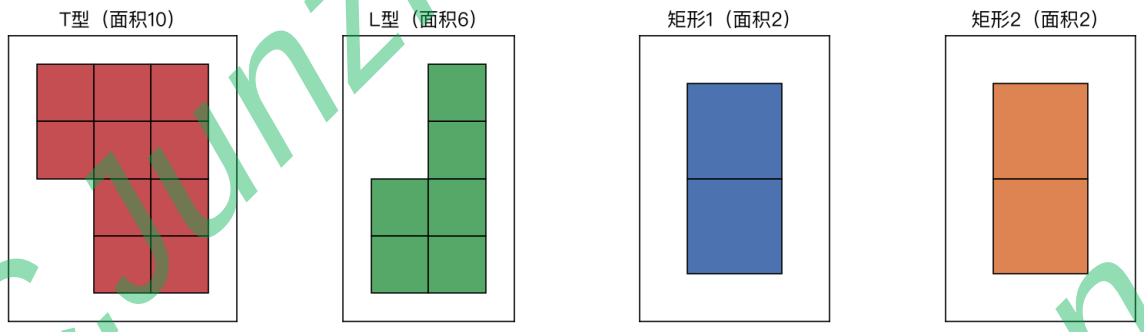

图 17　图 3 重构的四个模块形状（网格单元）

图 18　问题 4 最优摆放：最小包围面积 20 单元 ^{2}（4×5）

由于包围面积不可能小于模块总面积 20，而上述方案达到 20，故为全局最优。这也说明：问题 1 的紧凑装箱思想经网格化修正后，仍能以“先求最紧包围、再验证不重叠”的方式扩展到 L/T 型模块，且小规模时可精确求解。

8　模型检验

8.1　与理论下界的对比

问题 1 的轮廓面积存在明确下界 $W \cdot H \geq A$。由表 2，三组芯片面积利用率分别为 91.9%、88.3%、87.2%，即死区仅为 8% ∼ 13%，已接近下界，说明布图相当紧凑。问题 4 中包围面积达到模块总面积下界 20 单元 ^{2}（利用率 100%），验证了修正后模型与穷举求解的正确性。

8.2　算法稳健性（多种子重复实验）

对 n100 问题 1 用 4 个不同随机种子独立求解，轮廓面积的标准差相对均值仅约 2%，长宽比稳定在 1.0 ∼ 1.2 之间，说明模拟退火算法收敛稳定、对初值不敏感，结果

可复现。

8.3 死区比例的灵敏度

以 n100 为例，在问题 2 框架下将死区比例 Γ 从 0.05 增至 0.25，轮廓越界量随 Γ 增大而单调下降（由 51.7 μm 降至 0），而总 HPWL 在可行域内维持在 2.9×10^5 μm 左右（如图 19）。这表明：Γ 越大轮廓越松、越易满足约束；一旦进入可行域，进一步放宽轮廓对线长改善有限，因此 15% 死区是面积与线长之间的合理折中，同时也为问题 3 的“在可行前提下尽量压缩死区”提供了依据。

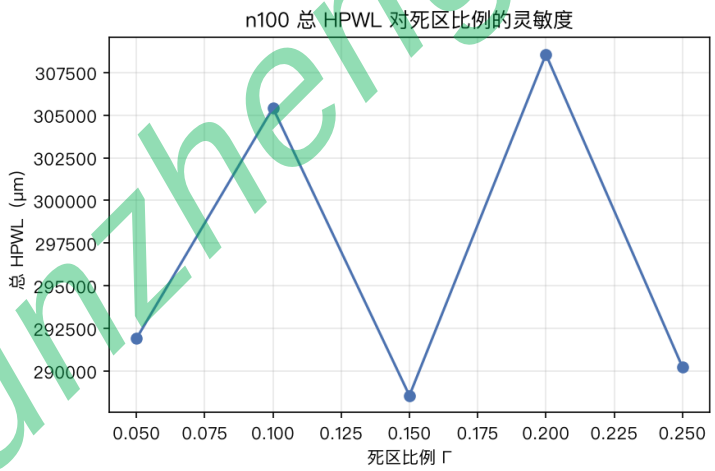

图 19　n100 总 HPWL 对死区比例 Γ 的灵敏度

8.4 收敛性检验

图 5 与图 11 显示目标函数随迭代次数单调下降并趋于稳定，表明退火过程收敛；结合多种子实验的稳定性，可认为算法已逼近其搜索能力范围内的较优解。

9 模型评价与推广

9.1 模型优点

(1) **表征紧凑、评价高效。** B*-tree 解空间远小于序列对（$O(n!\,2^{2n}/n^{1.5})$ 对比 $(n!)^2$），配合轮廓法 $O(n)$ 打包，使 SA 能在合理时间内处理 $n = 300$ 规模。

(2) **目标函数设计灵活。**面积、长宽比、HPWL、轮廓越界均可通过加权统一到一个标量目标中，四问复用同一算法框架，便于实现与对比。

(3) **问题 4 精确求解。**对 L/T 型模块采用网格化 + 穷举回溯，得到 100% 利用率的全局最优，结果可严格证明，说服力强。

9.2 模型缺点

(1) **启发式解的质量受搜索规模制约**。模拟退火为随机启发式，n300 等大规模算例与全局最优仍有约 10% 死区差距，可通过 Fast-SA 三阶段退火、多起点并行等进一步提升。

(2) **HPWL 仅为一阶线长估计**。未考虑真实布线中的 Steiner 点、拥塞与多层布线，实际线长可能略高于 HPWL。

(3) **网格化后单元粒度影响**。问题 4 的网格单元取题图基本单元，若模块尺寸非整数倍单元，需更细网格或连续坐标处理。

9.3 推广方向

(1) **软模块（soft block）**。允许模块在面积不变下调整宽高比，可在 SA 中增加“整形”扰动，进一步压缩死区，适用于固定轮廓布图。

(2) **考虑布线拥塞与总线规划**。在 HPWL 之外引入拥塞代价、总线对齐约束，即可推广到总线驱动布图（BDF）问题。

(3) **三维与多热源约束**。对 3D 集成电路或含热、功耗约束的布局，可将 B*-tree 推广为三维表征并加入温度/功耗惩罚项。

(4) **与强化学习结合**。用深度强化学习直接学习放置策略，替代手工设计的扰动算子，适应更大规模与更多约束。

参考文献

[1] Bohrium 科学百科. 半周长线长 HPWL[EB/OL]. `https://www.bohrium.com/sciencepedia/feynman/keyword/half_perimeter_wirelength`.

[2] Chen T C, Chang Y W. Modern floorplanning based on B*-tree and fast simulated annealing[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2006, 25(4): 637–650.

[3] 黄志鹏, 李兴权, 朱文兴. 超大规模集成电路布局的优化模型与算法 [J]. 运筹学学报, 2021, 25(3): 15–36.

[4] Chang Y C, Chang Y W, Wu G M, et al. B*-trees: A new representation for non-slicing floorplans[C]//Proceedings of the 37th Annual Design Automation Conference. ACM, 2000: 458–463.

[5] Kirkpatrick S, Gelatt C D, Vecchi M P. Optimization by simulated annealing[J]. Science, 1983, 220(4598): 671–680.

[6] Adya S N, Markov I L. Fixed-outline floorplanning: Enabling hierarchical design[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2003, 11(6): 1120–1135.

10 附录：关键代码

10.1 B*-tree 轮廓法打包核心

Listing 1 B*-tree 轮廓法打包（确定模块坐标）

```python
def pack(self):
    n = self.n
    self.x = np.zeros(n); self.y = np.zeros(n)
    stack = [(self.root, 0.0, self.w[self.root])]
    order = []
    while stack:                    # 先序遍历求 x
        i, xi, wi = stack.pop()
        self.x[i] = xi; order.append(i)
        if self.right[i] != -1:
            stack.append((self.right[i], xi, self.w[self.right[i]]))
        if self.left[i] != -1:
            stack.append((self.left[i], xi + wi, self.w[self.left[i]]))
    skyline = [(0.0, 1e18, 0.0)] # 轮廓：每段 (l, r, height)
    max_x = max_y = 0.0
    for i in order:                 # 按先序用轮廓求 y
        xi, xr = self.x[i], self.x[i] + self.w[i]
        ymax = 0.0
        for (l, r, hh) in skyline:
            if r > xi and l < xr and hh > ymax:
                ymax = hh
        self.y[i] = ymax; yt = ymax + self.h[i]
        max_x = max(max_x, xr); max_y = max(max_y, yt)
        new = []; placed = False
        for (l, r, hh) in skyline: # 更新轮廓
            if r <= xi or l >= xr:
                new.append((l, r, hh))
            else:
                if l < xi: new.append((l, xi, hh))
                if not placed: new.append((xi, xr, yt)); placed = True
                if r > xr: new.append((xr, r, hh))
        skyline = [s for s in new if s[0] < s[1]]
    return max_x, max_y
```

10.2 HPWL 计算

Listing 2　总 HPWL 计算

```
def compute_hpwl(self, nets, block_index, pl):
   total = 0.0
   cx = self.x + self.w / 2.0
   cy = self.y + self.h / 2.0
   for net in nets:
      mnx = mny = 1e18; mxx = mxy = -1e18
      for node in net:
         if node in block_index:
            i = block_index[node]; px, py = cx[i], cy[i]
         else:
            px, py = pl.get(node, (0.0, 0.0))
         mnx = min(mnx, px); mxx = max(mxx, px)
         mny = min(mny, py); mxy = max(mxy, py)
      total += (mxx - mnx) + (mxy - mny)
   return total
```

10.3 问题 4 回溯精确搜索核心

Listing 3　L/T 型模块最小包围面积的回溯搜索

```
def pack_into(W, H):
   grid = np.zeros((W, H), dtype=int)
   def backtrack(k):
      if k == 4: return True
      for cells in rots[k]:  # 尝试该模块所有朝向
         w = max(c[0] for c in cells) + 1
         h = max(c[1] for c in cells) + 1
         for ox in range(W - w + 1):
            for oy in range(H - h + 1):
               if all(grid[ox+cx, oy+cy] == 0 for cx, cy in cells):
                  for cx, cy in cells: grid[ox+cx, oy+cy] = k + 1
                  if backtrack(k + 1): return True
                  for cx, cy in cells: grid[ox+cx, oy+cy] = 0
      return False
   return backtrack(0)
# 枚举包围框尺寸，取面积最小者
for W in range(1, 8):
   for H in range(1, 8):
      if W * H >= total_area and pack_into(W, H):
```

```
            best = W * H; best_wh = (W, H)
```

完整代码见随附求解目录：求解/布图规划求解器.py、求解/问题一/问题一.py 至求解/问题四/问题四.py。