VLSI实验报告

一、算法逻辑与实现思路

核心目标

实现基于**模拟退火（SA）**与**FM局部优化**的双分区算法，在满足分区大小约束（48%~52%）的前提下，最小化割代价（切割边的数量）。

关键步骤

1. 初始化分区

- **度数贪心策略**：按节点度数降序排列，优先将度数高的节点加入X分区，直至X达到 `min_size`（48%）。剩余节点加入Y，确保初始分区满足大小约束。

```
//基于度数贪心策略
vector<pair<int, int>> node_degrees;
for (const auto& node : nodes) {
    node_degrees.emplace_back(node->get_index(), node->get_nets().size());
}
sort(node_degrees.begin(), node_degrees.end(),
    [](const auto& a, const auto& b) { return a.second > b.second; });

X.clear(); Y.clear();
for (const auto& [node_id, _] : node_degrees) {
    if (X.size() < min_size || (X.size() < max_size && Y.size() >= max_size)) {
        X.insert(node_id);
    } else {
        Y.insert(node_id);
    }
}
```

2. 模拟退火框架

- **温度参数**：初始温度设为当前割代价的10倍（`T = 10 * current_cost`），冷却速率0.98，最低温度1e-6，最大迭 1000次。
- **邻域搜索**：每轮随机选择一个节点尝试跨分区移动，若违反大小限制则撤回。

- Metropolis准则：
  - 若移动后割代价降低，直接接受。
  - 若割代价增加，以概率 `exp(-Δ/T)` 接受较差解，避免陷入局部最优。

```
// 模拟退火参数
double T = 10 * current_cost; // 初始温度
cout << "Initial Temperature: " << T << endl;
double cooling_rate = 0.98;
double min_T = 1e-6; // 最小温度
int max_iterations = 1000; // 最大迭代次数
int iteration = 0;
```

3. 周期性FM局部优化

- 每25次迭代调用 `FM_local_optimize`：
  - **增益计算**：基于当前分区，计算节点移动后的割代价变化（`external_cost - internal_cost`）。
  - **最大增益选择**：优先移动增益最大的节点，锁定已移动节点防止循环。
  - **增益更新**：更新邻居节点的增益表，保持状态一致性。

```
if (iteration % 25 == 0)
    FM_local_optimize(graph, X, Y);
```

4. 保留最佳解

- 记录全局最佳割代价对应的分区（`best_X` 和 `best_Y`），最终输出最优结果。

实现细节

- **分区约束**：移动节点时严格检查 `min_size` 和 `max_size`，确保不越界。
- **时间复杂度**：SA的迭代次数（1000）与FM的增益计算（O(节点数×边数)）共同决定整体效率。
- **随机性**：初始分区和SA的随机移动引入多样性，多次运行可提升解质量。

二、算法运行结果

数据集	割代价	与最优解的距离(欧氏距离)
ibm01.hgr	1319	95.5615
ibm02.hgr	5818	101.44
ibm03.hgr	3481	95.8541
ibm04.hgr	5534	132.812
ibm05.hgr	7026	106.151
ibm06.hgr	4471	128.697
ibm07.hgr	8069	145.781
ibm08.hgr	10752	167.443
ibm09.hgr	7993	176.963
ibm10.hgr	8639	195.597
ibm11.hgr	8749	221.777
ibm12.hgr	15453	203.561
ibm13.hgr	12426	175.031
ibm14.hgr	21686	268.516
ibm15.hgr	18860	287.955
ibm16.hgr	31429	327.918
ibm17.hgr	46389	332.896
ibm18.hgr	21434	370.742

三、实验总结

方法优点

1. **混合策略有效性**
    - **全局探索（SA）+ 局部优化（FM）**：模拟退火提供跳出局部最优的能力，FM算法快速

优化局部邻域，二者结合显著提升解的质量。

- **动态平衡**：SA的温度调度（从高温到低温）逐步降低随机性，从探索转向开发，避免过早收敛。

2. **约束满足能力**
   - 严格检查分区大小约束（48%~52%），移动节点时自动撤回越界操作，确保分区合法性。
3. **周期性优化机制**
   - 每25次迭代调用FM优化，弥补SA在局部搜索上的不足，加速收敛。
4. **多样性保留**
   - 随机初始化和Metropolis准则接受较差解，增加解的多样性，避免单一策略的局限性。

方法缺点

1. **计算开销较大💀（最显著的缺点）**
   - 自己的电脑计算能力不足，在追求性能和结果优化上难以做到平衡。
   - FM局部优化需频繁计算增益并更新邻居节点，时间复杂度较高，尤其在大规模图上表现明显。
   - SA的 1000次迭代上限可能导致收敛速度不足，无法适应超大规模实例。
2. **FM优化的局部性限制**
   - 当前FM实现仅关注直接邻居的增益更新，未处理间接影响的远距离节点，可能遗漏全局优化机会。
   - 增益计算未考虑多节点协同移动（如交换操作），限制局部优化潜力。
3. **随机移动效率低**
   - SA的随机节点选择缺乏方向性，尤其在低温阶段可能频繁尝试无效移动，浪费计算资源。
4. **温度调度固化**
   - 固定冷却速率（0.98）和初始温度（10倍当前割代价）缺乏适应性，无法动态调整。
5. **缺乏多样性机制**
   - 仅依赖单次SA迭代，未引入重启策略或多线程并行搜索，可能陷入次优解。

下一步改进方向

1. **优化计算效率**
   - **增量式增益更新**：仅更新受移动节点直接影响的网络和节点的增益，避免全量计算。
   - **并行化SA迭代**：利用多线程同时探索多个温度链，提升搜索速度。
2. **增强FM的全局性**
   - **多节点协同移动**：支持节点交换或组移动策略，扩大局部搜索范围。
   - **动态边界节点维护**：精确识别并优先处理边界节点，减少冗余计算。
3. **改进SA策略**
   - **自适应温度调度**：根据割代价变化动态调整冷却速率（如快降初期、慢降后期）。
   - **方向性扰动**：结合节点增益或度数的概率分布选择移动节点，提升扰动有效性。
4. **引入多样性机制**
   - **多起点重启**：运行多次SA，从不同初始分区出发，保留全局最优解。
   - **种群进化策略**：维护多个候选分区，通过交叉和变异生成新解。
5. **混合深度学习**
   - **强化学习调参**：自动优化温度、冷却速率等超参数，适应不同图结构。**（需要更多计算资源）**

总结

当前方法通过混合SA与FM算法，在中小规模图上表现出较好的平衡性与解质量，但计算效率和全局搜索能力仍有提升空间。