基于混合离散粒子群优化的 Slew 约束下 X 结构 Steiner 最小树算法

刘耿耿 [1),2)] 黄逸飞 [1),2)] 王鑫 [3),4)] 郭文忠 [1),2)] 陈国龙 [1),2)]

[1)](福州大学 数学与计算机科学学院 福州 350116)

[2)](福建省网络计算与智能信息处理重点实验室 福州 350116)

[3)](天津大学 智能与计算学部 天津 300354)

[4)](天津市认知计算与应用重点实验室 天津 300350)

摘 要 Steiner 最小树是超大规模集成电路中布线阶段的最佳模型，进一步考虑能够有效防止信号失真的电压转换速率(Slew)约束这一个更为贴近实际芯片设计模型和更具线长优化能力的 X 结构，首次提出基于混合离散粒子群优化的 Slew 约束下 X 结构 Steiner 最小树算法. 首先，为了避免频繁的 Slew 约束计算，提出了高效的预处理策略，并且提出一种能够有效考虑 Slew 约束的针对性的惩罚机制. 其次，为了能够有效求解该离散问题，基于遗传算子重新设计了粒子群优化算法的离散更新机制，并提出一种更适合遗传算子的引脚对编码方式. 然后，为了进一步优化布线树的长度，提出一种有效的精炼策略. 最终，提出一种混合修正策略以完全满足 Slew 约束. 实验表明，所提算法可完全满足电压转换速率约束并取得同类工作中最佳的布线结果.

关键词 粒子群优化; Steiner 树; 电压转换速率约束; X 结构; 超大规模集成电路

中图法分类号 TP18

Hybrid Discrete Particle Swarm Optimization Algorithm for X-Architecture Steiner Minimal Tree Construction with Slew Constraints

LIU Geng-Geng[1),2)] HUANG Yi-Fei[1),2)] WANG Xin[3),4)] GUO Wen-Zhong[1),2)] CHEN Guo-Long[1),2)]

[1)](College of Mathematics and Computer Sciences, Fuzhou 350116)

[2)](Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou 350116)

[3)](College of Intelligence and Computing, Tianjin 300354)

[4)](Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin 300350)

Abstract Steiner minimal tree is the best model of routing stage in modern very large-scale integration (VLSI) chips, and is often used for pre-routing, wirelength optimization, and congestion estimation. Therefore, it is of great significance to construct a high-performance Steiner minimal tree algorithm. However, with the emergence of obstacles such as IP blocks and the continuous improvement of circuit density, obstacles have become a factor that cannot be ignored in the Steiner minimal tree construction problem. Considering that in modern multi-layer routing, obstacles often only occupy the device layer and the lower metal layer. Therefore, the routing on the top of the obstacles is possible, and it can make full use of routing resources and further

收稿日期：2020-08-14；在线发布日期：2020-12-21.本课题得到国家自然科学基金(61877010, 11501114)、国家重点基础研究发展计划(2011CB808000)、计算机体系结构国家重点实验室开放课题(CARCHB202014)、福建省自然科学基金(2019J01243)资助. 刘耿耿，博士，副教授，博士生导师，CCF高级会员（75198S），主要研究方向为EDA算法研究、计算智能及其应用. 第1作者手机号码: 13950363682, E-mail: liugenggeng@fzu.edu.cn. 黄逸飞，硕士研究生，主要研究方向为EDA算法研究. E-mail: fzu_hyf@126.com. 王鑫，博士，教授，博士生导师, CCF杰出会员（14972D），主要研究领域为大规模知识处理、计算智能及其应用. E-mail: wangx@tju.edu.cn. 郭文忠（**通信作者**），博士，教授，博士生导师, CCF高级会员（10092S）,主要研究方向为EDA算法研究、计算智能及其应用. E-mail: guowenzhong@fzu.edu.cn. 陈国龙，博士，教授，博士生导师, CCF高级会员（07567S）,主要研究方向为EDA算法研究、计算智能及其应用. E-mail: cgl@fzu.edu.cn.

optimize the wirelength. Since the Steiner minimal tree construction problem is an NP-hard problem, the particle swarm optimization algorithm has a good application prospect in solving NP-hard problems. Therefore, on the basis of the particle swarm optimization algorithm, further considering the model of slew constraints that can effectively prevent signal distortion, and the X-architecture with better wirelength optimization, this paper is the first work to propose an X-architecture Steiner minimal tree algorithm with slew constraints based on hybrid discrete particle swarm optimization. Firstly, in order to avoid frequent slew calculation and judgment between routing and obstacles, an efficient preprocessing strategy is proposed. In this preprocessing strategy, the information between all possible routing of any two pins and all obstacles is calculated in advance, and a suitable lookup table is generated based on this information for subsequent queries. Secondly, in order to effectively solve the discrete problem of Steiner minimal tree construction, an effective discrete update operation formula of particle swarm optimization algorithm is redesigned, which is based on the mutation operator and the crossover operator. For discrete particle swarm optimization, a pin-pair coding method that is more suitable for this discrete particle swarm optimization and a targeted penalty mechanism that can effectively consider slew constraints are proposed. In addition, so as to speed up the search efficiency of the particle swarm optimization algorithm, the Prim method is used to construct a minimum spanning tree under a given pin set, and initialize the population. Thirdly, in order to further optimize wirelength of the routing tree, an effective local optimal refining strategy is proposed. In this step, the Steiner tree is divided into multiple subtrees with roots as pins and a depth of 2. By traversing all possible routing structures in the subtree, the routing structure with the highest degree of routing resource sharing is selected to replace the original routing structure. Thereby the goal of optimizing the wirelength is achieved. Finally, in order to make the Steiner minimal tree fully satisfy the slew constraints, a hybrid correction strategy is proposed. In this part, by estimating the cost of adjusting the routing, this paper selects the overall adjustment strategy or the adjustment strategy along the barrier to correct the routing that violates the slew constraints. Experiments show that the proposed algorithm can achieved the best routing result and fully satisfy the slew constraints.

Key words particle swarm optimization; Steiner tree; slew constraints; X-architecture; VLSI

1 引言

Steiner 最小树是超大规模集成电路(Very Large-Scale Integration, VLSI)中物理设计的基础结构, 是布线问题的最佳模型[1], 对 VLSI 物理设计中最重要步骤之一的线网布线阶段有重要的指导意义. 随着集成电路的设计规模不断扩大, 芯片中障碍的数量规模也随之扩大, 芯片的密度不断增加, 给线网布线带来巨大挑战[2]. 而在 Steiner 树构造问题中, 为了考虑障碍等问题, Steiner 树的布线代价随之急剧增大. 由于制造工艺的不足, 以往对布线问题的研究大多集中在直角结构上, 但随着制造工艺的进步(从超深亚微米进入纳米阶段), 电子系统设计正从板上系统(System-on-a-Board, SoB), 向系统级芯片(System-on-a-Chip, SoC)方向发展. SoC 设计概念对电路性能提出更高要求[3]. 由于直角结构的布线方向有限, 不能充分利用布线区域, 在减少布线线长等优化目标上有一定的局限性. 为了进一步优化布线, 有不少学者将目光转向非直角结构. 洪先龙等[3]指出非直角结构的研究将会成为布线方向的热点. X 结构作为非直角结构的代表, 有更多的布线方向, 能够更充分利用布线区域资源, 在优化线长目标上有独特的优势. 文献[4]指出在布线问题上, X 结构相对于直角结构在布线线长及通孔数量上取得显著的优化, 且在由互连引起的延迟迅速增加以及纳米级通孔等制造挑战上, X 结构布线带来的布线长度和通孔的减少使得芯片性能提高, 功耗降低以及芯片制造成本减少. 文中指出, 虽然由于光刻方面的因素, 设计中不允许任意角度的布线, 但几乎所有当前的制造工艺都完全支持 X 结构. 在文献[5]中, 将 X 结构运用在线网布线的多层总体布线阶段, 在多

层总体布线中最重要的两个优化目标上也均取得最佳.

在实际的 VLSI 布线过程中, 障碍没有完全阻断布线. 布线不严格禁止穿过障碍情况称作布线资源松弛. 相对于绕障 Steiner 最小树, 在考虑布线资源松弛的情况下, 可以减少线长和时延、降低功耗和拥挤度. 然而, 信号经过障碍会发生衰减, 为了避免信号的失真, 在障碍内部的 Steiner 树的连通分量需要满足电压转换速率(Slew)约束. 本文将该问题构建为电压转换速率约束下 X 结构 Steiner 最小树(X-architecture Steiner Minimal Tree considering Slew Constraints, XSMT-SC)问题. 目前对于 Slew 约束下 Steiner 最小树问题的研究主要集中在直角结构, 尚未涉及到 X 结构.

Steiner 最小树的构造问题被证明是 NP 难问题[6]. 而粒子群优化(Particle Swarm Optimization, PSO)算法是一种基于种群的随机优化算法, 由 Eberhart 和 Kennedy 于 1995 年提出[7], 具有搜索速度快、效率高等优点, 以 PSO 为代表的群智能算法对解决 NP 难问题展现出良好的应用前景[8-16]. 在 PSO 算法中, 种群的每个粒子都是优化问题的潜在解, 每个粒子都有决定自身飞行的方向和速度以及评定自身位置优劣的适应值. 粒子通过两种“最优粒子”更新自己, 一种是当前迭代中自身搜索到的个体最优粒子, 另一种是当前迭代中种群搜索到的全局最优粒子. PSO 算法经过合适的迭代次数得到优化问题的高质量解, 并在 VLSI 领域得到很好的应用[8,12-17].

鉴于 (1) Slew 约束下 Steiner 最小树问题侧重于直角结构, 尚未考虑基于 X 结构, (2) PSO 在求解诸如 Steiner 最小树等 NP 难问题展现出良好的应用前景, 本文第一次提出基于混合离散粒子群优化的 Slew 约束下 X 结构 Steiner 最小树的构造算法(Hybrid Discrete Particle Swarm Optimization for X-Architecture Steiner Minimal Tree Construction Algorithm with Slew Constraints, HDPSO-XSMT-SC). 本文主要贡献如下：

(1) 首次提出考虑 Slew 约束的 X 结构 Steiner 最小树构造算法. 在满足 Slew 约束前提下, 与同类算法相比, 取得最佳的布线结果.

(2) 为了能够有效求解本文的离散问题, 基于遗传算子重新设计了粒子群优化算法的离散更新操作, 同时也提出一种更适合这种离散粒子群优化的引脚对编码方式.

(3) 提出有效的预处理策略, 有力地减少 Slew 约束的计算次数. 同时设计一种有效的惩罚机制, 使得 PSO 算法在同样的迭代次数下, 得到一棵高质量的 Steiner 树.

(4) 提出一种局部最优策略. 通过该策略, 使得 Steiner 树的局部结构达到最优, 从而更为接近全局线长优化的效果. 同时提出一种混合修正策略, 通过两种修正策略, 将 Steiner 树中违反约束的部分进行调整, 使 Steiner 树完全满足 Slew 约束.

本文第 2 节介绍相关工作. 第 3 节介绍 XSMT-SC 问题及相关定义. 第 4 节给出本文提出的 HDPSO-XSMT-SC 算法的实现细节, 并分析 HDPSO-XSMT-SC 算法的相关性质及定理. 第 5 节中给出本文相关策略有效性的验证及实验结果比较分析. 最后是结束语.

2 相关工作

对于 Steiner 树构造问题, 在忽略障碍情况下, 研究人员提出许多有效的方法. Chu 和 Wong 提出一种称作 FLUTE 的直角结构 Steiner 最小树算法[18]. FLUTE 基于预先计算的查找表, 在引脚较少的线网中能够快速又准确地得到一个最优解, 而在引脚较多的线网中, 通过使用线网分解技术, 从而得到一个较好的解. Lin 和 Kim 提出一种方法可以在较短时间内建立一个在哈南网格上构造所有直角 Steiner 树的数据库[19], 并运用于时间驱动的直角 Steiner 树构造及考虑拥塞的全局布线中. 文献[20]通过分治法和深度优先搜索获取最小生成树的方法进一步改进 FLUTE 模型, 在线长相近情况下, 显著减少了内存开销及计算时间. 文献[21]基于伪布尔满足性(Pseudo-Boolean Satisability, PB-SAT)提出一种直角结构 Steiner 布线方法, 并利用区域划分及聚类方法来处理大规模的线网, 相对于使用 FLUTE 的同类算法, 大大提高运行速度. 文献[22]基于差分进化(Differential Evolution, DE)算法思想提出一种 X 结构 Steiner 最小树算法. 该算法结合了变异算子和

交叉算子，可以解决离散的 Steiner 树构造问题，从而得到高质量的解. 文献[17]将梯度下降用于粒子群算法的局部搜索，在直角结构下 Steiner 树的构造问题上的布线代价取得优化. 而在考虑障碍的情况下，Huang 和 Young 提出一种几何方法来解决复杂障碍的绕障直角 Steiner 最小树问题[23]. 该算法能够同时解决凹凸障碍问题并得到较优解. 文献[24]利用图形处理单元实现了一种并行算法，这是首次提出一种构建绕障 Steiner 树的并行方法. 由于直角结构自身的局限性，在线长目标上不能得到进一步的改进. 因此在 X 结构下，Coulston 提出了一种精确算法[25]，通过引入多种剪枝技术构造绕障 Steiner 树，相对于直角结构，在线长目标上得到较好的优化，但付出很大的时间代价. Huang 等提出一种有效的基于粒子群优化算法的绕障 Steiner 最小树算法[26]. 该算法成功地结合了遗传算子，将解决连续问题的 PSO 算法运用于离散的绕障 Steiner 树构建问题，并提出一系列的有效启发式策略. 在运行时间和线长等方面均得到优化，并扩展到多层模型[12,27]. 而在多层模型下，文献[28]将引脚间相连的 Steiner 树构造问题转换为区域相连，大大降低时间复杂度，并取得更好的求解质量. 文献[29]基于迷宫布线及最小生成树算法，首次提出一种多维环境下的绕障 Steiner 算法，并分别在直角结构与 X 结构下得到验证.

为了充分利用障碍内部的布线资源，在考虑布线松弛的情况下[30-35]，文献[30]提出解决考虑布线资源松弛的 Steiner 树构建问题的简化模型，称为限制长度的 Steiner 最小树(Length-Restricted Steiner Minimum Tree, LRSMT). 在该模型中，将在障碍内部的布线长度限制在一个门限值下，并设计了相应的 Steiner 树构造算法. 文献[31]中，Held 等在 LRSMT 模型下，基于构建范围可视图，提出了一个最坏情况下运行时间为 $O((k\log k)^2)$的 2-近似算法，与绕障 Steiner 树相比，大大优化了线长目标. 而在文献[32, 33]中引入了一种更加精确和更为接近实际芯片设计的模型，称为带电压转换速率约束的 Steiner 最小树(Steiner Minimal Tree with Slew Constraint, SMT-SC). SMT-SC 模型中使用 PERI 模型[34]计算具体的电压转换速率值，进而精确满足约束. 文献[32]提出一种启发式算法，在 SMT-SC 模型下，引入三种降低电压转换速率的操作，并在不同的松弛条件下均取得较好的优化. Zhang 等通过修改最短路启发式(Shortest Path Heuristic, SPH)算法[35]，设计了一种逐步生长的启发式算法，在不同模型下均取得较好的优化. LRSMT 模型虽然提高了求解效率，但容易违反实际约束或绕行，对后续布线工作增加难度，SMT_SC 模型更多地考虑障碍内部的 Steiner 树连通分量的拓扑，求得的解方案也更加满足实际芯片设计约束，对后续布线工作有更好的帮助.

以上工作[17-26]均未考虑布线资源松弛从而使得消耗过多的布线长度，而文献[32, 33, 35]虽然进一步考虑更为接近实际芯片设计的布线资源松弛约束即考虑 Slew 约束，但均集中在直角结构，尚未涉及到非曼哈顿结构，即本文研究的 Slew 约束下 X 结构 Steiner 最小树问题.

3 相关定义与问题模型

3.1 相关定义

定义 1 引脚. 引脚是布线层上待连接的端点，引脚不能位于障碍内，但能位于障碍的边上.

定义 2 障碍. 本文中障碍形状为矩形且障碍之间不能重叠，同时由于缓冲器不能放置于两障碍的公共边，所以规定布线不经过障碍公共边.

定义 3 内部树与外部树. Steiner 树被障碍的边分割为两类子树：内部树与外部树. 内部树为某障碍内部连通分量，外部树为所有障碍外的连通分量.

定义 4 驱动节点与接收节点. 内部树与障碍的交点称为叶子节点，叶子节点中离信号源最近的称为驱动节点，其余的叶子节点称为接收节点.

定义 5 曼哈顿距离. 两点之间的垂直距离与水平距离的和称作两点间曼哈顿距离.

定义 6 片段. 在引脚间布线中，障碍内部的线段称为片段. 一条布线可能有零片段、单片段或多片段.

定义 7 伪 Steiner 点. 除了引脚外的端点称为伪 Steiner 点.

3. 2 Slew 约束相关知识

本文采用 PERI 模型[36]和 Elmore 模型[37,38]计算 Slew. 由于缓冲器不能放置于障碍内部，所以在 PERI 模型中，在内部树的驱动节点前放置缓冲器，而在每个接收节点后放置缓冲器. 如图 1 所示，在引脚 S, P, Q 构成叶节点为 3 的内部树中，驱动节点 v_{in} 前放置缓冲器 b_{in}，而在接收节点 v_{out0} 和 v_{out1} 后分别放置缓冲器 b_{out0} 和 b_{out1}. 公式(1)为具体的电压转换速率公式.

$$s(v_{out})=\sqrt{s(v_{in})^2+s_{step}(v_{in},v_{out})^2} \tag{1}$$

其中 $s(v_{in})$是 b_{in} 的输出电压转换速率，文献[39]给出一种简化的计算公式，具体如公式(2)所示.

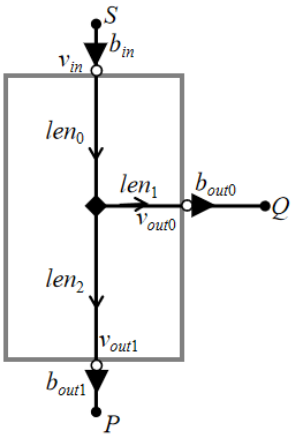


图 1 节点数为 3 的内部树

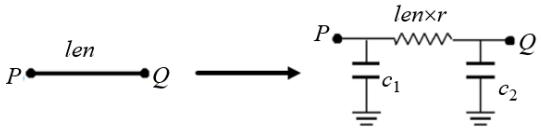
$$s(v_{in})=K_{b_{in}}+R_{b_{in}}\times C(v_{in}) \tag{2}$$

其中 $K_{b_{in}}$ 为 b_{in} 的固有电压转换速率，$R_{b_{in}}$ 为 b_{in} 电压转换速率阻抗，$C(v_{in})$为节点 v_{in} 的后继电容. 而公式(1)中的 $s_{step}(v_{in},v_{out})$ 是驱动节点 v_{in} 与接收节点 v_{out} 之间的步进电压转换速率，通过 Elmore 模型计算得出，具体如公式(3)所示.

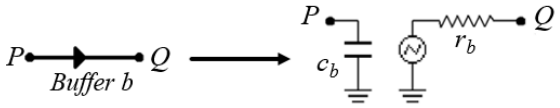
$$s_{step}(v_{in},v_{out})=\alpha\times D(v_{in},v_{out}) \tag{3}$$

其中 α 大小为 ln9，$D_p(v_{in},v_{out})$表示驱动节点 v_{in} 与接收节点 v_{out} 的 Elmore 时延.

在 Elmore 时延模型中，将每个内部树都建模成电阻电容电路. 如图 2(a)所示，在该模型中将连线建模成一半电容 c_1 处于上游节点，一半电容 c_2 处于下游节点的电阻电容模型. 而图 2(b)中展示了，将缓冲器建模成输入电容 c_b 与上游节点连接，输出电阻 r_b 与下游节点连接的电阻电容模型. 公式(4)给出自身电容的计算公式(即 c_1 与 c_2 的计算公式)，由公式可知与线长成正相关.



(a) 互连线电容电阻电路建模



(b) 缓冲器电容电阻电路建模

图 2 电阻电容组合模型

$$c_{len}=\frac{1}{2}\times len\times \tag{4}$$

文献[36]中给出具体的计算 Elmore 时延的方法，公式如下.

$$D_p(v_{in},v_{out})=\sum_{e\in path(v_{in},v_{out})}R_e\times\left(c_{len_e}+C(v_r)\right) \tag{5}$$

其中边 e 为驱动节点 v_{in} 到接收节点 v_{out} 的路径上由内部树节点构成的边集的元素，v_r 为路径上的当前节点，$C(v_r)$为节点 v_r 的后继电容. 以图 3 为例，公式(6)为节点 v_{in} 到 v_{out1} 的 Elmore 时延的具体计算公式.

$$\begin{aligned}D_p(v_{in},v_{out1})=&r_b\times(\sum_{i=0}^{2}len_i\times c+\sum_{j=0}^{1}c_{bout\,j})+len_0\times\\&r\times(\sum_{i=1}^{2}len_i\times c+\frac{(len_0\times c)}{2}+\sum_{j=0}^{1}c_{bout\,j})\\&+len_2\times r\times(\frac{(len_2\times c)}{2}+c_{bout1})\end{aligned} \tag{6}$$

其中 $(\sum_{i=0}^{2}len_i\times c+\sum_{j=0}^{1}c_{bout\,j})$ 是驱动节点 v_{in} 的后继电容，r_b 是缓冲器 b_{in} 的输出电阻，$(len_0\times r)$为边 len_0 的电阻，$(len_0\times c)/2$ 为布线边的自身电容，括号中其余项式为下游节点的后继电容，同条路径上的边 len_2 的计算方式同边 len_0.

3. 3 问题模型

在考虑 Slew 约束的情况下，$P=\{P_1, P_2, \dots, P_n\}$为线网上需要连接的一组引脚，$O=\{O_1, O_2, \dots, O_k\}$为线网上的一组矩形障碍. 每个引脚 P_i 对应一个二维坐标(x_i, y_i)，分别表示引脚的横坐标和纵坐标. 每个障碍 O_j 对应两个二维坐标 (x_{j1}, y_{j1}), (x_{j2}, y_{j2})，分别表示障碍对角线两端点的横坐标和纵坐标，对于障碍集合 O 中存

在公共边的障碍则合并成更大的障碍. 本文需要构建一棵连接引脚集合 P 中所有引脚的 Steiner 最小树, 并满足以下条件：(1)布线树的每条边都需要满足 X 结构的布线连接方式; (2)障碍内部的连通分量满足电压转换速率约束.

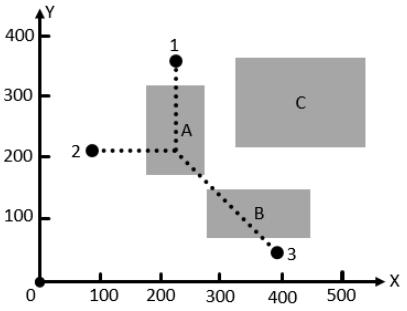
表 1 为引脚集合 P ={1, 2, 3}的坐标, 表 2 为障碍集合 O ={A, B, C}的坐标, 图 3(a)表示连接 P 中全部引脚所构成一棵 Steiner 树. 假设图 3(a)构成的 Steiner 树在障碍 A 中内部树有部分节点不满足电压转换速率约束, 则需要对其进行调整.图 3(b)是进行调整后符合电压转换速率约束的 Steiner 树.

表 1 线网中引脚的二维坐标

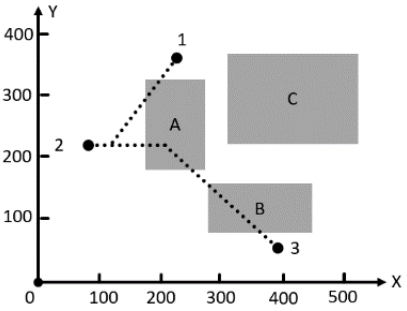
编号	1	2	3
X 坐标	230	98	385
Y 坐标	360	205	50

表 2 线网中障碍的坐标信息

编号	A	B	C
X_1	180	280	315
Y_1	180	80	215
X_2	270	450	540
Y_2	315	150	370



(a) 引脚数为 3 的 Steiner 树



(b) 调整后满足约束的 Steiner 树

图 3 线网布线

4 HDPSO-XSMT-SC

对于 Slew 约束下的 Steiner 最小树这一 NP 难问题的研究都集中在直角结构, 而对 X 结构布线的研究大多只考虑完全绕障的情况. 为此, 本文研究在更贴近实际芯片设计的 Slew 约束模型下, 基于在求解 NP 难问题中展现出良好应用前景的 PSO, 提出 HDPSO-XSMT-SC 算法. 本节从引脚对编码方式与初始化、预处理策略、PSO 搜寻、局部最优策略及混合修正策略 5 个方面分别介绍 HDPSO-XSMT-SC 算法的设计细节.

4.1 引脚对编码方式与初始化

本文设计一种更适合进化算法的引脚对编码方式. 将 Steiner 树用引脚对方式进行编码, 将树的每条边用三位数进行编码, 前两位数代表了连接的两个引脚序号, 第三位表示了两引脚间的布线选择, 例如(1, 3, 2)表示了引脚 1 和引脚 3 用 2 选择布线方式进行连接.

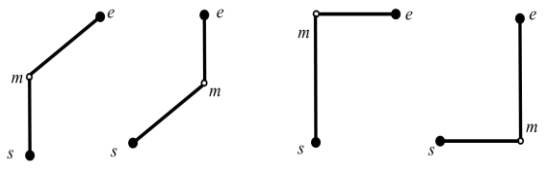
定义 8 布线选择. 在 X 结构中, 有 4 种布线选择, 分别为 0 选择, 1 选择, 2 选择, 3 选择.

定义 9 0 选择. 在图 4(a)中, 引脚 s, e 是待连接的两引脚(假定引脚 s 在引脚 e 的左下侧), 0 选择布线方式先从引脚 s 引垂直边到伪 Steiner 点 m, 再从点 m 引 45°边连接到引脚 e.

定义 10 1 选择. 在图 4(b)中, 1 选择布线方式先从引脚 s 引 45°边到伪 Steiner 点 m, 再从点 m 引垂直边连接到引脚 e.

定义 11 2 选择. 在图 4(c)中, 2 选择布线方式先从引脚 s 引垂直边到伪 Steiner 点 m, 再从点 m 引水平边连接到引脚 e.

定义 12 3 选择. 在图 4(d)中, 3 选择布线方式先从引脚 s 引水平边到伪 Steiner 点 m, 再从点 m 引垂直边连接到引脚 e.



(a) 0 选择 (b) 1 选择 (c) 2 选择 (d) 3 选择

图 4 引脚对的 4 种布线选择方式

假设 Steiner 树的点集规模大小为 N, 则树的边集规模大小为(N-1). 再用 1 位数评估 Steiner 树的质量即适应度函数值, 所以 Steiner 树可以编码为 3×(N-1)+1 位数. 以图 3(a)为例, 数字串(2, 1, 3, 1, 3, 0, 506.2)表示一棵 N=3 的 X

结构 Steiner 树，其中前六位数字表示边的具体布线方式, 506.2 表示这棵树的适应值.

由于最小生成树(Minimum Spanning Tree, MST)能够较快构造并转换为 Steiner 树，所以生成树算法被广泛用于 Steiner 最小树的初始化. 本文用 Prim 算法，并以引脚间的曼哈顿距离为权重生成最小树，从而初始化 PSO 的种群，同时设定好 PSO 算法的规模、迭代次数与各种相关参数. 具体步骤如下：

(1) 为引脚和障碍分别设置连续且唯一的编码.

(2) 计算任意两引脚所构成引脚对的曼哈顿距离并设为 Prim 算法的权重，再根据计算出的权重构建基于不同起点的多种最小生成树.

(3) 使用构建出来的 MST 集合作为 PSO 算法中所有粒子的初始位置，并初始化粒子群的历史最佳位置和种群的最佳位置.

(4) 为 PSO 设定适当的权重、加速因子和最大迭代次数.

4. 2 预处理策略

Slew 约束模型下 Steiner 最小树的构造需要频繁计算 Slew，而 Slew 的计算与线长紧密相关. 由于当前工艺下芯片的密度急剧增加，问题规模比以往任何时候都大，预先记录下引脚与障碍之间的布线信息，对 PSO 搜寻过程以及之后步骤的执行节约大量判断与计算时间.

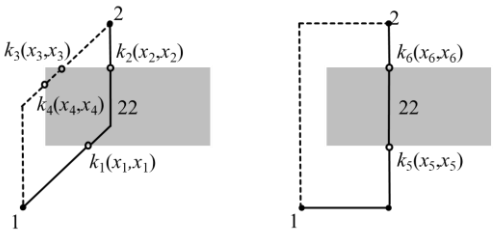
为此，本文设计两张查找表：经障判断表与障碍信息记录表. 如图 5(a)所示，引脚 1, 2 之间，0 选择布线方式与障碍相交于 k_3, k_4, 1 选择与障碍相交于 k_1, k_2. 在图 5(b)中，2 选择不经过障碍，3 选择与障碍相交于 k_5, k_6, 交点用二维坐标表示. 在表 3 中，经障情况一列表示布线经过障碍的个数，表 4 记录了经过的各个障碍的编号与交点的坐标.

4. 3 PSO 搜寻

PSO 算法是基于种群的一种优化算法，在粒子群中的每一个粒子都是问题的一个可能解，通过粒子个体的简单行为，及群体内的信息交互来实现问题的优化. 原先的 PSO 是针对连续性问题而提出的，而本文所求解的 Slew 约束下 X 结构 Steiner 最小树问题是离散问题，需要将 PSO 操作算子离散化. 本文算法在课题组先前求解相关离散问题的工作经验上[8,12-16,22]，结合并查集思想，引入变异操作和交叉操作将 PSO 操作离散化以搜索全局最优粒子，从而能够有效求解 Slew 约束下 X 结构 Steiner 最小树这一离散问题. 公式(7)为粒子更新公式.

$$P_i^t = F_2(F_1(V(P_i^{t-1}, w), c_1), c_2) \tag{7}$$

其中 w 是惯性权重，代表粒子进行变异操作的概率，c_1，c_2 是加速因子，代表粒子进行交叉操作的概率，F_1 为个体经验感知，F_2 为全局经验感知，V 为粒子的惯性部分.



(a) 0, 1 选择　　(b) 2, 3 选择

图 5　X 结构下的引脚 1, 2 的布线

表 3 经障判断表

引脚编号	引脚编号	布线选择	经障情况
1	2	0	1
1	2	1	1
1	2	2	0
1	2	3	1

表 4 障碍信息记录表

引脚编号	引脚编号	布线选择	障碍编号	交点 1	交点 2
1	2	0	22	k_3	k_4
1	2	1	22	k_1	k_2
1	2	3	22	k_5	k_6

(1) 粒子的惯性部分更新公式如下：

$$S_i^t = V(P_i^{t-1}, \omega) = \begin{cases} M(P_i^{t-1}), r_1 < \omega \\ P_i^{t-1}, else \end{cases} \tag{8}$$

其中 M 为变异操作，r_1 为均匀生成的 0~1 的随机概数.

(2) 粒子个体经验感知如下：

$$H_i^t = F_1(S_i^{t-1}, c_1) = \begin{cases} C(S_i^{t-1}), r_2 < c_1 \\ S_i^{t-1}, else \end{cases} \tag{9}$$

其中 C 为交叉操作，r_2 为均匀生成的 0~1 的随机数.

(3) 粒子全局经验感知如下：

$$P_i^t = F_2(H_i^{t-1}, c_2) = \begin{cases} C(H_i^{t-1}), r_3 < c_2 \\ H_i^{t-1}, else \end{cases} \quad (10)$$

其中 C 为交叉操作，r_3 为均匀生成的 0~1 的随机数.

4.3.1 离散 PSO 的操作算子

4. 3. 1. 1 变异操作算子

为了增强算法的搜索质量，避免 PSO 陷入局部最优，本文算法引入变异算子以代替粒子惯性部分的更新. 变异算子的具体操作是在 n 个节点的 Steiner 树中，随机从 n-1 条边中选择一条边删除，将 Steiner 树分割为两棵子树，并分别随机从两棵子树中选择出两个节点进行连接，使所构成 Steiner 树保持连接且无环边. 步骤如下：

(1) 从 Steiner 树中随机选择一条边，然后将其删除.

(2) 扫描 Steiner 树剩余的边，并使用并查集将所有点划分为两个连通分量.

(3) 从两个连通分量中分别随机选择两个点 P 和 Q (用并查集检查点 P 和点 Q 是否在同一集合中).

(4) 连接点 P 与点 Q 形成变异后的新边.

如图 6 所示，在一个节点为 7 的 Steiner 树中随机选择边 PQ 删除，并从两棵中选择点 P', Q 进行连接，构成新的 Steiner 树. 为避免环的出现，本文算法利用并查集记录点的情况.

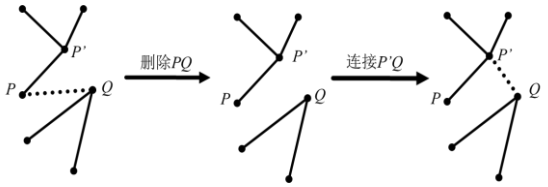


图 6 变异操作

4. 3. 1. 2 交叉操作算子

鉴于学习 PSO 粒子的历史最佳位置和种群的最佳位置，以保留 Steiner 树的最佳子结构的目的，本文算法引入交叉算子. 交叉算子具体操作是从两棵点集相同但边集不同的 Steiner 树 T_1, T_2 中，将所有的边划分成两个边集. 对于树 T_1, T_2 中，相同的边划分进集合 S_1，其余的边划分进 S_2. 以 S_1 中的边为新树的基本架构，随机从 S_2 中选择边，构成新的连接边且无环的 Steiner 树，用并查集记录点的连接情况. 步骤如下：

(1) 设定边的两个端点中较小的编号作为起点，较大的编号作为终点. 将两棵 Steiner 树中所有边分别按起点编号大小排序，如果起点编号相同，则按终点编号大小排序.

(2) 比较两个 Steiner 树，将具有相同起点、终点的边划分为 S_1，其余边划分为 S_2.

(3) 从 S_2 中随机选择边加入 S_1，直到 S_1 成为完整的 Steiner 树，其中用并查集检查点的引入是否会产生环边.

如图 7 所示，T_1={a_1, a_2, a_3, a_4, a_5, a_6}, T_2={a_2, a_3, a_5, a_6, a_7, a_8}, 集合 S_1={a_2, a_3, a_5, a_6 }, 集合 S_2={a_1, a_4, a_7, a_8}，对于 T_1, T_2 相同的边 a_2, a_3, a_5, a_6 进行保留，并随机选择边 a_1, a_8 构成新的 Steiner 树.

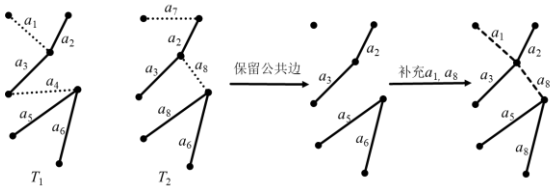


图 7 交叉操作

4. 3. 2 适应值函数与惩罚函数

在 XSMT-SC 问题中，最重要的优化目标为线长，所以本文算法的适应值与 Steiner 树线长紧密相关. 本文采用引脚对方式为 Steiner 树编码，公式(11)~(13)为引脚 P, Q 在 X 结构下的距离公式, sp 表示引脚 P, Q 之间的布线选择方式.

$$XDis_{01}(P,Q) = \max(|x_P - x_Q|, |y_P - y_Q|) + (\sqrt{2}-1) \times \min(|x_P - x_Q|, |y_P - y_Q|) \quad (11)$$

$$XDis_{23}(P,Q) = |x_p - x_Q| + |y_p - y_Q| \quad (12)$$

$$XDis_{sp}(P,Q) = \begin{cases} XDis_{01}(P,Q), sp \in \{0,1\} \\ XDis_{23}(P,Q), sp \in \{2,3\} \end{cases} \quad (13)$$

其中公式(11)是当布线边选择方式是 0 和 1 时引脚对之间的距离计算公式，公式(12)是当布线边选择方式是 2 和 3 时引脚对之间的距离计算公式.

由于引脚对之间布线存在共用边的情况，如图 8 所示，引脚对(A, C)与引脚对(A, B)的布

线共用线段 *AD*, 所以 Steiner 树的线长代价计算如公式(14)所示.

$$WL(T)=\sum_{(P,Q)\in T} XDis_{sp}(P,Q)-\sum_{(A,D)\in T} XDis_{sp}(A,D) \quad (14)$$

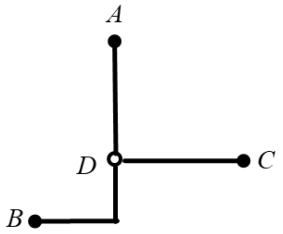


图 8 引脚数为 3 的布线图

在 Steiner 树中, 对于穿过障碍但不满足约束的布线需要花费额外的布线代价进行修复, 所以需要对该类型布线进行一定惩罚以减少该类型布线的产生, 其惩罚函数为公式(15), 其中 k 为单个片段的惩罚因子, 公式(16)给出具体公式.

$$P(T)=\sum_{(P,Q)\in T}\sum_{k\in(P,Q)} k \quad (15)$$

$$k=\begin{cases} dx_i+dy_i-len_i & \text{布线为}0^\circ\text{或}90^\circ\text{边} \\ \left(\sqrt{2}-1\right)\times len_i & \text{布线为}45^\circ\text{或}135^\circ\text{边} \end{cases} \quad (16)$$

其中引脚对(P , Q)是 Steiner 树中不满足约束的布线, len_i 表示布线在障碍 i 中的长度, dx_i 和 dy_i 分别表示障碍 i 的长与宽. 由公式(15), (16)可知, 惩罚函数的大小与布线中违反约束片段数量以及片段穿过障碍的大小均相关.

综上所述, 粒子的适应值大小为线长代价与惩罚函数两部分组成. 其具体公式如下.

$$Fitness(T)=WL(T)+P(T) \quad (17)$$

4. 3. 3 参数的设定

性质 1. 不同惯性权重值的设定会影响 HDPSO-XSMT-SC 算法的全局搜索能力和局部搜索能力, 较大的惯性权重具有较好的全局搜索能力, 较小的惯性权重具有较好的局部搜索能力.

文献[40]指出较大的惯性权重 w 有较好的全局搜索能力, 较小的惯性权重 w 有较好的局部搜索能力. 从 HDPSO-XSMT-SC 算法的速度更新公式可看出, 较大的惯性权重更容易进行变异操作, 这意味着较大的惯性权重具有更强的搜索能力, 不容易陷入局部最优, 而较小的惯性权重则会使得粒子对局部区域进行搜索, 使得算法较快收敛.

性质 2. 过大的加速因子 c_1 会导致在局部空间过度的搜索, 而过大的加速因子 c_2 会导致 HDPSO-XSMT-SC 算法的搜索过早的收敛, 加速因子 c_1, c_2 用于粒子间的信息交互, 主要用于对粒子历史最佳位置与种群最佳位置的学习. 文献[41]提出一种策略, 前期选择较大的 c_1 和较小的 c_2, 以保证在局部能够充分搜索, 后期则选择较大的 c_2 和较小的 c_1, 加快收敛速度. 文献[16]提出一种线性调整参数策略, 使得 PSO 更好地应用于解决离散问题.

综上所述, 本文提出的 HDPSO-XSMT-SC 算法采用一种动态的参数调整策略：w 随着迭代次数的增加从 0.95 到 0.4 线性递减, c_1 随着迭代次数的增加从 0.9 到 0.15 线性递减, c_2 随着迭代次数的增加从 0.1 到 0.85 线性递增. 公式(18)~(20)分别为 w, c_1, c_2 所对应的更新公式.

$$w_{new}=w_{start}-\frac{w_{start}-w_{end}}{evals}\times eval \quad (18)$$

$$c_{1_{new}}=c_{1_{start}}-\frac{c_{1_{start}}-c_{1_{end}}}{evals}\times eval \quad (19)$$

$$c_{2_{new}}=1-c_{1_{new}} \quad (20)$$

其中 *evals* 为最大迭代次数, *eval* 为当前迭代次数.

4. 4 局部最优策略

本文算法的 PSO 搜寻阶段的目标是为了得到一棵适应值小的 Steiner 树, 但不能完全保证所得到的 Steiner 树对应的布线方案达到最佳. 为了进一步提高布线间的共享程度, 从而达到线长代价减少的目的, 本文提出一种有效的局部最优策略.

局部最优策略是在经过 PSO 搜寻阶段从而确定了 Steiner 树有效拓扑的基础上, 将 Steiner 树的每个节点作为树根, 与相邻引脚和引脚对应邻边构建成一个深度为 2 的局部 Steiner 树进行优化, 通过对 Steiner 树局部结构达到最优以期得到全局优化, 最终达到进一步减小适应值的目的. 假定局部 Steiner 树有 m 个节点, 则该树有 m-1 条边, 每条边有 4 种布线选择, 要使该树适应值达到最优, 需要进行

4^{m-1} 的遍历计算. 如图 9 (a)所示, 引脚 *A*, *B*, *C* 构成局部 Steiner 树, 引脚 *A*, *B*, *C* 构成的局部 Steiner 树有图 9(a)~9(p)共 16 种结构, 可看出图 9(i)的 Steiner 树线长得到最大程度上的优化.其具体步骤如下：

(1) 遍历 Steiner 树的每个边, 计算每个端点的相邻端点, 每个端点与其相邻端点构成局部的 Steiner 树.

(2) 对于每棵局部 Steiner 树, 遍历所有的布线结构, 然后选择适应度值最小的布线结构进行更新.

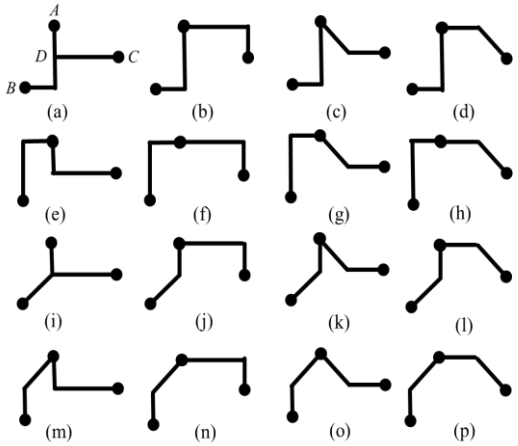


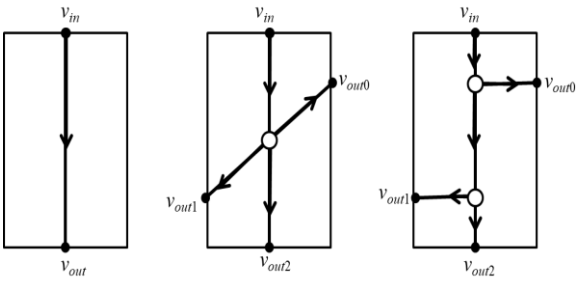
图 9 节点为 3 的布线树局部最优策略

4.5 混合修正策略

由于上述的策略得到的 Steiner 树中可能存在不满足 Slew 约束的布线, 但经过惩罚函数的加权, 其布线代价仍少于其他替换布线边. 为了充分利用该类型布线边的线长较短的优势, 本文提出混合修正策略, 通过调整违反约束的布线方案, 从而在保证整棵 Steiner 树完全满足 Slew 约束的同时, 得到线长优化的目标.

本文将内部树分为两种结构：两节点结构和多节点结构. 如图 10(a)所示, 内部树中只有一个接收节点和一个驱动节点称为双节点结构. 而如图 10(b)所示, 内部树中有多个接收节点和一个驱动节点称为多节点结构. 对于双节点结构, 由于内部树的结构简单, 可直接判断是否违反约束, 而对于多节点结构, 则需要重新构建内部树(如图 10(c)所示), 再逐点判断节点的电压转换速率是否满足约束. 对于同个障碍中存在两棵内部树情况, 需要用并查集记录内部树线段的情况.

本文将内部树分为两种结构：两节点结构和多节点结构. 如图 10(a)所示, 内部树中只有一个接收节点和一个驱动节点称为双节点结构. 而如图 10(b)所示, 内部树中有多个接收节点和一个驱动节点称为多节点结构. 对于双节点结构, 由于内部树的结构简单, 可直接判断是否违反约束, 而对于多节点结构, 则需要重新构建内部树(如图 10(c)所示), 再逐点判断节点的电压转换速率是否满足约束. 对于同个障碍中存在两棵内部树情况, 需要用并查集记录内部树线段的情况.



(a) 两节点结构 (b) 多节点结构 (c) 重构结构

图 10 内部树结构

定理 1. 与使用选择 2 或选择 3 进行连接的引脚对(p_i, p_j)的线长相比, 由 p_i 和 p_j 所形成的矩形区域中任意点作伪 Steiner 点, 与 p_i 和 p_j 使用任意布线选择进行连接都不会增加线长.

定理 2.与使用选择 0 或选择 1 进行连接的引脚对(p_i, p_j)的线长相比, 由 p_i 和 p_j 所形成的平行四边形区域中的任意点作为伪 Steiner 点, 与 p_i 和 p_j 使用选择 0 或选择 1 进行连接不会增加线长.

证明：根据线段平行关系, 对引脚对(p_i, p_j)组成的布线区域中, 利用伪 Steiner 点进行连接的具体布线的线段的简单平移和原始布线的线段比较可以容易得证明定理 1 和定理 2.

定理 3. 在多节点结构中, 上游接收节点(靠近驱动节点)的电压转换速率低于同一侧的下游接收节点的电压转换速率.

证明：假设内部树具有多节点结构, 本文逐点计算各个驱动节点的具体的电压转换速率(如图 11(a)所示). 在不失一般性的前提下, 假设 k_0 是驱动节点, k_1, k_2 和 k_3 是接收节点, 而 len_i 是内部树各线段的具体长度. k_1 是上游接收节点, k_2 是不同侧的下游接收节点, k_3 是同一侧的下游接收节点. 如图 11(a)所示, 从公式(2)可以

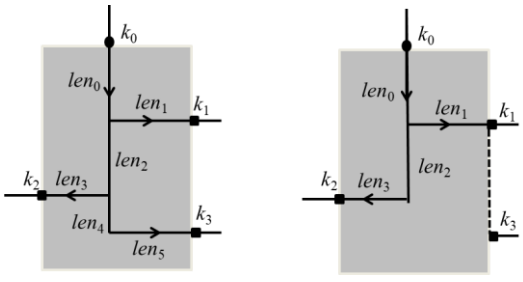
得出，$s(k_0)$的值不变，并且各个接收节点的 D_p 的计算值如下所示.

$$D_p(k_0,k_1)=r_b\times(\sum_{i=0}^{5}len_i\times c+3\times c_b)+len_0\times r\times(\sum_{i=1}^{5}len_i\times c+\frac{(len_0\times c)}{2}+3\times c_b)+len_1\times r\times(\frac{(len_1\times c)}{2}+c_b)$$

$$D_p(k_0,k_2)=r_b\times(\sum_{i=0}^{5}len_i\times c+3\times c_b)+len_0\times r\times(\sum_{i=1}^{5}len_i\times c+\frac{(len_0\times c)}{2}+3\times c_b)+len_2\times r\times(\sum_{i=3}^{5}len_i\times c+\frac{(len_2\times c)}{2}+2\times c_b)+len_3\times r\times(\frac{(len_3\times c)}{2}+c_b)$$

$$D_p(k_0,k_3)=r_b\times(\sum_{i=0}^{5}len_i\times c+3\times c_b)+len_0\times r\times(\sum_{i=1}^{5}len_i\times c+\frac{(len_0\times c)}{2}+3\times c_b)+len_2\times r\times(\sum_{i=3}^{5}len_i\times c+\frac{(len_2\times c)}{2}+2\times c_b)+len_4\times r\times((len_5+0.5\times len_4)\times c+c_b)+len_5\times r\times(\frac{(len_5\times c)}{2}+c_b)$$

由于 len_1 和 len_5 的大小相同，因此可以明显看出 $D_p(k_0, k_1)$小于 $D_p(k_0, k_3)$，即上游接收节点的电压转换速率低于同一侧的下游接收节点的电压转换速率.



(a) 4 节点的内部树　　(b) 调整后内部树

图 11. 带多节点结构的内部树

定理 4. 删除下游接收节点到内部树的连通分量可减少所有接收节点的电压转换速率.

证明：如图 11(b)所示，在不失一般性的情况下，本文通过增加外部树的线长，删除了下游接收节点 k_3 与内部树之间的连通分量. 从公式(2)可以得出，$s(k_0)$的值减小，各个接收节点的 D_p 的值如下所示.

通过比较，可以明显看出 $D_p(k_0, k_1)$和 $D_p(k_0, k_2)$的值变小，并且由于 $s(k_0)$也变小，根据公式(1)，可以得出 $s(k_1)$和 $s(k_2)$减小，即所有接收节点的电压转换速率减小.

$$D_p(k_0,k_1)=r_b\times(\sum_{i=0}^{3}len_i\times c+2\times c_b)+len_0\times r\times(\sum_{i=1}^{3}len_i\times c+\frac{(len_0\times c)}{2}+2\times c_b)+len_1\times r\times(\frac{(len_1\times c)}{2}+c_b)$$

$$D_p(k_0,k_2)=r_b\times(\sum_{i=0}^{3}len_i\times c+2\times c_b)+len_0\times r\times(\sum_{i=1}^{3}len_i\times c+\frac{(len_0\times c)}{2}+2\times c_b)+len_2\times r\times((len_3+0.5\times len_2)\times c+c_b)+len_3\times r\times(\frac{(len_3\times c)}{2}+c_b)$$

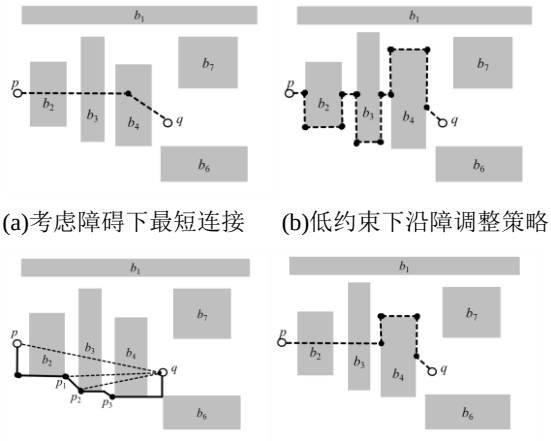
对于 Steiner 树中违反约束的片段，本文提出一种有效的混合修正策略进行调整. 针对不同情况下，本文提出沿障调整策略与总体调整策略两种有效策略. 沿障调整策略对于布线中存在违反约束的片段，沿着所在的障碍进行绕障处理. 在该策略中需要判断接收节点与驱动节点所处的障碍边，以及节点间沿障碍的最短路径. 但该策略在多片段布线中，存在由于布线经过障碍的数量与位置，而引起过多不必要绕障，从而导致线长代价的浪费.

图 12(a)是 X 结构下不考虑障碍引脚 p, q 最短的连接方式，而图 12(b)则是在电压转换速率较小下用沿障调整策略进行的修正，可以看出引脚 pq 之间三个片段独立绕障，引起过多不必要绕障. 针对这种情况，本文提出总体调整策略，从违反约束的每个布线中，对该布线经过的所有障碍的端点中选择合适的点作为伪 Steiner 点进行绕障处理. 其步骤如下.

(1) 首先对布线经过的所有障碍，按照障碍中心与该布线中空间位置靠左的引脚的距离大小进行排序.

(2) 删除原先布线，从该布线中空间位置靠左的引脚设置为出发点，另一引脚设置为终点. 按序选择障碍，并从布线经过的障碍的顶点中，选择距离与出发点和终点构成的直线最短的顶点，以 X 结构的连接方式进行处理，并

将该顶点设置为新的出发点，判断下一障碍，直至与终点相连.



(a)考虑障碍下最短连接　(b)低约束下沿障调整策略

(c)低约束下总体调整策略　(d)高约束下沿障调整策略

图 12. 混合修正策略

如图 12(c)所示，对于布线 pq，首先对经过的障碍 b_2, b_3, b_4 以和引脚 p 的距离排序，并按序选择障碍 b_2，从 b_2 的顶点中选择距离线段 pq 最近的点(p_1 点)作为伪 Steiner 点，并用 X 结构方式连接 pp_1，并将线段 p_1q 作为下个障碍判断依据，不断重复该过程，直至完全判断经过的每个障碍. 但在连接类似 pp_1 的过程中，存在较低可能性，新的布线可能会经过先前未经过的障碍. 且在电压转换速率较大情况下，如图 12(d)所示，沿障调整策略有更好的优势.

综上所述，本文结合以上两种策略，提出了一种混合修正策略. 对 Steiner 树的每条布线进行判断，对于满足该判断的布线使用总体调整策略进行修正，对使用总体调整策略调整后的 Steiner 树再进行沿障调整策略的处理，使其完全满足 Slew 约束. 在实验中，设定了判断两种策略优劣的参数 *judge*，当该参数大于 1 时选择总体调整策略，公式(21)给出具体的计算公式.

$$judge = len_1 / len_2 \tag{21}$$

其中 len_1 与 len_2 分别为沿障调整策略和总体调整策略的代价因子. 公式(22)和(23)分别给出相应的计算方法.

$$len_1 = \sum_{i \in set(o_i)} \left(\left| x_{i1} - x_{i2} \right| + \left| y_{i1} - y_{i2} \right| - l_i \right) \tag{22}$$

$$\begin{aligned} len_2 = & \left| \max(set(x_{i1})) - \min(set(x_{i2})) \right| \\ & + \left| \max(set(y_{i1})) - \min(set(y_{i2})) \right| \end{aligned} \tag{23}$$

其中 i 为障碍的序号，x 与 y 为障碍 i 对角线端点的横纵坐标，l_i 代表片段在障碍 i 内的长度，$set(O_i)$代表布线经过的所有障碍的集合.

4.6 算法时间复杂度分析

引理 1. 假设种群大小为 p，迭代次数为 $iters$，引脚个数为 n，障碍个数为 m，HDPSO-XSMT-SC 算法的时间复杂度为 $O(m \times n^2 + n \times m^2 + n^2 \log n + p \times iters \times n \log n)$.

证明. 在初始化阶段，本文利用 Prim 算法获取最小生成树，并用该最小生成树的(n-1)条边初始化种群，其时间复杂度为 $O(n^2 + np)$，在预处理阶段，算法需要记录任意的两个引脚间，即 $n \times (n - 1)$条边与 m 个障碍的相交的信息，故预处理阶段的时间复杂度为 $O(m \times n^2)$. 在 PSO 搜寻阶段，在 PSO 的内部循环中，主要包括变异操作，交叉操作，适应值计算操作. 其中，在变异操作中，删边与生成新边可在常数时间内完成，在交叉操作中，对树的遍历判断可在线性时间内完成，但由于在变异操作与交叉操作使用并查集策略，所以时间复杂度为 O($n$log$n$)，而在适应值中线长的计算取决于排序方法的复杂度，所以 PSO 的内部循环时间复杂度为 O(n log$n$)，而外部循环取决于种群规模及迭代次数，故 PSO 搜寻阶段的时间复杂度为 $O(p \times iters \times n \log n)$. 在局部最优策略中，算法判断了以 n 个引脚为根节点的子树，并为了避免子树的规模过大，将子树边的规模限制为常量，在边的变换过程中，需要进行适应值的计算，所以局部最优策略时间复杂度为 $O(n^2 \log n)$. 在混合修正策略中，总体调整策略最坏情况下需要为(n - 1)条边经过所有障碍都建立新边，并获取新边与障碍的相交信息，其时间复杂度为 $O(n \times m^2)$. 而在沿边调整策略中，需要调整 m 个障碍内部中，最多由 n 条边构成的内部树结构，所以沿边调整策略的时间复杂度为 $O(n \times m)$，故混合修正策略的时间复杂度为 $O(m \times n + n \times m^2) = O(n \times m^2)$. 综上所述，HDPSO-XSMT-SC 算法的时间复杂度为 $O(m \times n^2 + n \times m^2 + n^2 \log n + p \times iters \times n \log n)$.

在相同的问题模型下，文献[35]的时间复杂度为 $O(n^2 \times v \log v)$. 其中 v 为通过 GeoSteiner 软件构造出的布线图的顶点，其规模略大于引脚个数 n. 通过比较分析，$O(n^2 \times v \log v) \approx O(n^3 \log n)$，而在不失一般性情况下，$O(p \times iters) \approx O(n^2)$. 由此可见，两个算法均可在相同量级的多项式的时间内完成 NP 难的 Steiner 树构造问题.

5 实验结果

本文算法用 C/C++语言实现，所有的实验均在 2.8GHz CPU 以及 4G 内存的 PC 上单处理器单进程完成. 本文测试了在经典 OASMT 问题中常用的 16 组标准测试电路，在表 5 中给出对应的测试电路的引脚及障碍数量.

本文算法在五种不同大小的电压转换速率约束下进行测试验证. 具体约束如下：

(1) 0；

(2) 20% slew = $slew_{min}$ +20%($slew_{max}$-$slew_{min}$)；

(3) 50% slew = $slew_{min}$ +50%($slew_{max}$-$slew_{min}$)；

(4) 80% slew = $slew_{min}$ +80%($slew_{max}$-$slew_{min}$)；

(5) ∞；

其中约束条件(2)到(4)与文献[32]一致，(1)到(5)与文献[35]一致.

表 5 测试电路规模

测试电路	引脚数	障碍数
IND1	10	32
IND2	10	43
IND3	10	50
IND4	25	79
IND5	33	71
RC01	10	10
RC02	30	10
RC03	50	10
RC04	70	10
RC05	100	10
RC06	100	500
RC07	200	500
RC08	200	800
RC09	200	1000
RC10	500	100
RC11	1000	100

5.1 预处理策略的有效性验证

为了验证预处理策略的有效性，本文将未采用预处理策略与采用预处理策略的计算次数进行比较，其中将计算任意一引脚对的布线与任意一障碍之间信息视为一次计算次数.表 6 给出未采用和采用预处理策略的计算次数，其中每个约束条件中最后一列表示减少率，其具体公式如下：

$$IMP_0 = \frac{\text{未采取预处理-采取预处理}}{\text{未采取预处理}} \times 100\% \quad (24)$$

从表 6 中可以看出采用本文所提的预处理策略相对未采用预处理策略而言每个约束条件下均能减少 95.7%以上的大幅度计算量，从而有机会更好减少本文算法的时间成本. 这主要是由于本文提出的预处理策略设计了两种查找表用以判断是否经过障碍和经过障碍的信息，从而预先记录下引脚与障碍之间的布线信息，对 PSO 搜寻过程以及之后步骤的执行减少大量重复判断和计算的时间.

5.2 惩罚机制策略的有效性验证

为了验证本文提出的惩罚机制策略的有效性，本文将未采用惩罚机制策略所得到的线长与采用惩罚机制策略所得到的线长进行对比，并用指标 IMP_1 衡量优化程度，具体公式为公式(25).

$$IMP_1 = \frac{\text{未惩罚线长 - 惩罚线长}}{\text{未惩罚线长}} \times 100\% \quad (25)$$

从表 7 可以看出，在五种不同的约束条件下，采用惩罚机制策略相对未采用惩罚机制策略能取得平均 4.8%、3.2%、1.7%、0.3%以及 0.4%不同程度上的线长优化. 这是由于在 Steiner 树中，对于穿过障碍但不满足约束的布线需要花费额外的布线代价进行修复，所以需要对该类型布线进行一定惩罚以减少该类型布线产生. 在 Slew 约束越严格的情况下，惩罚机制策略所带来的优化效果越明显，即最大电压转换速率值最严格的前两种情况下(分别为 0 和 20% slew)，惩罚机制对布线代价分别带来 4.8%和 3.2%的优化效果.

5.3 局部最优策略的有效性验证

为了验证局部最优策略的有效性，本文将使用该策略前后的线长进行对比，并用指标 IMP_2 衡量优化程度，具体公式为公式(26).

$$IMP_2 = \frac{\text{优化前线长-优化后线长}}{\text{未优化前线长}} \times 100\% \quad (26)$$

从表 8 中可以看出在五种不同的约束条件下，局部最优策略能取得平均 5.5%、4.5%、4.4%、4.2%和 4.5%等不同程度上的线长优化效果. 在约束条件为 0 情况下，采用局部最优策略在测试电路 RC03 上可取得 12.2%的最大线长优化；在 20%slew 条件下，采用局部最优策略在测试电路 RC10 上可取得 9.3%的最大线长优化；在 50% slew 条件下，采用局部最优策略在测试电路 RC11 上可取得 8.7%的最大线长优化；在 80%slew 条件下，采用局部最优策略在测试电路 RC11 上可取得 8.6%的最大线长优化；在∞约束下，采用局部最优策略在测试电路 RC10 和 RC11 上可取得 8.4%的最大线长优化.

取得如此明显的优化效果主要是因为本文提出 程度，从而达到线长代价减少的目的.
的局部最优策略能够进一步提高布线间的共享 .

表 6 预处理策略有效性验证

测试电路	0			20% slew			50% slew			80% slew			∞		
	预处理(10^6)		IMP_0	预处理(10^6)		IMP_0	预处理(10^6)		IMP_0	预处理(10^6)		IMP_0	预处理(10^6)		IMP_0
	未采用	采用		未采用	采用		未采用	采用		未采用	采用		未采用	采用	
IND1	4.87	0.15	96.8%	4.95	0.16	96.8%	4.91	0.15	96.8%	4.90	0.15	96.8%	0.49	0.16	96.8%
IND2	4.23	0.16	96.3%	4.17	0.16	96.3%	4.21	0.16	96.3%	4.15	0.15	96.3%	4.17	0.16	96.3%
IND3	5.41	0.16	97.1%	5.42	0.16	97.1%	5.37	0.15	97.1%	5.34	0.15	97.1%	5.48	0.16	97.1%
IND4	34.48	0.45	98.7%	33.72	0.44	98.7%	34.15	0.44	98.7%	33.22	0.43	98.7%	33.40	0.43	98.7%
IND5	44.03	0.62	98.6%	41.87	0.58	98.6%	41.87	0.58	98.6%	41.46	0.58	98.6%	41.74	0.58	98.6%
RC01	1.55	0.16	90.0%	1.54	0.15	90.0%	1.39	0.15	88.9%	1.39	0.15	88.9%	1.38	0.15	88.9%
RC02	5.11	0.51	90.0%	5.15	0.52	90.0%	5.13	0.51	90.0%	5.15	0.52	90.0%	5.16	0.52	90.0%
RC03	8.91	0.89	90.0%	8.88	0.89	90.0%	8.90	0.89	90.0%	8.86	0.89	90.0%	8.99	0.90	90.0%
RC04	11.72	1.31	88.9%	11.61	1.29	88.9%	11.64	1.30	88.9%	11.56	12.87	88.9%	11.65	12.97	88.9%
RC05	19.13	1.92	90.0%	19.02	1.91	90.0%	19.22	1.93	90.0%	19.32	1.94	90.0%	19.20	1.92	90.0%
RC06	1156.52	2.67	99.8%	988.02	2.23	99.8%	989.73	2.22	99.8%	991.11	2.23	99.8%	990.30	2.22	99.8%
RC07	2559.75	5.71	99.8%	2252.94	4.98	99.8%	2247.02	4.96	99.8%	2277.21	5.02	99.8%	2257.60	4.98	99.8%
RC08	4812.26	7.14	99.9%	3724.69	5.39	99.9%	3759.79	5.44	99.9%	3714.03	5.37	99.9%	3726.36	5.38	99.9%
RC09	6093.42	7.61	99.9%	4498.64	5.44	99.9%	4513.78	5.47	99.9%	4163.13	5.33	98.7%	4454.30	5.37	99.9%
RC10	1489.70	15.10	99.0%	1464.47	14.84	99.0%	1468.15	14.88	99.0%	1464.46	14.85	99.0%	1467.30	14.87	99.0%
RC11	4345.50	44.30	99.0%	4304.12	43.87	99.0%	4279.51	43.62	99.0%	4274.93	43.58	99.0%	4284.98	43.68	99.0%
Ave			**95.9%**			**95.9%**			**95.8%**			**95.7%**			**95.8%**

表 7 惩罚机制策略的有效性验证

测试电路	0			20%slew			50%slew			80%slew			∞		
	未惩罚	惩罚	IMP_1	未惩罚	惩罚	IMP_1	未惩罚	惩罚	IMP_1	未惩罚	惩罚	IMP_1	未惩罚	惩罚	IMP_1
IND1	570	562	1.4%	564	562	0.4%	562	562	0.0%	562	562	0.0%	562	562	0.0%
IND2	9189	9007	2.0%	9148	9007	1.5%	9072	9007	0.7%	8931	8789	1.6%	8872	8789	0.9%
IND3	571	567	0.7%	585	558	4.6%	564	558	1.1%	554	546	1.4%	546	546	0.0%
IND4	1057	979	7.4%	984	968	1.6%	977	955	2.3%	969	952	1.8%	960	952	0.8%
IND5	1312	1283	2.2%	1296	1252	3.4%	1211	1172	3.2%	1188	1172	1.3%	1170	1155	1.3%
RC01	25194	25165	0.1%	25078	24547	2.1%	23969	23957	0.1%	23975	23957	0.1%	23846	23846	0.0%
RC02	41059	39482	3.8%	40441	36858	8.9%	38109	36534	4.1%	36291	36192	0.3%	36139	36124	0.0%
RC03	58874	53287	9.5%	53501	52396	2.1%	50923	48276	5.2%	48399	48346	0.1%	48510	48418	0.2%
RC04	65893	59318	10.0%	58280	52567	9.8%	58261	52157	10.5%	52099	51886	0.4%	51911	51886	0.0%
RC05	71765	70259	2.1%	73408	68029	7.3%	68137	68131	0.0%	68136	68113	0.0%	68126	68113	0.0%
RC06	82472	78302	5.1%	80528	75205	6.6%	74898	74124	1.0%	74520	74292	0.3%	74346	73136	1.6%
RC07	113088	106189	6.1%	103997	101080	2.8%	101692	101024	0.7%	100831	100638	0.2%	100682	99374	1.3%
RC08	123410	115978	6.0%	109996	107825	2.0%	106992	105878	1.0%	104716	104494	0.2%	105953	102915	2.9%
RC09	120639	113150	6.2%	105955	103706	2.1%	102387	101718	0.7%	101292	100984	0.3%	100800	99417	1.4%
RC10	170814	156049	8.6%	155129	152274	1.8%	151985	151887	0.1%	151853	151728	0.1%	151908	151466	0.3%
RC11	237723	212287	10.7%	218566	217575	0.5%	217194	217160	0.0%	216913	216859	0.0%	217025	216769	0.1%
Ave			**4.8%**			**3.2%**			**1.7%**			**0.3%**			**0.4%**

表 8 局部最优策略有效性验证

测试电路	0			20%slew			50%slew			80%slew			∞		
	优化前	优化后	IMP_2	优化前	优化后	IMP_2	优化前	优化后	IMP_2	优化前	优化后	IMP_2	优化前	优化后	IMP_2
IND1	562	562	0.0%	562	562	0.0%	562	562	0.0%	562	562	0.0%	562	562	0.0%
IND2	9067	9007	0.7%	9007	9007	0.0%	9007	9007	0.0%	9007	8789	2.5%	9007	8789	2.5%
IND3	584	567	3.0%	571	558	2.3%	565	558	1.3%	546	546	0.0%	546	546	0.0%
IND4	1015	979	3.7%	980	968	1.2%	968	955	1.4%	957	952	0.5%	959	952	0.7%
IND5	1292	1283	0.7%	1257	1252	0.4%	1206	1172	2.9%	1207	1172	3.0%	1181	1155	2.3%
RC01	25203	25165	0.2%	25159	24547	2.5%	24753	23957	3.3%	24707	23957	3.1%	24659	23846	3.4%
RC02	43088	39482	9.1%	37701	36858	2.3%	37446	36534	2.5%	37457	36192	3.5%	36866	36124	2.1%
RC03	59782	53287	12.2%	55298	52396	5.5%	51177	48276	6.0%	50141	48346	3.7%	50917	48418	5.2%
RC04	62514	59318	5.4%	55654	52567	5.9%	55331	52157	6.1%	54920	51886	5.8%	54374	51886	4.8%
RC05	77320	70259	10 0%	73408	68029	7.9%	72723	68131	6.7%	72761	68113	6.8%	72322	68113	6.2%
RC06	82256	78302	5.0%	79692	75205	6.0%	78899	74124	6.4%	77760	74292	4.7%	77830	73136	6.4%
RC07	112258	106189	5.7%	108795	101080	7.6%	106598	101024	5.5%	106291	100638	5.6%	106185	99374	6.9%
RC08	123278	115978	6.3%	114971	107825	6.6%	111845	105878	5.6%	110658	104494	5.9%	111135	102915	8.0%
RC09	119572	113150	5.7%	110684	103706	6.7%	107606	101718	5.8%	105836	100984	4.8%	105977	99417	6.6%
RC10	170056	156049	9.0%	166505	152274	9.3%	164387	151887	8.2%	163866	151728	8.0%	164225	151466	8.4%
RC11	236677	212287	11.5%	235725	217575	8.3%	236044	217160	8.7%	235543	216859	8.6%	234948	216769	8.4%
Ave			**5.5%**			**4.5%**			**4.4%**			**4.2%**			**4.5%**

5.4 混合修正策略的有效性验证

为了验证混合修正策略的有效性，本文将未修正前 Steiner 树的违反 Slew 约束的片段数量、沿障修正策略修复的违反 Slew 约束的片段数量、总体修正策略修复的违反 Slew 约束的片段数量进行了统计.

从表 9 可看出，在 Slew 约束较为严格情况下，两种调整策略均能修复违反约束的片段，且采用混合修正策略后，违反约束的片段为 0. 在 Slew 约束较为宽松情况下，部分或全部测试电路中，未采用调整策略前 Steiner 树的违反约束片段数量为 0，即可视为忽略障碍下进行布线，无需进行调整. 取得这种良好优化效果的原因主要是本文提出的混合修正策略，分别通过沿障修正策略与总体修正策略两种方式修正违反 Slew 约束的布线方案,可保证采用混合的修正策略后最终的 Steiner 树完全满足 Slew 约束，从而说明本文所提混合修正策略的有效性

5.5 本文算法的有效性验证

为了验证本文算法的有效性，本文与两种考虑 Slew 约束的同类工作进行比较,本文使用并用指标 IMP_3 衡量优化程度，公式(27)给出相应公式.

$$IMP_3=\frac{\text{其他算法线长-本文算法线长}}{\text{本文算法线长}}\times 100\% \quad (27)$$

在表 10 和表 11 中, BOB 列表示文献[32]算法得到的线长，RRH 列表示文献[35]算法得到的线长, OUR 列表示本文算法得到的线长. 从表 10 可以看出，与 BOB 算法相比，在 20%slew 条件下，本文算法取得 3.0%~14.5%的线长优化，平均优化 6.6%；在 50%slew 条件下，取得 3.5%~16.9%的线长优化，平均优化 7.9%；在 80%slew 条件下，取得 3.9%~13.9%的线长优化，平均优化 7.5%. 从表 11 可以看出，与 RRH 算法相比，在约束条件为 0 情况下，取得-0.5%~19.9%的线长优化，平均优化 7.7%；在 20%slew 条件下，取得 1.6%~21.5%的线长优化，平均优化 8.9%; 在 50%slew 条件下，取得 3.5%~16.6%的线长优化，平均优化 8.6%；在 80%slew 条件下，取得 3.5%~14.7%的线长优化，平均优

化 7.9%；在∞约束下，取得 3.5%~14.7%的线长优化，平均优化 8.0%.

综上，本文算法相对于现有同类工作，融入了离散 PSO 搜索策略、惩罚机制策略、预处理策略、局部最优策略、混合修正策略等多种有效的策略,在不同角度上对布线树进行优化，从而可取得最佳的布线代价，最终验证本文算法的有效性.

表 9 调整策略有效性验证

测试电路	0			20%slew			50%slew			80%slew			∞		
	无	沿障	总体	无	沿障	总体	无	沿障	总体	无	沿障	总体	无	沿障	总体
IND1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
IND2	2	1	1	1	1	0	1	1	0	0	0	0	0	0	0
IND3	3	3	0	2	2	0	2	2	0	0	0	0	0	0	0
IND4	10	3	7	2	0	2	0	0	0	0	0	0	0	0	0
IND5	25	10	15	10	5	5	1	1	0	0	0	0	0	0	0
RC01	3	0	3	3	2	1	3	2	1	1	1	0	0	0	0
RC02	7	7	0	1	0	1	0	0	0	0	0	0	0	0	0
RC03	9	8	1	3	1	2	3	3	0	1	1	0	0	0	0
RC04	6	5	1	4	2	2	0	0	0	0	0	0	0	0	0
RC05	10	9	1	0	0	0	0	0	0	0	0	0	0	0	0
RC06	101	35	66	23	11	12	5	2	3	0	0	0	0	0	0
RC07	106	38	68	22	8	14	3	1	2	0	0	0	0	0	0
RC08	214	101	113	32	17	15	6	3	3	0	0	0	0	0	0
RC09	229	111	118	43	26	17	7	3	4	0	0	0	0	0	0
RC10	59	40	19	13	9	4	0	0	0	0	0	0	0	0	0
RC11	40	18	22	14	6	8	4	2	2	3	2	1	0	0	0

表 10 本文算法与 BOB 算法对比

测试电路	20%slew			50%slew			80%slew		
	BOB	OUR	IMP_3	BOB	OUR	IMP_3	BOB	OUR	IMP_3
RC01	25290	24547	3.0%	25290	23957	5.6%	25290	23846	6.1%
RC02	42218	36858	14.5%	42710	36534	16.9%	41210	36192	13.9%
RC03	54480	52396	4.0%	54480	48276	12.9%	52910	48346	9.4%
RC04	55450	52567	5.5%	55450	52157	6.3%	55447	51886	6.9%
RC05	73400	68029	7.9%	73400	68131	7.7%	73730	68113	8.2%
RC06	78650	75205	4.6%	76593	74124	3.3%	77481	74292	4.3%
RC07	110250	101080	9.1%	108947	101024	7.8%	107809	100638	7.1%
RC08	111810	107825	3.7%	109564	105878	3.5%	108569	104494	3.9%
RC09	109661	103706	5.7%	109661	101718	7.8%	108218	100984	7.2%
RC10	164720	152274	8.2%	164770	151887	8.5%	164770	151728	8.6%
RC11	232535	217575	6.9%	231730	217160	6.7%	231780	216859	6.9%
Ave			**6.6%**			**7.9%**			**7.5%**

表 11 本文算法与 RRH 算法进行对比

测试电路	0			20%slew			50%slew			80%slew			∞		
	RRH	OUR	IMP_3	RRH	OUR	IMP_3	RRH	OUR	IMP_3	RRH	OUR	IMP_3	RRH	OUR	IMP_3
IND1	614	562	9.3%	614	562	9.3%	614	562	9.3%	604	562	7.5%	604	562	7.5%
IND2	10800	9007	19.9%	10800	9007	19.9%	10500	9007	16.6%	9100	8789	3.5%	9100	8789	3.5%
IND3	678	567	19.6%	678	558	21.5%	600	558	7.5%	600	546	9.9%	587	546	7.5%
IND4	1155	979	18.0%	1097	968	13.3%	1092	955	14.3%	1092	952	14.7%	1092	952	14.7%
IND5	INF.	1283	INF	1357	1252	8.4%	1333	1172	13.7%	1320	1172	12.6%	1315	1155	13.9%
RC01	25980	25165	3.2%	25980	24547	5.8%	25290	23957	5.6%	25290	23957	5.6%	25290	23846	6.1%
RC02	42570	39482	7.8%	40670	36858	10.3%	40670	36534	11.3%	40670	36192	12.4%	40040	36124	10.8%
RC03	54660	53287	2.6%	53240	52396	1.6%	53010	48276	9.8%	53100	48346	9.8%	52110	48418	7.6%
RC04	59980	59318	1.1%	57360	52567	9.1%	55720	52157	6.8%	55720	51886	7.4%	55570	51886	7.1%
RC05	75110	70259	6.9%	72930	68029	7.2%	72520	68131	6.4%	72460	68113	6.4%	71950	68113	5.6%
RC06	81306	78302	3.8%	78888	75205	4.9%	77773	74124	4.9%	77607	74292	4.5%	77483	73136	5.9%
RC07	111084	106189	4.6%	108353	101080	7.2%	106799	101024	5.7%	106525	100638	5.8%	106525	99374	7.2%
RC08	115414	115978	-0.5%	111008	107825	3.0%	109622	105878	3.5%	109249	104494	4.6%	109027	102915	5.9%
RC09	115017	113150	1.7%	109403	103706	5.5%	107548	101718	5.7%	107023	100984	6.0%	107023	99417	7.7%
RC10	167330	156049	7.2%	165030	152274	8.4%	165010	151887	8.6%	164890	151728	8.7%	164910	151466	8.9%
RC11	234603	212287	10.5%	233957	217575	7.5%	233321	217160	7.4%	233204	216859	7.5%	233282	216769	7.6%
Ave			**7.7%**			**8.9%**			**8.6%**			**7.9%**			**8.0%**

结束语

针对 Slew 约束这一更贴近实际芯片设计的布线问题，为了优化线长这一最重要目标，本文首次提出基于混合离散粒子群优化的 Slew 约束 X 结构 Steiner 最小树构造算法. 首先，为了能够有效求解该离散问题，算法结合了引入变异算子与交叉算子的 PSO 离散更新操作，同时提出一种更适合遗传算子的引脚对编码方式. 其次，使用预处理策略，通过记录引脚间以及与障碍内的信息，大大节省电压转换速率的计算. 再次，设计一种合理的惩罚机制能够有效考虑到电压转换速率约束. 然后，通过多次迭代搜索出质量较好的全局最优粒子，并通过局部最优策略，使得 Steiner 最小树的局部结构达到最优，从而进一步缩短了 Steiner 树的线长. 最后，设计了一种高效的混合修正策略，使得 Steiner 最小树能够完全满足 Slew 约束. 该算法与同类算法相比，实现最佳的布线结果.

参考文献

[1] Xu Ning, Hong Xian-Long. Very large scale integration physical design theory and method. Beijing: Tsinghua University Press, 2009. (in Chinese)
(徐宁，洪先龙. 超大规模集成电路物理设计理论与方法. 北京: 清华大学出版社, 2009)

[2] Sherwani N A. Algorithms for VLSI physical design automation. Berlin,Germany: Springer Science & Business Media, 2012.

[3] Hong Xian-Long, Zhu Qi, Jing Tong, Wang Yin, Yang Yang, Cai Yi-Ci. Non-rectilinear on-chip interconnect-an efficient routing solution with high performance. Chinese Journal of Semiconductors, 2003, 24(3): 225-233. (in Chinese)
(洪先龙,朱祺,经彤,王垠,杨旸,蔡懿慈. 非直角互连——布线技术发展的新趋势. 半导体学报, 2003, 24(03): 225-233)

[4] Teig S L. The X architecture: not your father's diagonal wiring//Proceedings of the 2002 International Workshop on System-Level Interconnect Prediction. San Diego, California, USA, 2002: 33-37.

[5] Liu Geng-Geng, Zhuang Zhen, Guo Wen-Zhong, Chen Guo-Long. A high performance X-Architecture multilayer global router for VLSI. Acta Automatica Sinica, 2020, 46(1): 79-93. (in Chinese)
(刘耿耿，庄震，郭文忠，陈国龙. VLSI 中高性能 X 结构多层总体布线器. 自动化学报, 2020, 46(1): 79-93)

[6] Garey M R, Johnson D S. The rectilinear Steiner tree problem is NP-complete. SIAM Journal on Applied Mathematics, 1977, 32(4): 826-834.

[7] Eberhart R, Kennedy J. A new optimizer using particles swarm theory//Proceedings of the 6th International Symposium on Micro Machine and Human Science. Los Alamitos, USA, 1995: 39-43.

[8] Su Jin-Shu, Guo Wen-Zhong, Yu Chao-Long, Chen Guo-Long. Fault-tolerance clustering algorithm with load-balance aware in wireless sensor network. Chinese Journal of Computers, 2014, 37(02): 445-456. (in Chinese)
(苏金树, 郭文忠, 余朝龙, 陈国龙. 负载均衡感知的无线传感器网络容错分簇算法. 计算机学报, 2014, 37(02): 445-456)

[9] Hu Xin-Ping, He Yu-Zhi, Ni Wei-Wei, Zhang Yong. A privacy-preserving data publishing method based on genetic algorithm with roulette wheel. Journal of Computer Research and Development, 2012, 49(11): 2432-2439. (in Chinese)
(胡新平, 贺玉芝, 倪巍伟, 张勇. 基于赌轮选择遗传算法的数据隐藏发布方法. 计算机研究与发展, 2012, 49(11): 2432-2439)

[10] Li Jie, Bai Zhi-Hong, Yu Rui-Yun, Cui Ya-Meng, Wang Xing-Wei. Mobile location privacy protection algorithm based on PSO optimization. Chinese Journal of Computers, 2018, 41(05): 1037-1051. (in Chinese)
(李婕, 白志宏, 于瑞云, 崔亚盟, 王兴伟. 基于 PSO 优化的移动位置隐私保护算法. 计算机学报, 2018, 41(05): 1037-1051)

[11] Wang Zhu-Rong, Xue Wei, Hei Xin-Hong, Fei Rong, Yi Zhen-Zhen. The multi-phase particle swarm optimization for solving the capacitated p-median problem. Chinese Journal of Computers, 2019, 43(6): 1-27. (in Chinese)
(王竹荣, 薛伟, 黑新宏, 费蓉, 伊珍珍. 多阶段粒子群优化算法求解容量约束 p-中位问题. 计算机学报, 2019, 43(6): 1-27)

[12] Liu G G, Huang X, Guo W Z, Niu Y Z, Chen G L. Multilayer obstacle-avoiding X-architecture steiner minimal tree construction based on particle swarm optimization. IEEE Transactions on Cybernetics, 2015, 45(5): 989-1002.

[13] Liu G G, Chen Z S, Zhuang Z, Guo W Z, Chen G L. A unified algorithm based on HTS and self-adapting PSO for the construction of octagonal and rectilinear SMT. Soft Computing, 2020, 24(6): 3943–3961.

[14] Guo Wen-Zhong, Chen Xiao-Hua, Liu Geng-Geng, Chen Guo-Long. Track assignment algorithm based on hybrid discrete particle swarm optimization. Pattern Recognition and Artificial Intelligence, 2019, 32(08): 758-770. (in Chinese)
(郭文忠, 陈晓华, 刘耿耿, 陈国龙. 基于混合离散粒子群优化的轨道分配算法. 模式识别与人工智能, 2019, 32(08): 758-770)

[15] Liu Geng-Geng, Chen Zhi-Sheng, Guo Wen-Zhong, Chen Guo-Long. A self-adapting PSO algorithm with efficient hybrid transformation strategy for X-architecture Steiner minimal tree construction algorithm. Pattern Recognition and Artificial Intelligence, 2018, 31(5): 398-408. (in Chinese)
(刘耿耿, 陈志盛, 郭文忠, 陈国龙. 基于自适应 PSO 和混合转换策略的 X 结构 Steiner 最小树算法. 模式识别与人工智能, 2018, 31(5): 398-408)

[16] Liu G G, Chen G L, Guo W Z, Chen Z. DPSO-based rectilinear steiner minimal tree construction considering bend reduction//Proceedings of the Seventh International Conference on Natural Computation. Shanghai, China, 2011, 1161-1165.

[17] Nath S, Gupta S, Biswas S, Banerjee R, Sing J K, Sarkar. GPSO hybrid algorithm for rectilinear steiner tree optimization//Proceedings of the 2020 IEEE VLSI Device Circuit and system (VLSI DCS). Kolkata, India, 2020: 365-369.

[18] Chu C, Wong Y C. FLUTE: fast lookup table based rectilinear Steiner minimal tree algorithm for VLSI design. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2007, 27(1): 70-83.

[19] Lin S E D, Kim D H. Construction of all rectilinear steiner minimum trees on the hanan grid and its applications to VLSI design. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2019, 39(6): 1165-1176.

[20] Latha N R, Prasad G R. Memory and I/O optimized rectilinear steiner minimum tree routing for VLSI. International Journal of Electrical and Computer Engineering, 2020, 10(3): 2959-2968.

[21] Kundu S, Roy S, Mukherjee S. Rectilinear steiner tree construction techniques using PB-SAT-based methodology. Journal of Circuits, Systems and Computers, 2020, 29(04): 2050057:1-2050057:22.

[22] Wu H L, Xu S J, Zhen Z, Liu G G. X-Architecture steiner minimal tree construction based on discrete differential evolution//Proceedings of the International Conference on Natural Computation, Fuzzy Systems and Knowledge

Discovery. Kunming, China, 2019: 433-442.

[23] Huang T, Young E F Y. Obstacle-avoiding rectilinear Steiner minimum tree construction: An optimal approach //Proceedings of the International Conference on Computer Aided Design. New York, USA, 2010: 610-613.

[24] Chow W K, Li L, Young E F Y, Sham C W. Obstacle-avoiding rectilinear Steiner tree construction in sequential and parallel approach. Integration, the VLSI journal, 2014, 47(1): 105-114.

[25] Coulston C S. Constructing exact octagonal Steiner minimal tree//Proceedings of the 13th ACM Great Lakes Symposium on VLSI. New York, USA, 2003: 1-6.

[26] Huang X, Guo W Z, Liu G G, Niu Y Z Chen G L. Obstacle-avoiding algorithm in X-architecture based on discrete particle swarm optimization for VLSI design. ACM Transactions on Design Automation of Electronic Systems, 2015, 20(2): 1-28.

[27] Huang X, Guo W Z, Liu G G, Chen G L. MLXR: multi-layer obstacle-avoiding X-architecture Steiner tree construction for VLSI routing. Science China Information Sciences, 2017, 60(1), 19102: 1-19102: 3.

[28] Wang R Y, Pai C C, Wang J J, Wen H T, Pai Y C, Chang Y M, James C M L, Jiang J H. Efficient multi-layer obstacle-avoiding region-to-region rectilinear steiner tree construction//Proceedings of the 55th Annual Design Automation Conference. San Francisco, USA, 2018: 1-6.

[29] Lee M C, Jan G E, Luo C C. An efficient rectilinear and octilinear steiner minimal tree algorithm for multidimensional environments. IEEE Access, 2020, 8: 48141-48150.

[30] Müller-Hannemann M, Peyer S. Approximation of rectilinear Steiner trees with length restrictions on obstacles. //Proceedings of the Algorithms and Data Structures. Berlin, Germany, 2003: 207-218.

[31] Held S, Spirkl S T. A fast algorithm for rectilinear steiner trees with length restrictions on obstacles//Proceedings of the 2014 on International Symposium on Physical Design. Petaluma, USA, 2014: 37-44.

[32] Zhang Y, Chakraborty A. Chowdhury S, Pan D Z. Reclaiming over-the-IP-block routing resources with buffering-aware rectilinear Steiner minimum tree construction//Proceedings of the International Conference on Computer-Aided Design. California, USA, 2012: 137-143.

[33] Huang T, Young E F Y. Construction of rectilinear Steiner minimum trees with slew constraints over obstacles//Proceedings of the International Conferenceon Computer-Aided Design. California, USA, 2012: 144-151.

[34] Kashyap C V, Alpert C J, Liu F, Devgan A. PERI: a technique for extending delay and slew metrics to ramp inputs//Proceedings of the 8th ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems. Monterey, USA, 2002: 57-62.

[35] Zhang H, Ye D Y, Guo W Z. A heuristic for constructing a rectilinear Steiner tree by reusing routing resources over obstacles. Integration: The VLSI Journal, 2016, 55: 162-175.

[36] Kashyap C V, Alpert C J, Liu F, Devgan A. Closed-form expressions for extending step delay and slew metrics to ramp inputs for RC trees. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2004, 23(4): 509-516.

[37] H.Bakoglu, Circuits, Interconnections and Packaging for VLSI, Addison-Wesley, 1990: 81-133.

[38] Elmore W C. The transient response of damped linear networks with particular regard to wideband amplifiers. Journal of applied physics, 1948, 19(1): 55-63.

[39] Hu S, Alpert C J, Hu J, Karandikar S K, Li Z, Shi W, Sze C N. Fast algorithms for slew-constrained minimum cost buffering. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2007, 26(11): 2009-2022.

[40] Shi Y, Eberhart R C. Parameter selection in particle swarm optimization//Proceedings of the International conference on evolutionary programming. Berlin, Germany,1998: 591-600.

[41] Ratnaweera A, Halgamuge S K, Watson H C. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Transactions on evolutionary computation, 2004, 8(3): 240-255.

LIU Genggeng, Ph.D., associate professor, Ph.D. supervisor. His research interests include EDA algorithm, computational intelligence and its application.

HUANG Yifei, master student. His research interests include EDA design algorithm.

WANG Xin, Ph. D., professor. His main research interests include large-scale knowledge processing, computational intelligence and its application.

GUO Wenzhong, Ph.D., professor, Ph.D. supervisor. His research interests include EDA algorithm, computational intelligence and its application.

CHEN Guolong, Ph.D., professor, Ph.D. supervisor. His research interests include EDA algorithm, computational intelligence and its application.

Background

As the basic model for very large scale integration (VLSI) routing, the Steiner minimal tree (SMT) can be used in various practical problems, such as wirelength optimization, congestion, and time delay estimation. With the development of IC technology, more and more obstacles appear in the physical design process, such as some pre-routed nets and macro cells which makes the obstacle-avoiding Steiner tree construction become a hot spot for researchers to study. It is indeed possible to avoid obstacles by increasing the wirelength, but it can lead to timing violations. Therefore, it is necessary to relax the routing resources in the routing process, and then the Steiner minimal tree construction with slew constraints which can effectively prevent signal distortion is proposed.

At present, most of the researches on Steiner trees considering routing resource relaxation focus on the Manhattan architecture. Further considering the X-architecture with better wirelength optimization and the model of slew constraints that can effectively prevent signal distortion, this paper is the first work to propose an X-architecture Steiner minimal tree algorithm with slew constraints which has following contributions: (1) an effective discrete update operation formula of particle swarm optimization algorithm is proposed based on the genetic operators, (2) a pin-pair coding method is proposed to be more suitable for genetic operators, (3) an effective preprocessing strategy is proposed to avoid the frequent calculation of slew, (4) a targeted penalty mechanism is proposed to effectively consider the slew constraint, (5) an effective local optimal refining strategy is presented to further optimize wirelength of the Steiner tree, (6) a hybrid correction strategy is proposed to fully satisfy the slew constraint. Experiments show that the proposed algorithm can be achieved the best results and fully satisfy the slew constraints.

This Research is supported by National Natural Science Foundation of China ("Research on VLSI performance-driven multilayer routing under advanced Via-Pillar technology", 61877010; National Basic Research Program of China (973 Program) "Research on Key Applied Mathematical Theory and Method in Physical Design of LSI, 2011CB808000", "Research on Low-power Global Routing for Multiple Dynamic Supply Voltage Designs", 11501114; "Research on VLSI Multilayer Global Routing Algorithms in Non-Manhattan Architecture",11271002)