**基于竞争粒子群优化的X结构Steiner最小树构造算法**

周茹平1, 刘耿耿1，\*, 郭文忠1

1.福州大学 数学与计算机科学学院,福建省 福州市 350100

摘 要：粒子群优化(Particle Swarm Optimization, PSO)算法因其高效的搜索能力和自组织能力，成为构建总体布线最佳连接模型，即X结构Steiner最小树(X-architecture Steiner Minimum Tree, XSMT)的有力工具。然而，PSO算法由于开发强度过大，容易陷入局部最优。为平衡PSO的开发与探索能力，提高种群多样性，获得更优解，本文提出了基于竞争粒子群优化算法(Competitive Swarm Optimizer, CSO)的XSMT构建算法(CSO-XSMT)。该算法使用配对竞争以及轮盘赌的方法，挑选粒子的学习对象，以增强种群的探索能力，提高算法的寻优能力。此外，为了进一步减小Steiner树线长，本文提出了一种基于公共边的精炼策略，对CSO寻优得到的Steiner树进行调整，提高最终布线树的质量。实验结果证明，相比其它Steiner树构建算法，本文所提算法具有更强的线长优化能力以及算法稳定性。

**关键词:** 竞争机制、轮盘赌策略、X结构Steiner 最小树、线长优化、精炼策略

**CSO-based X-Architecture Steiner Minimum Tree Construction**

Ruping Zhou1, Genggeng Liu1,\*, Wenzhong Guo1

1. College of Mathematics and Computer Science, Fuzhou University, China

**Abstract:** Particle Swarm Optimization (PSO), with its efficient searching ability and self-organizing ability, has become a powerful tool for constructing the best connection model for multi-terminal nets in global routing, namely X-architecture Steiner Minimum Tree (XSMT). However, PSO is easy to fall into local optimum because of its excessive exploitation intensity. In order to balance the exploitation and exploration ability of PSO, improve the diversity of population and obtain a better solution, this paper proposes an XSMT construction algorithm based on Competitive Swarm Optimizer (called CSO-XSMT). The algorithm adopts the methods of pairwise competition and roulette wheel selection to select the learning objects of particles so as to enhance the exploration ability of population and improve the optimization ability of the algorithm. In addition, in order to further reduce the wirelength of Steiner tree, a refine strategy based on sharing edges is proposed, which adjusts the Steiner tree obtained by CSO to improve the quality of the final routing tree. Experimental results show that compared with other Steiner tree construction algorithms, the proposed algorithm has better wirelength optimization capability and superior stability.

**Key words**：competitive swarm optimizer; roulette wheel selection; X-architecture Steiner minimum tree; wirelength optimization; refine strategy

# 1 引言

超大规模集成电路(Very Large Scale Integration, VLSI)物理设计中，总体布线是非常关键的一环节。作为总体布线多端线网的最佳模型，Steiner最小树(Steiner Minimum Tree ,SMT)通过引入额外的点来构建具有最小代价的布线树以连接给定引脚，是近年来VLSI的研究热点之一。

SMT的工作按照布线树的走线方向可分为曼哈顿和非曼哈顿两种，目前大部分的研究主要基于曼哈顿结构[1-3]。然而，曼哈顿结构受到布线走线方向的限制，难以找到高质量的解。为了充分利用布线资源以提高布线质量，越来越多的学者将目光投向非曼哈顿结构。非曼哈顿结构的布线方式比起传统的水平垂直方向，多了45°和135°方向，能够更充分的搜索SMT的问题解空间。精确算法[4,5]、传统启发式算法[6,7]和群智能技术[8-14]都可用于求解非曼哈顿SMT。文献[4]基于分支限界法构造X结构Steiner树，但只适用于小规模问题。文献[5]使用精确算法的同时考虑了对非曼哈顿结构布线树的剪枝，虽然获得了较好的线长减少量，但算法复杂度却过高。为克服精确算法复杂度过高的问题，已有许多工作将启发式策略应用于非曼哈顿布线树的构造。文献[6]基于贪心的思想提出了时间复杂度为的启发式算法求解Y结构Steiner。同样基于贪心思想，文献[7]提出了基于三角收缩技术以及边替换技术两种X形生成图非曼哈顿结构Steiner树算法。传统启发式算法大多数基于贪心策略，存在极易过早收敛的弊端，而群智能算法因其计算效率高、自组织性、可扩展性和低成本等有点，逐渐成为求解VLSI布线问题的热门技术。文献[8]使用蚁群优化技术来构建Steiner树，并分别应用在曼哈顿和非曼哈顿结构的总体布线中。文献[9]提出了一个基于离散差分进化的X结构Steiner最小树算法，通过设计新的交叉和变异算子来获取高质量的解。文献[10-14]使用粒子群优化(Particle Swarm Optimization, PSO)技术来构建X结构Steiner最小树(X-architecture Steiner Minimum Tree, XSMT)。其中，文献[10-13]分别考虑了线长、障碍物、时延和拐弯数等重要优化目标。文献[14]基于自适应PSO以及混合转换策略提出了一个适用于矩形SMT和X结构SMT构建的统一算法。

基于以上的分析，本文设计并实现了一种基于竞争粒子群优化(Competitive Swarm Optimizer, CSO)的XSMT构造算法。主要的研究内容包括：

（1）提出了一种基于配对竞争以及轮盘赌的学习机制。传统的PSO通过学习个体历史最优位置(*pbest*)以及种群历史最优位置(*gbest*)执行粒子的更新，使得粒子极易受到二者的影响，逐渐陷入局部最优而难以跳出。而本文剔除了*pbest*和*gbest*，通过配对竞争的方法随机选择个体学习对象，社会学习对象则采用轮盘赌的方法进行选择，这种学习机制很大程度上增强了种群的多样性，避免粒子陷入局部极值。

（2）考虑Steiner树的特点，将变异与交叉算子结合到算法的更新公式中，以实现CSO的离散化，更好解决XSMT构造问题。

（3）在CSO算法的基础上，进一步提出了考虑公共边的精炼策略，寻找所有引脚的最佳局部拓扑结构，最大化公共边长度，进一步优化CSO寻优得到的布线树，获得最终的XSMT。

论文的架构安排如下：第二部分介绍了问题模型；第三部分阐述了CSO-XSMT算法的原理以及设计与实现；第四部分为实验结果与分析；第五部分对论文做出总结。

# 2 问题模型

表1 一个线网的引脚坐标信息

Table 1 The coordinate information of the pins

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Pi* | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| *xi*, | 1 | 3 | 5 | 6 | 7 | 9 | 10 |
| *yi* | 6 | 4 | 1 | 6 | 3 | 1 | 5 |

XSMT问题可以描述如下：给定待布线线网的引脚集合*P*={*P1*, *P2*, *P3*,…, *Pn*}，每个引脚*Pi*由坐标（*xi*, *yi*）表示，通过添加Steiner点构建一棵XSMT以连接所有给定引脚，其中布线方向除了传统的水平垂直外，还可以是45°和135°。以一组10个引脚的待布线线网为例，表1给出了10个引脚的坐标信息。

如图1(a)所示，连线L连接引脚A与引脚B相连。X结构互连模型中连线L有四种连接选择方式，如图1(b), (c), (d) , (e)所示。

**定义1 伪Steiner 点.** 为了构成代价最小的X结构Steiner最小树，所引入的额外的点称为伪Steiner点（Pseudo-Steiner Point, PSP），。

**定义2 选择0**(如图1(b)所示)**.** 选择0方式先从点A引直角边到PSP点S，再从S引45°或135°的非直角边到点B。

**定义3 选择1**(如图1(c)所示)**.** 选择1方式先从点A引45°或135°的非直角边到PSP点S，再从S引直角边到点B。

**定义4 选择2**(如图1(d)所示)**.** 选择2方式先从点A引竖直边到PSP点S，再从S引水平边到点B。

**定义5 选择3**(如图1(e)所示)**.** 选择3方式先从点A引水平边到PSP点S，再从S引竖直边到点B。



(a)连接线L (b)选择0 (c)选择1



(d)选择2 (e)选择3

图1 X结构的互连线模型

Fig.1 The routing model of X-architecture

# 3 基于竞争粒子群优化的XSMT构造算法

## 3.1 CSO

粒子群优化具有原理简单，搜索效率高的优点，已被成功地应用于许多优化问题。但是在大多数情况下，优化问题的维数增长会显著增加局部最优解的数量。而PSO算法受到种群最优位置以及个体历史最优位置的强烈影响，难以避免陷入局部最优，导致算法过早收敛。为了解决早熟收敛的问题，文献[15]提出了CSO，尝试去除*gbest*和*pbest*，种群飞行的驱动由不同粒子之间的配对竞争的学习机制提供。这种学习机制使得种群中任何粒子都可能成为潜在的领导者，粒子在进化过程中可以学习到不同优秀个体学习来提高自身，增强了种群的多样性，算法的探索与开发能力得到了很好的平衡。CSO的粒子更新公式如下：

|  |  |  |
| --- | --- | --- |
|  |  | (1) |
|  |  | (2) |

其中，是第*t*次迭代中第*k*次配对竞争中的获胜的粒子，则是对应配对竞争中失败的粒子，是当代的种群中所有粒子的平均位置，又称均值中心，是控制影响度的参数，*R1*(*k*, *t*), *R2*(*k*, *t*)和*R3*(*k*, *t*)为[0,1]之间的随机数。

在每次迭代中，CSO随机选择两两粒子进行竞争，适应度较差的粒子为输家，适应值较优的粒子为赢家，输家粒子根据公式(1-2)更新，而赢家粒子不做操作直接进入下一代，即CSO算法一次迭代中仅有一半的粒子得到更新。虽然CSO仍有传统PSO中的个体认知分量与社会认知分量，但不再保留*pbest*和*gbest*对象，个体认知表现为输家粒子向赢家粒子学习，社会认知表现为输家粒子向均值中心学习。

文献[17]通过实验证明CSO在大规模问题上表现尤为优越。为了将其应用在离散问题上，受CSO算法的启发并结合遗传算法的变异与交叉操作，本文设计了一种适用于Steiner树的离散学习策略以融入粒子的更新公式中。同时，由于均值中心离散化后算法表现不佳，所以本文采用轮盘赌的方法作为新的社会学习策略。基于这种全新的学习机制，本文提升了粒子群算法的性能。

## 3.2 粒子编码

本文采用更适用于进化算法的边点对方式进行。其中，CSO中的一个粒子代表一棵候选Steiner树。每一棵候选Steiner树由生成树的边列表和PSP选择方式(定义2-5)表示。

对于一个含*n*个引脚的线网来说，需要条边构成合法的生成树。而每条边需要由长度为3的子串(*estart*, *eend*, *pspc*)表示，该子串代表边(*estart*, *eend*)之间的PSP选择方式为*pspc*。对边进行编码后，还需要一位数值表示粒子的适应值。所以一个粒子由一个长度为的数字串编码。图2是表1引脚对应的一种X结构Steiner树布线方案，相应的粒子编码为：

5 7 *2* 5 4 *0* 2 4 *1* 5 3 *1* 2 1 *1* 5 6 *0* **18.7279**

其中粒子的编码长度为19，粒子的适应值为粗体数字18.7279，即。以第三个子串(5, 3, *1*）为例，该子串表示引脚5和引脚3之间的PSP选择为选择1。

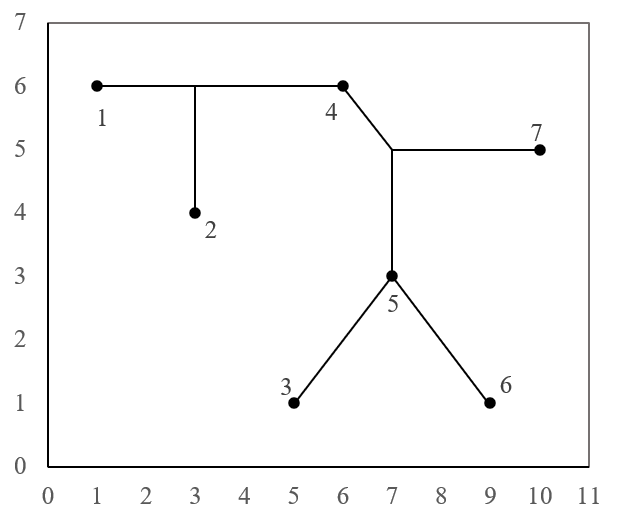


图2 表1给定引脚的一棵X结构Steiner树

Fig.2 An X-architecture Steiner tree with the given pins in Table 1

## 3.3 适应度函数

由于种群中一个粒子代表问题的一个候选解，即一棵候选Steiner树，则粒子的适应度值需要体现Steiner树的优劣。本文以Steiner树的长度作为粒子的适应度值，计算公式如下：

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

其中，*l*(*ei*)表示布线树*Tx*中边线段*ei*的长度。

## 3.4 粒子更新公式

根据3.1分析，本文算法基于CSO的竞争机制，在每次迭代中随机挑选两个粒子进行适应值的比较，通过这种方式将种群粒子划分为输家和赢家。算法仅针对输家进行更新，赢家保持不变，以下所讨论的更新操作仅用于输家粒子。

CSO中粒子遵循以下的更新公式：

|  |  |  |
| --- | --- | --- |
|  |  | (4) |

其中，为惯性权重因子，*c*1和*c*2为加速因子，SF1为惯性分量，SF2和SF3分别为新的个体认知分量和社会认知分量。在CSO-XSMT中，通过变异操作实现SF1，变异程度决定了上一次迭代的状态对当前状态的影响程度；通过交叉操作实现SF2和SF3，分别与配对竞争赢家和轮盘赌选出的对象进行部分基因交叉。

### 3.4.1 惯性分量

算法使用SF1进行粒子的变异操作，公式如下：

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

其中，Mp()为变异操作，是变异后的粒子，惯性权重决定了粒子进行变异操作的概率，*r*1是[0,1)之间的随机数。



图3 CSO-XSMT算法的变异操作

Fig.3 Mutation operator of CSO-XSMT

CSO-XSMT的变异操作针对PSP进行变换，考虑两点变换。主要变异过程为：当生成的随机数*r*1<，算法随机选择两条边，替换其PSP选择。否则，保持粒子不变。如图2所示，对于一棵含有7个引脚的布线树，算法随机选择粒子*Xl*的两条边*m*1(3, 2, *1*)和*m*2(2, 5, *3*)变换其PSP选择，变异后*m*1和*m*2的布线方式分别由选择1，选择3变成选择2和0。

### 3.4.2 个体认知分量

CSO算法中，输家粒子不再向其个体历史最优学习，而是受到赢家粒子的吸引更新自身。算法使用SF2表示粒子的个体认知分量，通过交叉操作实现，公式如下：

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

其中Cp()为交叉操作，是交叉后的粒子，*c*1决定了输家粒子与赢家粒子交叉的概率，*r*2是[0,1)内的随机数。



图4 CSO-XSMT算法的交叉操作

Fig.4 Crossover operator of CSO-XSMT

当产生的随机数，执行交叉操作：对一棵含*n*个引脚的Steiner树，算法首先随机确定需要交叉的边区间，并找到赢家粒子对应区间上的编码，然后用赢家粒子的编码替换输家粒子该区间上的编码。如图3所示，粒子是经过变异后的输家粒子，粒子是配对竞争中适应值更优的赢家粒子。算法选择输家区间(即图中红色的边*e*1, *e*2和*e*3)进行交叉，赢家对应边的编码：12*1* 25*0* 42*3*，即图中蓝色的边；将粒子该区间上编码替换为上述编码串的值。交叉操作后得到的粒子边*e*1，*e*2和*e*3的布线方式分别由选择2，选择0和选择3转变成选择1，选择0和选择3。

上述交叉操作使粒子学习到了更优秀的粒子的部分基因，经过多次迭代学习，粒子可以逐渐向最优解位置靠拢。

### 3.4.3 社会认知分量

CSO算法使用轮盘赌的方法随机挑选粒子的社会学习对象，每个粒子的概率根据如下公式计算：

|  |  |  |
| --- | --- | --- |
|  |  | (7) |
|  |  | (8) |

其中*pop*[*i*].fit\_value是种群中第*i*个粒子的适应值，*poolVec*[*i*].fit\_value是按适应值升序排序后种群中第*i*个粒子的适应值。

从公式可以看出，粒子越优秀(即适应值越小)，其被选择的概率越低，从而避免算法过快收敛至局部极值。

算法使用SF3表示粒子的社会认知分量，通过交叉操作实现，公式如下：

|  |  |  |
| --- | --- | --- |
|  |  | (9) |

其中，是经过以上所有操作后得到的粒子，是通过轮盘赌方法选出的粒子，*c*2决定了粒子与交叉的概率，*r*3是[0,1)内的随机数。具体的交叉操作与SF2相同，这里不再赘述。

## 3.5 精炼策略

CSO-XSMT算法通过CSO中粒子的迭代更新寻找代价最小的布线树，然而由于算法的随机性，大多数情况下，CSO所得到解离未知的最优解还有一定的距离。

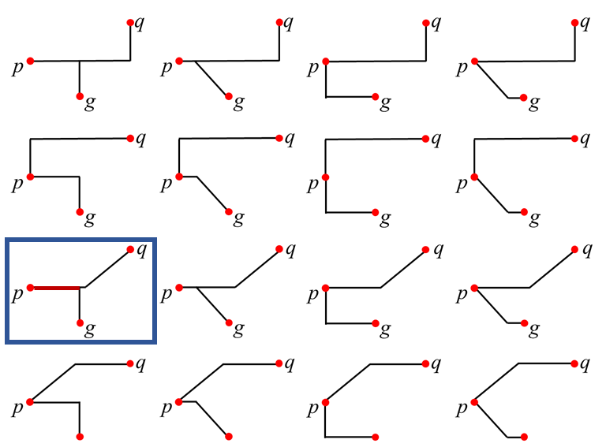


图5 具有两个度的引脚p的16种拓扑结构

Fig.5 16 topologies with pin p of two degrees

对于XST的任何一个引脚*pi*，其所有互连方式中至少有一个最佳结构。图4显示了2度引脚*p*的所有局部拓扑结构，共有16种。其中蓝色矩形框中的就是其最佳局部拓扑结构。由于引脚*p*与引脚*q*以及引脚*g*相连，走线会产生一段公共边，即图中的红色边。不难看出，布线所共享的公共边越长，线长代价越小。所以，本文提出一种基于公共边的精炼策略，具体细步骤如下：

**步骤1：**计算每个引脚*pi*的度，并记录*pi*的邻接引脚表。

**步骤2：**对于每个引脚*pi*，若其度数大于1，计算其公共边长度。

**步骤3：**根据公共边长度对每个引脚按降序排序。

**步骤4：**依次枚举每个引脚的所有布线选择组合，选择具有最小线长的组合作为该引脚的最佳局部拓扑结构，直到所有引脚遍历完成。

## 3.6 算法总体流程

CSO-XSMT的算法总体步骤如下：

**步骤1：**初始化参数，并基于最小生成树的方法构建布线树作为初始种群。

**步骤2：**随机挑选两个粒子进行配对竞争。

**步骤3：**使用轮盘赌选择社会学习对象。

**步骤4：**根据公式(5)、（6）、（9）对输家粒子进行更新，赢家不更新，直接进入下一代。

**步骤5：**如果M/2个输家粒子都得到了更新(种群大小为M)，则进行下一次迭代。否则返回步骤2。

**步骤6：**如果满足终止条件(达到设定的最大迭代次数)，则执行精炼策略(3.5节步骤1-4)，否则返回步骤2进入下一次迭代。

**步骤7：**输出精炼得到的最优解。

## 3.7 算法复杂性分析

**引理1:** 假设种群大小为*M*，迭代次数为*T*，引脚个数为*P*, 则CSO-XSMT算法的时间复杂的为。

**证明：**CSO-XSMT算法可以分为两个部分，分别是CSO搜寻XST和精炼策略求解XSMT。在CSO寻优过程中，内部循环主要包括配对竞争、轮盘赌机制、变异交叉操作以及适应值计算。其中，配对竞争、变异和交叉操作的时间复杂度皆为常数时间*O*(1)。对于轮盘赌概率的计算，主要为排序所花的时间，即。粒子适应值计算同样由排序时间决定，为。故算法内部循环的复杂度为。考虑外部循环条件种群规模*M*和引脚数*P*，这部分的近似时间复杂度为。

算法进行精炼操作时，计算每个引脚的度数、其邻接点列表以及计算公共边所花时间为*O*(*P*)，对所有引脚按公共边排序需要时间。算法遍历*n*个引脚进行局部拓扑的调整，时间复杂度为*O*(*P*)。所以精炼策略近似时间复杂度为。

综上，CSO-XSMT算法的近似时间复杂度为。

# 4 实验

为了验证CSO-XSMT算法和相关策略的有效性，本文在基准电路GEO[16]上进行实验。算法设置种群大小为100，迭代次数为1000，其余参数与文献[20]一致。考虑到算法的随机性，所有实验数据取10次运行结果的平均值。

## 4.1 CSO学习机制有效性验证

为验证CSO算法所提出的学习机制的有效性，本节将CSO与DPSO(X)[17]进行比较。实验对比了两种算法得到的最佳线长，平均线长与标准差。如表2所示，在测试电路4-10上，CSO算法都能找到比DPSO(X)更好的线长。其中，CSO对测试电路7的优化效果最好，得到的平均线长比DPSO(X)优化了0.923%。对于实验测试的三个指标，CSO分别为达到了0.317%，0.399%，38.24%的优化率。分析原因，CSO的学习机制扩大了粒子学习对象的范围。而DPSO(X)方法只能通过学习*gbest*以及*pbest*进行粒子更新，无法保证种群多样性，算法平衡开发与探索的能力较差。且DPSO(X)中落后的粒子与*gbest*的距离较远，一次更新可能产生很大的位置改动，算法稳定性较差。而基于轮盘赌的方法，从一定程度缓解了这种跳跃的情况，从而提高算法稳定性。

表2 CSO和DPSO(X)算法性能对比

Table 2 Comparison between CSO and DPSO(X) algorithms

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 电路 | 引数 | 最佳线长 | | | 平均线长 | | | 标准差 | | |
| DPSO(X)  (X) | CSO | 优化率(%)  ()  (%) | DPSO(X)  (X) | CSO | 优化率(%)  (%) | DPSO(X)  (X) | CSO | 优化率(%)  (%) |
| 1 | 8 | 16918 | 16918 | 0.000 | 16918 | 16918 | 0.000 | 0 | 0 | 0.00 |
| 2 | 9 | 18041 | 18041 | 0.000 | 18041 | 18041 | 0.000 | 0 | 0 | 0.00 |
| 3 | 10 | 19696 | 19696 | 0.000 | 19696 | 19696 | 0.000 | 0 | 0 | 0.00 |
| 4 | 20 | 32214 | 32193 | 0.065 | 32214 | 32197 | 0.019 | 10 | 6 | 40.00 |
| 5 | 50 | 48039 | 47959 | 0.167 | 48045 | 47965 | 0.118 | 74 | 7 | 90.54 |
| 6 | 70 | 56295 | 56277 | 0.032 | 56548 | 56313 | 0.191 | 138 | 41 | 70.29 |
| 7 | 100 | 68777 | 68334 | 0.644 | 69115 | 68461 | 0.311 | 180 | 80 | 55.56 |
| 8 | 410 | 141642 | 140496 | 0.809 | 141942 | 140672 | **0.499** | 186 | 120 | 35.48 |
| 9 | 500 | 154677 | 153487 | 0.769 | 155037 | 153635 | 0.443 | 147 | 90 | 38.78 |
| 10 | 1000 | 220824 | 219512 | 0.594 | 221211 | 219700 | 0.408 | 203 | 98 | 51.72 |
| 均值 |  |  |  | **0.308** |  |  | **0.406** |  |  | **38.24** |

表3 CSO-XSMT和DPSO(X)算法性能对比

Table 3 Comparison between CSO-XSMT and DPSO(X) algorithms

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 电路 | 引数 | 最佳线长 | | |  | 平均线长 | | |  | 标准差 | | |
| DPSO(X) | CSO-XSMT | 优化率(%)  (%) |  | DPSO(X) | CSO-XSMT | 优化率(%)  (%) |  | DPSO(X) | CSO-XSMT | 优化率(%)  (%) |
| 1 | 8 | 16918 | 16918 | 0 |  | 16918 | 16918 | 0 |  | 0 | 0 | 0.00 |
| 2 | 9 | 18041 | 18041 | 0 |  | 18041 | 18041 | 0 |  | 0 | 0 | 0.00 |
| 3 | 10 | 19696 | 19696 | 0 |  | 19696 | 19696 | 0 |  | 0 | 0 | 0.00 |
| 4 | 20 | 32214 | 32193 | 0.065 |  | 32214 | 32210 | 0.012 |  | 10 | 8 | 20.00 |
| 5 | 50 | 48039 | 47952 | 0.181 |  | 48045 | 47955 | 0.187 |  | 74 | 5 | 93.24 |
| 6 | 70 | 56295 | 56271 | 0.043 |  | 56548 | 56288 | 0.460 |  | 138 | 19 | 86.23 |
| 7 | 100 | 68777 | 68334 | 0.644 |  | 69115 | 68370 | 1.078 |  | 180 | 37 | 79.44 |
| 8 | 410 | 141642 | 138989 | 1.873 |  | 141942 | 139060 | 2.030 |  | 186 | 40 | 78.49 |
| 9 | 500 | 154677 | 151345 | 2.154 |  | 155037 | 151378 | 2.360 |  | 147 | 30 | 79.59 |
| 10 | 1000 | 220824 | 214982 | 2.646 |  | 221211 | 215047 | 2.786 |  | 203 | 38 | 81.28 |
| 均值 |  |  |  | **0.761** |  |  |  | **0.891** |  |  |  | **51.83** |

## 4.2 精炼策略的有效性验证

为了验证本文算法精炼策略的有效性，本节对比了采取精炼策略后的CSO-XSMT与DPSO(X)算法获得的最佳线长、平均线长以及标准差，如表3所示。

从表3数据可以看到，增加了精炼策略的CSO-XSMT算法性能得到了进一步的改进，尤其在标准差的指标上优化尤为明显，对比DPSO(X)，标准差平均优化率达到了51.83%。其中对于引脚个数为50的测试用例5，CSO-XSMT标准差优化达到了93.24，针对规模最大的测试用例10，本文算法比起DPSO能够减少2.646%的最佳线长，2.789%的平均线长，以及81.28%的标准差。表3和表4的实验数据证明了精炼策略在减少线长，稳定算法方面的有效性。

## 4.3 与RSMT算法的对比

为验证本文提出的CSO-XSMT算法的有效性，本节对比了CSO-XSMT算法与现有的RSMT工作(DPSO(R)[18]和HTS-PSO(R)[14])的线长优化能力，实验结果如表5所示。表中数据显示，在测试电路1至10上，DPSO(R)，HTS-PSO(R)和本文算法CSO-XSMT得到的线长平均比为1.115：1.090：1。说明本文算法总体上优于这两种RSMT算法。其中，针对测试电路10，DPSO(R)算法和HTS-PSO(R)算法得到的SMT线长分别是本文算法的1.156倍，1.117倍。观察可以看出，线网引脚越多，CSO-XSMT线长优化效果越明显。本节实验证明，本文算法基于X结构构造的SMT能够获得代价最小的布线树。

表4 CSO-XSMT和RSMT算法

线长优化能力对比

Table 4 Comparison between CSO-XSMT and RSMT algorithms

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 电路 | 引脚数 | 平均线长 | | | 归一化值 | | |
| DPSO  (R) | HTS-PSO  (R) | Ours | DPSO  (R) | HTS-PSO  (R) | Ours |
| 1 | 8 | 17931 | 17693 | 16918 | 1.060 | 1.046 | 1.000 |
| 2 | 9 | 20503 | 19816 | 18041 | 1.136 | 1.098 | 1.000 |
| 3 | 10 | 21910 | 21214 | 19696 | 1.112 | 1.077 | 1.000 |
| 4 | 20 | 35173 | 35153 | 32210 | 1.092 | 1.091 | 1.000 |
| 5 | 50 | 52682 | 52087 | 47955 | 1.099 | 1.086 | 1.000 |
| 6 | 70 | 61455 | 61155 | 56288 | 1.092 | 1.086 | 1.000 |
| 7 | 100 | 75997 | 74489 | 68370 | 1.112 | 1.089 | 1.000 |
| 8 | 410 | 159732 | 154210 | 139060 | 1.149 | 1.109 | 1.000 |
| 9 | 500 | 173352 | 166590 | 151378 | 1.145 | 1.100 | 1.000 |
| 10 | 1000 | 248624 | 240231 | 215047 | 1.156 | 1.117 | 1.000 |
| 均值 |  |  |  |  | **1.115** | **1.090** | **1.000** |

# 5 总结

本文提出了一种基于竞争粒子群优化的X结构Steiner最小树的构造算法，以优化VLSI布线问题中的Steiner树线长。首先，算法在传统PSO的基础上做出了巨大的改动，通过配对竞争的方法，使粒子有机会向不同的优秀粒子学习，又通过轮盘赌的机制，进一步扩大了粒子学习对象的范围。这样的学习机制使粒子在迭代后期依旧能够保持种群的多样性，从而克服PSO易陷入局部极值的缺点，其次，结合了遗传算法的变异与交叉操作，考虑Steiner树的特点，实现了对CSO的离散化，从而构建一棵理想的X结构Steiner树。最后，设计了基于公共边的精炼策略，通过尽可能增加公共边的长度，减小最终布线树的线长。实验证明，本文提出的CSO-XSMT算法具有优秀的线长优化能力以及稳定性。在未来的工作中，我们将进一步考虑提高算法的性能以更好地运用于各种VLSI布线问题中。

# 参考文献

1. LIN K W, LIN Y S, LI Y L, et al. A maze routing-based methodology with bounded exploration and path-assessed retracing for constrained multilayer obstacle-avoiding rectilinear Steiner tree construction[J]. ACM Transactions on Design Automation of Electronic Systems, 2018, 23(4): 1-26.
2. LIN S E D, KIM D H. Construction of all rectilinear Steiner minimum trees on the Hanan grid[C]// The 2018 International Symposium on Physical Design. ACM, 2018: 18-25.
3. WUERGES E. 3-Step rectilinear minimum spanning tree construction for obstacle-avoiding component-to-component routing [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020. doi: 10.1109/TCAD.2020.2972534.
4. TeigS. The X architecture: not your father's diagonal wiring. In: ACM International Workshop on System-Level Interconnect Prediction(SLIP02).San Diego,USA,2002.33-37.
5. Thurber A, Xue G . Computing hexagonal steiner trees using PCx for VLSI. In:The 6th IEEE International Conference on Electronic, Circuits & System(ICECS99). Pafos,Spain,1999.381-384.
6. Chiang C,Chiang C S.Octilinear Steiner tree construction. In: The 2002 45th Midwest Symposium on Circuits and Systems(MWSCAS2002).2002,1:603-606.
7. Samanta T,Ghosal P, Rahaman H,et al.A heuristic method for constructing hexagonal Steiner minimal trees for routing in VLSI.In:Proceedings. 2006 IEEE International Symposium on Circuits and Systems(ISCAS2006).Islang of Kos,Korea,2006.1788-1791.
8. ARORA T, MOSES M E. Ant colony optimization for power efficient routing in manhattan and non manhattan VLSI architectures[C]// 2009 IEEE Swarm Intelligence Symposium. IEEE, 2009: 137-144.
9. WU H, XU S, ZHUANG Z, et al. X-architecture Steiner minimal tree construction based on discrete differential evolution[C]// The International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery. Springer, Cham, 2019: 433-442.
10. LIU G, CHEN G, GUO W. DPSO based octagonal Steiner tree algorithm for VLSI routing[C]// 2012 IEEE Fifth International Conference on Advanced Computational Intelligence. IEEE, 2012: 383-387.
11. HUANG X, LIU G, GUO W, et al. Obstacle-avoiding octagonal steiner tree construction based on particle swarm optimization[C]// 2013 Ninth International Conference on Natural Computation. IEEE, 2013: 539-543.
12. HUANG X, LIU G, GUO W, et al. Obstacle-avoiding algorithm in X-architecture based on discrete particle swarm optimization for VLSI design[J]. ACM Transactions on Design Automation of Electronic Systems, 2015, 20(2): 1-28.
13. LIU G, GUO W, NIU Y, et al. A PSO-based timing-driven Octilinear Steiner tree algorithm for VLSI routing considering bend reduction[J]. Soft Computing, 2015, 19(5): 1153-1169.
14. LIU G, CHEN Z, ZHUANG Z, et al. A unified algorithm based on HTS and self-adapting PSO for the construction of octagonal and rectilinear SMT[J]. Soft Computing, 2020, 24(6): 3943–3961.
15. Cheng R , Jin Y . A Competitive Swarm Optimizer for Large Scale Optimization[J]. IEEE Transactions on Cybernetics, 2015, 45(2):191-204.
16. WARME D, WINTER P, ZACHARIASEN M. Geosteiner software for computing Steiner trees[EB/OL]. [2017-11-30]. http://geosteiner.net.
17. LIU G, CHEN G, GUO W. DPSO based octagonal Steiner tree algorithm for VLSI routing[C]// 2012 IEEE Fifth International Conference on Advanced Computational Intelligence. IEEE, 2012: 383-387.
18. LIU G, CHEN G, GUO W, et al. DPSO-based rectilinear Steiner minimal tree construction considering bend reduction[C]// 2011 Seventh International Conference on Natural Computation. IEEE, 2011, 2: 1161-1165.