# VLSI 课程总结

-----最小直角斯坦纳树布线问题的
粒子群优化(RSMT_PSO)算法实现与测试

计 23 黄必胜
计 23 肖迪
计 23 章晔

## 算法介绍

- 项目名称
  最小直角斯坦纳树(RSMT)布线问题的粒子群优化算法实现与测试

- 背景
  最小直角斯坦纳树( RSMT) 问题是超大规模集成电路布线中的重要问题之一,是典型的 NP 困难组合优化问题。研究最小直角斯坦纳树问题是为了有效地解决超大规模集成电路布线中的 RSMT 问题。

- 基本思想
  VLSI 布线问题中连接树的目标是连接树的总长度最短,对于多端线网的最佳布线结果是构造最小直角Steiner树(RSMT)。这是一个NP完全问题，研究者提出了许多基于智能优化技术的解决方法,包括了本文中提到的粒子群优化算法。
  粒子群优化算法 PSO 是 Kenney 何 Eberhart 于 1995 年提出的一种基于种群搜索策略的自适应随机算法，它源于对鸟群和鱼群等群体运动行为的研究，是一种基于迭代的智能优化算法，可用于求解大部分优化问题。
  我们利用一种粒子群优化算法,借助直角 Steiner 树的一些性质,采用 Steiner 点编码方案,寻找优化的 Steiner 点位置以减少直角 Steiner 树的长度，解决超大规模集成电路布线中的RSMT问题。

- 输入输出
  输入：布线结点
  输出：一个布线拥挤较低，且布线灵活度更高的线网结构

## 算法实现

- 建模与分析
  1. RSMT问题模型
  斯坦纳树是一棵连接特定要求点集合和一些斯坦纳点的连接树。由于它连接树总长度比其他方法更小,常被用作总体布线中构造连接树的方法。因此，VLSI 总体布线问题可以看作是在总体布线图中，目标函数最优化的条件下，针对 每个线网寻找一棵斯坦纳树的问题。通常典型的目标函数是所选择的连接树的总长度最小。

定义1. 一棵斯坦纳树的长度为所有边长度之和。

定义2. 一棵斯坦纳树中，若该斯坦纳树的每条边均为直角矩形边，则此斯坦纳树称为直角斯坦纳树( rectilinear Steiner tree，RST)。

定义3. 给定一个无向图 G( V, E)，一个要连接的端点集合N，最小 Steiner 树( SMT) 就是一棵通过 V 中的点连接 N 中所有点的生成树，以边长最短为目标。与欧氏距离 Steiner 树不同，最小直角 Steiner 树( rectilinear Steiner minimal tree,RSMT) 两点间的距离是直线距离，横轴距离和纵轴距离之和，即连线只有水平和垂直两种形式。

**定义4.** Hanan点：在RSMT问题的布线端点集合N中，通过这些端点分别各自引一条水平和竖直线，这些线的交叉点形成Hanan点集合。从Hanan点集合中按一定规则选取部分点作为Steiner点集合。这些Steiner点和原来点的生成树构成RSMT。

**2. 最小直角Steiner树(RSMT)的相关性质：**

**性质1. 设RSMT有n个端点，则RSMT的Steiner点个数 ≤ n-2。**

**性质2. (Hanan定理)任意一棵RSMT的Steiner点均在它的Hanan网格上。**

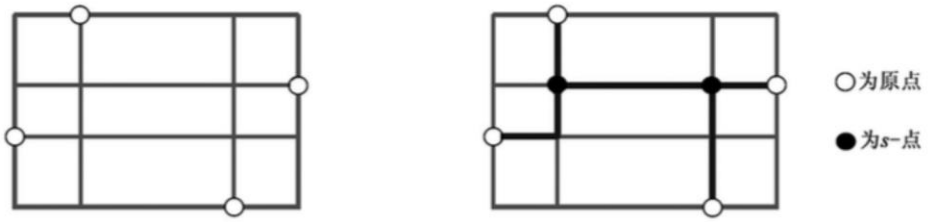



图 1 Hanan 定理

Fig. 1 Hanan theorem

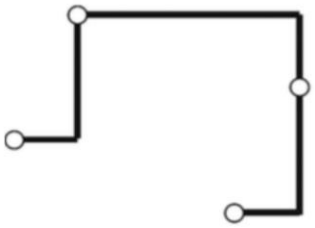


图 2 最小生成树

Fig. 2 Minimum spanning tree

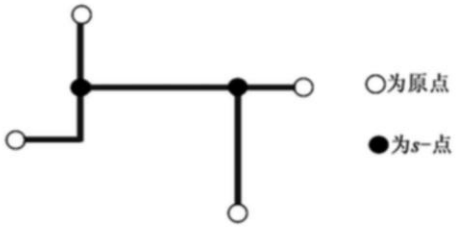



图 3 最小直角斯坦纳树

Fig. 3 Rectilinear Steiner minimal tree

算法步骤：

以下是求解 RSMT 布线问题的 PSO 算法( PSO-RSMT) 具体步骤:

输入:n个端点的坐标位置;

输出:RSMT 的值和对应 s 点坐标位置;

Step1:读入数据集，生成 Hanan 点;

Step2:初始化算法参数并随机生成初始群;

Step3:评价各个粒子的适应度函数值，并初始化个体最优解和全局最优解;

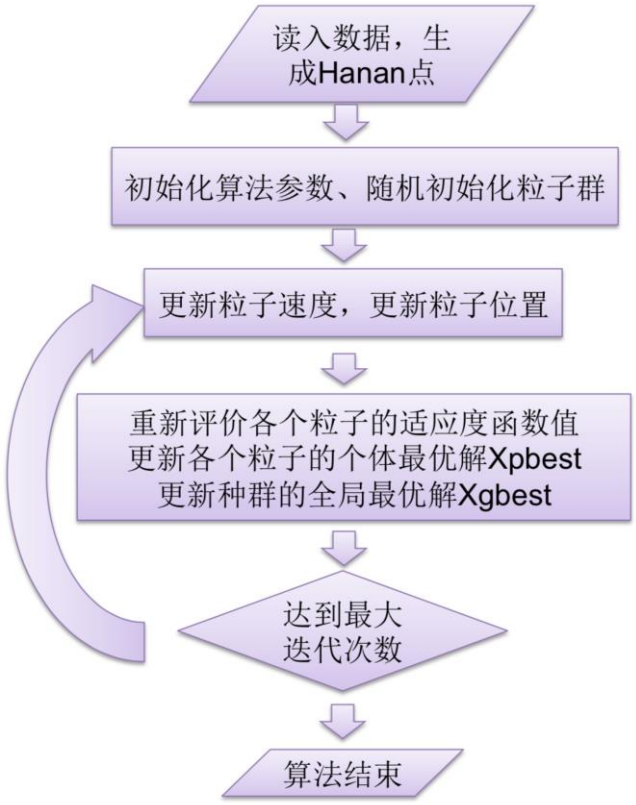
Step4:计算惯性权值w，更新粒子速度和粒子位置，动态更新s点数量;

Step5:重新评价各个粒子的适应度函数值，并更新各个粒子的个体最优解;

Step6: 更新种群的全局最优解;

Step7: 若满足终止条件，则循环结束，否则返回 Step4。

- 算法结构框图




- 主要数学过程：

1. **粒子的编码**

- 由n-2个Hanan点的坐标位置构成: X=(x1,x2,x3,...,xi,...,xn-2),其中第i维数据xi表示这个Hanan点在平面上的坐标
- 速度v也是由n-2个二维向量组成

2. **适应度函数：**

$$\text{fitness}(X) = L - \sum_{e} l(e), e \in \text{RSMT}(N \cup S_1),$$

- 用于判断种群进化过程中粒子所在位置的优劣。

3. **粒子速度和位置更新**

$$v_{ij}^{t+1} = wv_{ij}^{t} + c_1 r_1 \left( \mathrm{Xpbest}_{ij}^{t} - x_{ij}^{t} \right) + c_2 r_2 \left( \mathrm{Xgbest}_{ij}^{t} - x_{xj}^{t} \right),$$

$$x_{ij}^{t+1} = x_{ij}^{t} + v_{ij}^{t},$$

- w是惯性系数，$c_1$和$c_2$ 是学习因子，$r_1$和$r_2$是介于[0,1]之间的随机数，$x_{ij}^{t}$是t时刻粒子i的第j维分量值，$v_{ij}^{t}$是t时刻粒子i的第j维速度分量值。

**4. 惯性系数**

$$w(t) = 0.9 - \frac{t}{\mathrm{MaxNumber}} \times 0.5,$$

- 其中MaxNumber是算法的最大迭代次数。

- 关键代码实现：

该部分将对核心的代码部分进行解释说明。

***PSO 粒子信息***

首先是 PSO 粒子的信息初始化部分，这些变量记录着当前各粒子的运动信息，在接下去的每一次迭代，都是在当前粒子的信息基础上进行，利用位移公式和速度公式进行每一轮的迭代。

```
const int N_particle = 50;      //粒子个数
int N_steiner_particle[N_particle] = {0};   //各粒子的 steiner 点个数
pair<double,double> v_particle[N_particle][N_maxpoint] ;      //当前速度
pair<double,double> pos_particle[N_particle][N_maxpoint];   //当前位置
pair<int, int>          edge_particle[N_particle][N_maxpoint];   //最小生成树边表
```

每一个粒子包含的信息是它所对应的若干个 steiner 点。

其中 steinter 点的个数由 N_steiner_particle 指定，按照算法流程，每个粒子先初始化该变量为 rand(0,N-2)，其中 N 为输入点数，之后它会在迭代的过程中会动态地增长。

二维变量 pos_particle 保存着各个粒子的 steiner 点的坐标。其中第一维指定了粒子序号，第二维指定了 steiner 点序号，每个坐标是一个二维向量 pair<double, double>。

***迭代部分***

算法的主要环节，迭代部分伪代码：

算法的终止以迭代达到指定为标准。

```
while(n_iter++ < N_MAXITER) //迭代过程主循环
{
    for(int particle = 0; particle != N_particle; particle++)//枚举每一个粒子
    {
        for(int dim = 0; dim != N_steiner_particle[particle]; dim++)//枚举每一维
        {
            //利用速度更新式子，更新粒子 dim 维的速度
            //更新粒子 dim 维的位置
            //更新 steiner 点个数
            //根据需要更新个体最优解和全局最优解
        }
    }
}
```

***最小生成树及复杂度***

算法的主要运算部分在于最小生成树的计算。对于每一个粒子而言，求适应值的过程是，首先在原有的输入点基础上，添加进该粒子所对应的 steiner 点，接下去对所有的点求最小生成树的边长和。

由于任意两个点间都有可能建立边连接，所以待求最小生成树的是一个边稠密的完全图，可以通过 PRIM 算法来求解。

设输入点个数为N，粒子数为K，迭代次数为I，PRIM 算法的复杂度为 $O(n*n)$，其中 n 为图中点的个数。在该算法中$\mathrm{n}=N+N-2$，则 PSO 算法的复杂度为 $\mathrm{O}(KI(N+N-2)^2)$。

## 结果分析

- 总述

***实验配置***

编程语言：C++

运行平台：Win7
输入端点的坐标范围：x、y 坐标都在 0~10000 间随机。
PSO 参数：粒子数 = 50，迭代次数 = 500。
速度学习参数：c1 = 1.0，c2 = 1.0。

- 实验结果分析

***性能分析***

| 输入端点数 | 最小生成树长度 | RSMT 长度 | RSMT 的减小幅度 | 运行时间/MS | 文章参考时间 |
|---|---|---|---|---|---|
| 9 | 22828 | 20920 | 8.4% | 768 | 86 |
| 12 | 29154 | 25821 | 11.4% | 1535 | 192 |
| 15 | 34503 | 31708 | 8.1% | 2629 | 675 |
| 18 | 38830 | 34609 | 10.9% | 4354 | 1253 |
| 20 | 41254 | 37057 | 10.2% | 5555 | 2863 |

从表中可以看到，程序运行的时间与文章的参考时间相比有一定的差距，花费更多的时间。可能的原因如下：

1、 文章的算法参数并未写全，迭代次数和 POS 粒子数等的参数设置可能有不同，因而造成了差异。

2、 其次，如果查看时间随输入端点数的增长速度会发现，我们的程序运行时间增长得较慢，与文章中的算法时间越来越接近，在更大的输入端点数上甚至有可能获得更短的运行时间。

所以在运行时间上与参考文章中的参考时间应该差异不大.

从 RSMT 长度相对于最小生成树的减小幅度上看，算法具有较好的性能，在这几个例子中达到了 9.8±1.5%的减小幅度。

***粒子收敛过程***

该算法有较好的收敛性，我们以以 N = 9 的例子来说明。

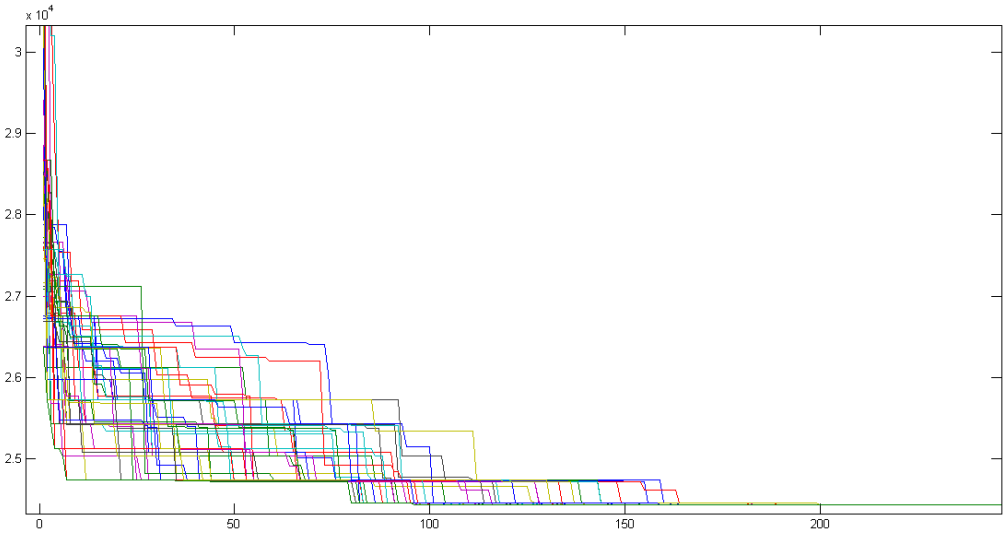



图 1 各粒子的个体最优解随迭代次数的变化

图表中横轴为迭代的次数，纵轴为粒子所对应个体最优解的 RMST 树长。其中每一条轨迹对应着一个粒子的运动状况。

从图中可以了解到以下结论，

1、粒子的运动具有整体性。各个粒子有一致的运动表现，即紧跟当前最优解，当一个粒子有结果的改进时，其余粒子也会很快地跟上。

2、粒子在迭代过程中没有发生突发性的早熟，收敛的速度比较平均和稳定。

3、在比较小的迭代次数范围内已经足够收敛。算法在大约 170 次迭代的时候就已经达到全局的最优解。而算法参数上我们设置了 500 作为迭代的总次数，这可能在一定程度上限制了程序的运行速度。可能采取更灵活的迭代次数选择。

***实验结论***

经试验，PSO 算法有较好的的收敛性，在 RMST 长度和运行时间方面也具有较好的性能。在尝试的例子中，相对于最小生成树长度，RMST 达到了 9.8±1.5%的减小幅度，在运行时间上也与参考实现的文章一致。

由此我们验证了一下 PSO 算法的特性：PSO 算法模拟了群体模型中的信息共享机制，由于粒子的运动紧跟当前最优解，并且惯性系数可以调整其搜索能力，防止早熟， 因此在搜索过程中用较小规模的种群就可以收敛到最优解.

## 实验总结

- 实验分工
  肖迪：算法调研、算法实现代码测试、展示 PPT 制作、总结报告撰写
  黄必胜：算法调研、实现代码编写与代码测试、总结报告撰写
  章晔：算法调研、算法实现代码测试、选题报告撰写

- 计划跟进

1、 初期计划

初期的时候我们进行了详尽的项目调研和前期规划，我们将团队计划安排如下：

- 细读相关文献，撰写选题报告
  对于不熟悉的知识点及时查找资料或与同学老师交流，将算法的相关实现细节弄明白。

- 寻找实现算法必要的工具
  如在本算法中，在获取初始线网 $T_0$ 的时候需要使用 FLUTE 算法，考虑寻找开源的工具实现。

2、 中期计划：

- 中期的时候我们进行了算法实现的尝试，并询问助教老师和任课老师算法实现的可行性。并及时对我们的项目进行修改。

3、 算法实现：

- 实现最小直角斯坦纳树布线问题的粒子群优化(RSMT_PSO)算法。

4、 实验测试：

- 对测试用例进行算法测试，与文章中的实现结果进行对比。

5、 实验总结：

- 撰写总结报告，完善项目中不完整的地方，定下最终版本，并对本次 实验进行反思。

- 项目心得

1、 制定好详细的计划

在小组任务当中，详细的计划是非常重要的。我们需要先对任务拥有一个整体的认知。再根据任务量和项目时间，在每个时间段安排相应合理任务量的任务。最后根据每个组员的具体能力和时间，给每个人分配相应的任务。

我们将这个项目当做一个小的软件工程来进行敏捷开发，每周为一个 sprint，在每个 sprint 中分别制定迭代开发的计划。每周对项目进度进行跟进，解决相应问题，使得整个开发进程十分流畅。

2、 合理的分工与合作

我们根据团队中每个人的能力与擅长的领域进行分工，分工的合理使得我们的开发效率非常高。在项目过程中遇到任何问题也解决得非常迅速。

3、 可视化

由于我们会在课程最后，对我们的项目进行一个展示。所以我们在项目开始的时候就构思用什么样的方式才能最好地将我们的成果展示出来。

最终我们选择了 QT 语言，制作可视化界面，进行 demo 展示。为了保障展示的时候不出现意外的 bug，在我们确定算法思想和实现代码不存在问题之后，在展示之前，我们先录制好视频，然后在现场演示部分通过播放 demo 视频的方法进行展示。当时也得到了助教老师的表扬。