小 型 微 型 计 算 机 系 统
Journal of Chinese Computer Systems
DOI:10.20009/j.cnki.21-1106/TP.2024-0459
2026 年 2 月 第 2 期
Vol.47 No.2 2026

时序松弛约束下绕障 X 结构布线算法

鲁 任[1],朱予涵[1],刘耿耿[1,2,3]

[1](福州大学 计算机与大数据学院,福州 350116)
[2](大数据智能教育部工程研究中心,福州 350116)
[3](福建省网络计算与智能信息处理重点实验室,福州 350116)
E-mail:liugenggeng@fzu.edu.cn

摘 要:超大规模集成电路布线结果的优劣将直接影响芯片设计的质量.随着互连延迟逐渐成为芯片延迟的主要来源,在布线过程中进行时序分析变得愈发重要,同时,考虑到障碍在芯片布线中无法避免,以及 X 结构的引入可以更充分利用布线空间资源,因此,首次综合考虑时序松弛、障碍物以及更具互连优势的 X 结构,提出了一种时序松弛约束下绕障 X 结构布线算法.首先,为了避免重复地判断引脚间的连线是否经过障碍,提出一种高效的预处理策略,通过构建边障表来记录任意两引脚间的 4 种布线方式与所有障碍的关系.其次,为了解决构造 Steiner 最小树这一离散问题,将遗传算法与粒子群优化算法相结合,引入变异算子和交叉算子作为粒子群的离散更新方式.然后,提出了一种局部时序优化策略,通过优化布线半径达到优化最坏负松弛值的目的.接着,提出了一种有效的绕障策略,使布线在不经过障碍的前提下尽可能优化线长.最后,提出了一种路径精炼策略,选择布线资源共享程度最高的布线结构对原有的布线结构进行替换,进而达到优化线长的目标.实验结果表明,所提算法优化了布线的线长以及半径并在时序的关键性指标—最坏负松弛值上比相关算法分别优化了 59%、51% 和 40%,极大提高了电路的稳定性.

关 键 词:超大规模集成电路;Steiner 最小树;绕障;X 结构布线;时序松弛约束
中图分类号:TP301 **文献标识码**:A **文 章 编 号**:1000-1220(2026)02-0495-09

Obstacle-avoiding X-architecture Routing Algorithm Under Timing Slack Constraints

LU Ren[1],ZHU Yuhan[1],LIU Genggeng[1,2,3]

[1](College of Computer and Data Science,Fuzhou University,Fuzhou 350116,China)
[2](Engineering Research Center of Big Data Intelligence,Ministry of Education,Fuzhou 350116,China)
[3](Fujian Key Laboratory of Network Computing and Intelligent Information Processing,Fuzhou 350116,China)

Abstract:The results of routing in very large-scale integration will directly affect the quality of the chip design. As the interconnection delay gradually becomes the main source of chip delay,the timing analysis in the routing process becomes more and more important, taking into account the unavoidable obstacles in chip routing and the introduction of X-architecture to make better use of routing space resources. Therefore,for the first time,an obstacle-avoiding X-architecture routing algorithm under timing slack constraints is proposed,considering the timing slack,obstacles and the more interconnected X-architecture. First,in order to avoid repeatedly judging whether the connection between pins passes through obstacles,an efficient preprocessing strategy is proposed to record the four routing modes between any two pins with all obstacles by constructing an edge-barrier table. Second,in order to solve the discrete problem of constructing the Steiner minimum tree,the genetic algorithm is combined with the particle swarm optimization algorithm,and the mutation operator and the crossover operator are introduced as the discrete update method of the particle swarm. Third,a local timing optimization strategy is proposed to optimize the worst negative slack value by optimizing the routing radius. Fourth,an effective obstacle-avoiding strategy is proposed to optimize the wirelength as much as possible without passing through the obstacle. Finally,an effective path refining strategy is proposed,in which the routing structure with the highest degree of routing resource sharing is selected to replace the original routing structure,so as to achieve the goal of optimizing the wirelength. Experimental results show that the proposed algorithm optimizes the wirelength and radius of the routing,and optimizes the key index of the timing,the worst negative slack value, by 59%,51% and 40%,respectively,which greatly improves the stability of the circuit.

Keywords:very large-scale integration;Steiner minimum tree;obstacle-avoiding;X-architecture routing;timing slack

---

收稿日期:2024-11-20 收修改稿日期:2024-12-13 基金项目:国家自然科学基金项目(62372109)资助;福建省杰出青年科学基金项目(2023J06017)资助. 作者简介:**鲁 任**,女,2000 年生,硕士研究生,CCF 学生会员,研究方向为 EDA 算法研究;**朱予涵**,女,1998 年生,博士研究生,CCF 学生会员,研究方向为 EDA 算法研究、微流体生物芯片;**刘耿耿**(通信作者),男,1988 年生,博士,教授,CCF 杰出会员,研究方向为 EDA 算法研究、计算智能及其应用.

0 引 言

超大规模集成电路(Very Large-Scale Integration, VLSI)将千万个电子元件集于一个芯片中,使其具有对信息进行收集、加工和存储的功能[1]. 近年来,随着微电子技术的不断发展,VLSI 设计的集成度以及复杂性迅速增长[2]. 现代电子设计技术中引入了电子设计自动化技术(Electronic Design Automation, EDA)以解决日益复杂的 VLSI 设计问题. 物理设计是 VLSI 设计的重要环节之一,由于其具有高复杂性,被划分为 6 个阶段,包括划分、芯片规划、布局、时钟树综合、布线和时序收敛[3].

在 VLSI 物理设计的布线阶段,它的互连效果将直接影响最终的电路性能,布线方案的优劣会直接影响芯片的大小、运行速度、功耗以及可靠性[4]. 因此,布线阶段在整个集成电路设计流程中越发重要. 传统布线算法的目标大多是尽可能缩短布线线长[5,6],但由于互连时延对布线效果的影响越来越大,工业界逐渐意识到时序约束的重要性. 在传统的布线算法中,由于忽略了时序的影响,导致某些引脚获取信息的实际到达时间比其要求到达时间更大,从而违背了时序松弛约束,延长了 VLSI 物理设计的周期,增加了电路的设计成本. 因此,时序驱动的布线算法已经成为了非常重要的研究热点.

Steiner 最小树(Steiner Minimum Tree, SMT)是 VLSI 物理设计常用的模型之一. 对于一个多引脚线网,为求解其最优布线结构,最好的方法是为其构建一棵 SMT[7]. 在给定一组引脚的二维坐标后,SMT 通过在路径中增加一些 Steiner 点来连通所有引脚并使布线的总线长最小. 现如今,由于 VLSI 芯片设计中存在宏单元、预布线网等布线障碍物,在此基础上构造一棵高质量的绕障 Steiner 最小树(Obstacle-Avoidance Steiner Minimum Tree, OASMT)问题引起了人们的关注[8].

根据连线的结构,SMT 被分为曼哈顿结构 SMT 和非曼哈顿结构 SMT. 曼哈顿结构 SMT 也被称为直角 Steiner 最小树(Rectilinear Steiner Minimum Tree, RSMT). 非曼哈顿结构主要包括 X 结构和 Y 结构,根据 λ 几何定理[9],X 结构的连线有 0°、45°、90°和 135°. 与 RSMT 相比,X 结构 Steiner 最小树(X-architecture Steiner Minimum Tree, XSMT)可以更加充分地利用布线的空间资源,从而减少线长、互连延迟和芯片面积,极大地提高了芯片的性能.

SMT 的构造问题属于 NP 难问题[10],而以粒子群优化(Particle Swarm Optimization, PSO)算法为代表的群智能算法在解决 NP 难问题上展现出良好的应用前景[10-14]. 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. 粒子通过个体最优粒子和全局最优粒子更新自己的位置. PSO 算法经过适当的迭代次数可以得到优化问题的高质量解,而且在 VLSI 领域得到了良好的应用[13,14,19,37].

基于上述分析,为了解决存在障碍物和时延的芯片布线问题,同时考虑更有线长优化空间的 X 结构,本文提出了一种时序松弛约束下绕障 X 结构布线(Obstacle-Avoiding X-architecture Routing algorithm under Timing Slack Constraints, OAXR-TSC)算法. 本文主要贡献如下:

1)首次综合考虑了 X 布线结构、芯片中障碍物以及引脚间连线的时延对布线效果的影响,提出了一种有效的时序松弛约束下绕障 X 结构布线算法,使布线能够在满足绕障约束的基础上尽可能地优化最坏负松弛值(Worst Negative Slack, WNS)这一时序驱动布线问题的关键指标.

2)提出了一种综合考虑半径和线长的粒子群优化(Particle Swarm Optimization considering Radius and Wirelength, PSO-RW)算法. 将遗传算法与粒子群优化算法相结合,利用遗传算法的变异操作和交叉操作来实现粒子群优化算法的离散更新操作.

3)设计了局部时序优化策略以优化具有 WNS 的关键路径的线长. 遍历关键路径的所有连线,选择最小化线长和时延的布线结果,并允许牺牲少量的线长来增大引脚的时序松弛值,使半径和最坏负松弛值得到最大程度的优化.

设计了一种有效的绕障策略使所有引脚间的连线均避开障碍物. 若引脚间的连线穿过了障碍,则根据连线与障碍组相交的情况来选择伪 Steiner 点,通过引脚与伪 Steiner 点的连接使布线结构完全绕障,并且有效减少了布线空间资源的浪费.

1 相关工作

本节介绍与本文问题相关的 3 类工作的研究现状,分别是 SMT、OASMT 和时序驱动的 Steiner 最小树(Timing-Driven Steiner Minimum Tree, TDSMT).

1.1 SMT

SMT 的构造问题是布线阶段的一个关键性问题,决定了布线结果的质量. 根据连线的结构,将 SMT 分为 RSMT 和 XSMT.

近年来,针对构建 RSMT 的研究有很多. 文献[15]提出了一种基于人工智能的布线算法,可以提高整体布线性能并缩短设计时间. 文献[16]提出的 FLUTE 算法是一种非常快速、准确的 RSMT 算法. 通过预先计算出的查找表,可以得到小规模线网的最佳 RSMT. 对于大规模的网络,提出了一种网络分解技术来分解网络. 文献[17]通过分治法和深度优先搜索法获得 MST,进一步改进了 FLUTE 模型,显著减少了线长较小时的计算时间. 文献[18]的方法采用区域划分和聚类的方法来处理大规模网络,与 FLUTE 算法相比,大大提高了运行速度. 文献[19]中的算法采用梯度下降的方法对 PSO 算法进行局部搜索,并在 RSMT 构造中实现了布线成本的优化.

针对 XSMT 的构造问题,文献[20]提出了一种结合边替换和三角收缩的方式来构造 XSMT. 文献[21]针对多层布线问题提出了一种有效的多层 XSMT 算法. 文献[22]提出了一种基于蚁群算法的 XSMT 构造算法,可以有效减少线长和电容. 文献[23]提出了一种基于差分进化算法的 XSMT 构造算法,结合遗传算法来解决离散的 SMT 构造问题,从而得到了一个高质量的解决方案. 文献[24]首次考虑了能够有效防止信号失真的电压转换速率约束的 XSMT 构造算法,并取得同类工作中最佳的布线结果.

但仅仅构建 SMT 并不能解决芯片中存在障碍物的问题,也无法缓解互连线间时延对芯片稳定性的影响.

1.2 OASMT

由于芯片中存在预布线网、宏单元等障碍物的存在,

OASMT 问题得到了人们的重视,而 OASMT 问题可以被分为绕障直角 Steiner 最小树(Obstacle-avoiding Rectilinear Steiner Minimum Tree,OARSMT)和绕障 X 结构 Steiner 最小树(Obstacle-avoiding X-architecture Steiner Minimum Tree,OAXSMT).

对于 OARSMT 问题,文献[25]提出的 FOARS 算法首先将一组引脚划分为几个子集,然后使用一个考虑障碍的 FLUTE 算法来生成 OARSMT,最后通过合并这些子树来生成最终的 OARSMT. 文献[26]中的方法基于迷宫布线,采用斯坦纳点预选策略来指导多层 OARSMT 的构建,从而能够很好地权衡布线结果和运行时间. 文献[27]提出了一种基于高效满足性的伪布尔值算法来构造 OARSMT,其主要目标是优化布线线长. 文献[28]提出了一种基于区域间生成图的多层 OARSMT 构造算法,用较小的成本避免所有障碍.

对于 OAXSMT 问题,文献[8]提出了一种基于 PSO 的多层 OAXSMT 算法,这是解决这一问题的第一个工作. 文献[29]提出了一种启发式的 λ-OASMT 构造算法. 基于对障碍物的三角剖分算法,构造了一个全连接树,并通过区域组合将其嵌入到 λ-OASMT 中. λ-OASMT 的高效和准确性使其在布线阶段非常实用. 文献[30]提出了一种基于粒子群优化算法的 OAXSMT 构造算法,该算法在各种基准测试中都取得了良好的效果,并在合理的运行时间下获得了高质量的解. 文献[31]提出了一种快速 4 步启发式算法,在不过度牺牲质量的情况下,其运行速度得到了极大的提升. 文献[32]的算法中首先基于查找表来构建三维无障碍物的 MST,再采用 3 种基于投影的避障策略,最后结合 2 种有效的精炼技术将三维 MST 转化为多层 OAXSMT.

OASMT 虽然考虑了芯片中障碍物的存在,但尚未考虑到时延这一影响布线质量的重要因素.

1.3　TDSMT

由于引脚间互连线的时延对布线效果的影响越来越大,在布线阶段考虑时序约束的研究随之增多. 根据连线结构的不同,TDSMT 可以被分为时序驱动的直角 Steiner 最小树(Timing-Driven Rectilinear Steiner Minimum Tree,TDRSMT)以及时序驱动的 X 结构 Steiner 最小树(Timing-Driven X-architecture Steiner Minimum Tree,TDXSMT).

文献[33]提出了一种基于新的绕障生成图的布线算法来有效地构造一个可以优化布线时延的 OARSMT. 文献[34]研究了具有预缓冲和旋转约束的时序驱动的 RSMT 构造问题,它可以在满足用户指定的时间约束的同时回收障碍上的"浪费"布线资源.

在给定任意线网的 RSMT 的情况下,文献[35]提出了一种有效的基于变换的 TOST 算法,通过对 X 结构距离的计算、Steiner 点再分配和路径重构,构造了一个 TDXSMT. 文献[36]提出了一种针对矩形和非矩形障碍物的 TDXSMT 算法,可以处理矩形和非矩形障碍物,减少从源点到汇点的最小延迟. 文献[37]提出了一种基于多目标 PSO 和 Elmore 延迟模型,构建了一棵高质量的时序驱动 X 结构 Steiner 最小树. 文献[38]提出了一种基于社会学习多目标 PSO 的有效算法来构造 TDXSMT.

关于 TDSMT 问题的研究中,文献[35-38]均未考虑芯片中障碍物的存在,而文献[33,34]虽然考虑了绕障约束,但其连线方式为曼哈顿结构,尚未利用 X 结构在布线中展现的优良特性.

综上所述,SMT、OASMT 和部分 TDSMT 的研究均未系统地考虑到时延和障碍物对布线结果的影响. 尽管文献[33]考虑到了两种约束,但其使用直角结构,并未涉及 X 结构,因此,本文在布线问题中,首次综合考虑到时序松弛、障碍物以及更具互连优势的 X 结构,并提出了 OAXR-TSC 算法.

2　问题模型

2.1　连线方式的定义

XSMT 的连线方向有 0°、45°、90°和 135°,如图 1 所示,将每 2 个引脚之间的连线分为以下 4 种方式:

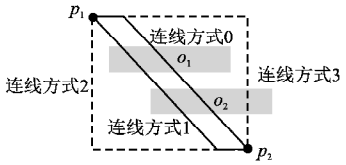


图 1　2 引脚的 4 种连线方式

Fig. 1　Four interconnection modes with two pins

定义 1. 连线方式 0. 从第 1 个引脚 p_1 开始用水平或垂直方向的连线连接到 Steiner 点 s,再从点 s 用 45°或 135°的斜线连接到第 2 个引脚 p_2.

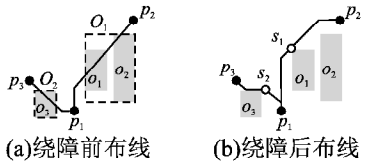
定义 2. 连线方式 1. 从第 1 个引脚 p_1 开始用 45°或 135°的斜线连接到 Steiner 点 s,再从点 s 用水平或垂直方向的连线连接到第 2 个引脚 p_2.

定义 3. 连线方式 2. 从第 1 个引脚 p_1 开始用垂直方向的连线连接到 Steiner 点 s,再从点 s 用水平方向的连线连接到第 2 个引脚 p_2.

定义 4. 连线方式 3. 从第 1 个引脚 p_1 开始用水平方向的连线连接到 Steiner 点 s,再从点 s 用垂直方向的连线连接到第 2 个引脚 p_2.

2.2　绕障

定义 5. 引脚. 由用户给定,可以位于障碍外部或者边界上但不能位于障碍的内部,如图 2(a)中的 p_1、p_2 和 p_3.



(a)绕障前布线　(b)绕障后布线

图 2　绕障示例

Fig. 2　Example of obstacle-avoiding

定义 6. 伪 Steiner 点. 除了用户给定的引脚之外的新增的连接点都是伪 Steiner 点,如图 2(b)中的 s_1 和 s_2.

定义 7. 源点与汇点. 源点为用户给定的第 1 个引脚,如图 2(a)中的 p_1,汇点为连接所有引脚的生成树的叶子节点,如图 2(a)中的 p_2 和 p_3.

定义 8. 总线长. 总线长指各个引脚之间的连线线长的总和.

定义 9. 半径. 半径指从源点到各个汇点之间的最长距离.

定义 10. 障碍. 所有障碍都为矩形,且各个障碍之间不能重叠,如图 2(a)中的 o_1、o_2 和 o_3.

定义 11. 障碍组边界. 一个障碍组是 2 个引脚之间连线经过一个或多个障碍的集合,障碍组边界是包含障碍组的最小矩形的边界,如图 2(a)中的 O_1 和 O_2.

当连线经过障碍时,通过增加伪 Steiner 点使连线绕障,如图 2(b)所示,s_1 和 s_2 为伪 Steiner 点,图 2(b)为图 3(a)的绕障结果.

2.3 时序松弛

本文基于静态时序分析(Static Timing Analysis, STA),其关键指标是时序松弛,当松弛值非负时,表示其对应节点的时序约束达到要求. 本文旨在进行时序优化以增加负松弛值,从而提高电路的可靠性.

本文利用 Elmore 时延模型来计算连线的时延[39]. 在该模型中,将连线建模成均匀分布的 Π 型电容电阻电路[40],其中一半电路上的电容位于上游节点,另一半电路上的电容位于下游节点.

引脚 p_i 与 p_j 之间的连线的时延计算公式为:

$$E(p_i,p_j)=\sum_{p_t\in Path(p_i,p_j)} r\times C_{down}(p_t) \tag{1}$$

其中,$Path(p_i,p_j)$表示引脚 p_i 与 p_j 之间的路径,r 表示连线的电阻,$C_{down}(p_t)$表示引脚 p_t 的下游电容,如公式(2)所示:

$$C_{down}(p_i)=c_i+\sum_{p_n\in succ(p_i)} C_{down}(p_n) \tag{2}$$

其中,c_i 表示 p_i 处的电容,$succ(p_i)$表示 p_i 的后继节点的集合.

某一引脚 p_i 的时序松弛值的计算方式如公式(3)所示:

$$slack(p_i)=rat(p_i)-aat(p_i) \tag{3}$$

其中,$rat(p_i)$为引脚 p_i 的要求到达时间,$aat(p_i)$为引脚 p_i 的实际到达时间,分别如公式(4)、公式(5)所示:

$$rat(p_i)=\max_{p_n\in suc(p_i)}(rat(p_n)-E(p_i,p_j)) \tag{4}$$

$$aat(p_i)=aat(p_{pre})+E(p_i,p_j) \tag{5}$$

其中,$suc(p_i)$表示直接与 p_i 相连的 p_i 的后继节点的集合,p_{pre}表示 p_i 的前一个引脚.

WNS 表示最坏负松弛值,是除源点外,所有汇点的负松弛值中的最小值,如公式(6)所示:

$$WNS=\min_{p_i\in P,i\neq 1}(slack(p_i)) \tag{6}$$

其中,P 为所有引脚的集合.

2.4 问题描述

由用户给定引脚数量 n,引脚 $P=\{p_1,p_2,\cdots,p_n\}$的坐标,障碍数量 m,障碍 $O=\{o_1,o_2,\cdots,o_m\}$的左下角坐标和右上角坐标以及汇点的要求到达时间. 其中,源点 p_1 的实际到达时间为 0. 本文的目标是构建一棵 XSMT 来连接所有引脚,并使连线完全绕障,在此基础上优化总线长和 WNS.

考虑到本文算法主要的优化目标是 WNS,而根据 Elmore 时延模型的计算公式(1)~公式(6)可以得知布线的半径与 WNS 呈正相关,因此本文算法需要着重优化布线的半径并允许牺牲少量线长来换取半径以及 WNS 的极大优化.

3 时序松弛约束下绕障 X 结构布线算法

本文提出了一种有效的时序松弛约束下绕障 X 结构布线算法,主要由 5 个部分组成,分别是障碍预处理与引脚对编码、PSO-RW 算法、时序优化策略、绕障策略以及路径精炼策略.

3.1 障碍预处理与引脚对编码

3.1.1 边障表的构建

由于后续策略都需要频繁地使用引脚间连线与障碍物的相交情况,所以本文算法构建边障表来记录这些信息,以便后续策略不用频繁重复判断连线与障碍物的相交情况,加快算法的运行时间.

表 1 图 1 对应的边障表

Table 1 Figure 1 corresponds to the wire-obstacle table

引脚 p_1 和 p_2 之间的连线方式	连线经过的障碍物编号
连线方式 0	o_1,o_2
连线方式 1	o_1,o_2
连线方式 2	$\emptyset$
连线方式 3	o_2

已知所有引脚坐标和障碍物坐标,且每 2 个引脚之间有 4 种连线方式,构建边障表需要判断每 2 个引脚之间的不同连线方式是否与障碍相交,并记录与之相交的障碍物的编号. 表 1 为根据图 1 生成的边障表.

3.1.2 Prim-Dijkstra 树的构建与编码

由于本算法需要构建综合考虑线长和半径的 X 结构 Steiner 树,因此利用 PD 算法来构建初始的生成树,以避免造成半径过长导致其中引脚的最坏负松弛值过小或者为了优化各个引脚的最坏负松弛值而过度牺牲线长的结果.

完成 PD 树的构造后,需要对 PD 树进行编码. 遍历 PD 树每 2 个引脚之间的连线,并将每条连线用 3 个数字保存,其中这 3 个数字分别代表 2 个引脚的编号和它们的连线方式,并在最后 1 位数中给出 PD 树的适应值. 假设一个 PD 树有 n 个引脚、$n-1$ 条边,由于每条边需要 3 个数字且需要在最后加上 PD 树的适应值,则需要 $3(n-1)+1$ 个数字来保存 PD 树的信息.

3.2 PSO-RW 算法

PSO 算法起初是为了解决连续性问题而提出的,但由于 Steiner 树的构建问题属于离散问题,需要将 PSO 操作算子离散化,因此本文采用一种基于遗传算子的离散位置更新方式,结合并查集的思想,通过变异操作和交叉操作使粒子可以通过个体变化和种群之间信息交互来实现粒子种群的搜索和更新,从而能够有效解决构建时序松弛约束下绕障 X 结构 Steiner 树这一离散问题. 公式(7)为粒子的更新公式:

$$S_i^t=C_2(C_1(V(S_i^t,w),r_1),r_2) \tag{7}$$

其中,S_i^t 表示第 t 代,第 i 个粒子,C_2 为全局交叉操作,C_1 为个体交叉操作,V 为个体变异操作,r_1 为个体交叉概率,r_2 为全局交叉概率,w 为个体变异概率.

3.2.1 种群初始化

粒子种群的初始化是设定种群数量 $popsize$、最大迭代次数 $count$、加速因子和权重的数值,并初始化每个粒子的历史最优位置和粒子群的全局最优位置[41]. 同时,本文根据一种线性增长公式使 w、r_1 和 r_2[42] 随迭代次数自适应改变. 其计算公式(8)~公式(10):

$$w_t=w_s-\frac{w_s-w_e}{count}\times t \tag{8}$$

$$r_{1t} = r_{1s} - \frac{r_{1s} - r_{1e}}{count} \times t \tag{9}$$

$$r_{2t} = r_{2s} - \frac{r_{2s} - r_{2e}}{count} \times t \tag{10}$$

其中,w_t、r_{1t}和 r_{2t}表示第 t 代的 w、r_1 和 r_2;w_s、r_{1s}和 r_{2s}表示 w、r_1 和 r_2 的初始值,w_e、r_{1e}和 r_{2e}表示 w、r_1 和 r_2 的最终值,$count$ 为粒子种群的迭代总数,t 为粒子种群的迭代次数.

3.2.2　个体变异操作

为了避免陷入局部最优,本文利用个体变异操作来扩大粒子群搜索范围,从而找到全局最优解.

变异操作如公式(11)所示:

$$Q_i^t = V(S_i^{t-1}, w) = \begin{cases} V(S_i^{t-1}), c_1 < w \\ S_i^{t-1}, c_1 \geqslant w \end{cases} \tag{11}$$

其中,V 表示变异操作,c_1 表示[0,1)区间的随机数.

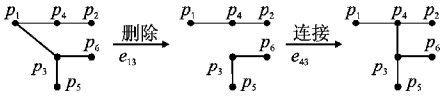


图 3　个体变异操作

Fig. 3　Example of the mutation operator

变异操作的基本思想为:随机删除生成树的一条边后,利用并查集将 2 棵子树的引脚并分别放入 2 个集合,再从 2 个集合中随机选出 2 个引脚,并将其相连. 如图 3 所示,随机删除了引脚 p_1 与 p_3 之间的连线,然后利用并查集从 2 个集合中随机选出引脚 p_4 和 p_3 并将其连接.

3.2.3　交叉操作

为了增加种群的多样性并加快粒子群的收敛速度,本文设计了个体交叉操作使粒子在局部范围内搜索和全局交叉操作使粒子群向全局最优粒子收敛.

个体交叉操作是将粒子与其本身的历史最优位置交叉,如公式(12)所示:

$$W_i^t = C_1(Q_i^t, r_1) = \begin{cases} C_1(Q_i^t), c_2 < r_1 \\ Q_i^t, c_2 \geqslant r_1 \end{cases} \tag{12}$$

其中,C_1 表示个体交叉操作,c_2 表示[0,1)区间的随机数.

全局交叉操作是将粒子与种群的最优粒子交叉,加快粒子群的收敛速度,如公式(13)所示:

$$S_i^t = C_2(W_i^t, r_2) = \begin{cases} C_2(W_i^t), c_3 < r_2 \\ W_i^t, c_3 \geqslant r_2 \end{cases} \tag{13}$$

其中,C_2 表示个体交叉操作,c_3 表示[0,1)区间的随机数.

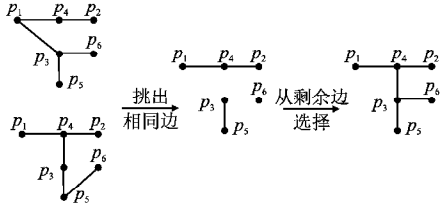


图 4　交叉操作

Fig. 4　Example of the crossover operator

交叉操作的原理如图 4 所示. 首先将 2 棵生成树中相同的连线放入一个集合作为下一代粒子的部分连线,接着将 2 个粒子中的不同连线放入另一个集合,利用并查集随机从第 2 个集合里挑选连线作为下一代粒子的剩余连线,直到构成一棵完整的生成树.

3.2.4　适应度的计算

为了构建综合考虑半径和线长的 X 结构 Steiner 树,本文的适应度计算以优化线长和半径为目标,如公式(14)所示:

$$fitness = (1-d) \times WL + d \times PL \tag{14}$$

其中,WL 表示线长,PL 表示半径,d 是一个均衡线长和半径的平衡因子.

由于障碍的存在,连线若经过障碍一定会导致线长增大,因此本文考虑在线长的计算中加入障碍组的部分周长,从而减少绕障的线长. 线长计算如公式(15)所示:

$$WL_{e_{se}} = \max(|x_e - x_s|, |y_e - y_s|) + (\sqrt{2} - 1) \times \min(|x_e - x_s|, |y_e - y_s|) + CL_{e_{se}} \tag{15}$$

其中,$WL_{e_{se}}$是边 e_{se}的线长,(x_e, y_e)和(x_s, y_s)分别为边 e_{se}的 2 个顶点坐标,$CL_{e_{se}}$为边 e_{se}穿过的障碍组的$\frac{1}{4}$周长,如公式(16)所示:

$$CL_{e_{se}} = \frac{y_{b\max} - y_{b\min} + x_{b\max} - x_{b\min}}{2} \tag{16}$$

其中,$(x_{b\min}, y_{b\min})$和$(x_{b\max}, y_{b\max})$分别表示连线 e_{se}经过的障碍组的左下角坐标和右上角坐标.

WL 是布线的总线长,其计算方式如公式(17)所示:

$$WL = \left(\sum_{e_{se} \in S_i^t} WL_{e_{se}}\right) - WL_{re} \tag{17}$$

其中,WL_{re}是连线中重叠边的线长.

PL 是布线的半径,其计算方式如公式(18)所示:

$$PL = \max_{p_n \in leaf(S_i^t)} \left(\sum_{e_{se} \in l_{p_n}} WL_{e_{se}}\right) \tag{18}$$

其中,$leaf(S_i^t)$表示生成树 S_i^t 的叶子节点,l_{p_n}表示从源点到引脚 p_n 的路径.

3.3　时序优化策略

考虑到时序的关键性指标—WNS 与布线树的半径有关,半径越长可能导致处于半径上引脚的 WNS 越小. 同时,由于优化半径可能导致布线树的总线长变大,因此为了进一步优化布线树的 WNS,本文在 PSO-RW 算法生成的最优粒子的基础上,设计了时序优化策略,以牺牲少量线长的代价来换取 WNS 的极大提升.

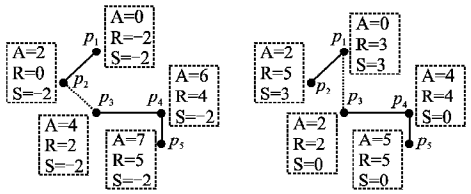
首先,构建最优更新数组来保存更新的连线以及更新后的线长和 WNS,将 PSO-RW 算法生成的最优粒子及其线长和 WNS 作为最优更新数组的初始值. 找到最优粒子中具有最坏负松弛值的的引脚,并确定该引脚所在的从源点到汇点的最长路径. 假设在该条路径上,从源点到叶子节点之间的引脚分别为 $p_1, p_2, \cdots, p_l$. 遍历该条路径上除源点以及源点的下一个引脚外的所有引脚 p_i(即 $i \neq 1, 2$),删除 p_i 与 p_{i-1}的连线,并将 p_i 与 $p_1, p_2, \cdots, p_{i-2}$逐一相连. 将每次更新后的线长和 WNS 与最优更新的线长和 WNS 进行对比,将得到以下 4 种结果之一:

1)线长以及 WNS 都得到优化,则将其替换为最优更新.

2)线长得到优化,WNS 未被优化. 若 WNS 没有发生劣化,则将其替换为最优更新,反之跳过该更新.

3)线长未得到优化,WNS 被优化,则将其替换为最优更新.

4)线长和 WNS 都未得到优化,则跳过该更新.



(a)时序优化前布线 (b)时序优化后布线

图 5 时序优化策略

Fig. 5 Example of timing optimization strategy

本文的时序优化策略可以对半径的所有连线进行遍历,找到最小化线长和半径的布线结构. 如图 5 所示,引脚旁边的 A、R、S 分别代表该引脚信号的实际到达时间、要求到达时间和松弛值. 而在图 5(a)的布线中,半径上的引脚均违反了时序约束,其 WNS 均为 -2. 在对半径进行优化后,最终的生成树结构如图 5(b)所示,该布线结构使所有引脚的松弛值均为非负数,满足了时序约束条件.

3.4 绕障策略

由于时序优化后的部分连线穿过了障碍,为了满足绕障约束,本文设计了一种绕障策略使所有连线完全绕障. 首先,通过边障表判断连线是否经过障碍,若经过障碍,则删除该条连线,选取伪 Steiner 点,并增添新的连线;若未经过障碍,则跳过该条连线,继续判断下一条连线是否经过障碍. 最后根据新的生成树更新边障表,循环以上操作直至所有连线都不经过障碍,则绕障结束.

3.4.1 连线方式的选取

当连线经过障碍时,删除该条连线,并分别用 4 种连线方式连接 2 个引脚,并通过边障表判断是否经过障碍,此时有以下 3 种情况:

1)若引脚通过连线方式 0 或连线方式 1 相连时未经过障碍,则用连线方式 0 或连线方式 1 连接 2 个引脚;

2)若引脚通过连线方式 0 和连线方式 1 相连时都经过障碍但通过连线方式 2 或连线方式 3 相连时未经过障碍,则对比引脚通过连线方式 2 和连线方式 3 相连时经过的障碍组的半周长,用半周长最小的连线连线方式连接 2 个引脚,并进入伪 Steiner 点的选取;

3)若引脚通过连线方式 0、连线方式 1、连线方式 2 和连线方式 3 相连时都经过障碍,则对比引脚通过连线方式 0、连线方式 1、连线方式 2 和连线方式 3 相连时经过的障碍组的半周长,用半周长最小的连线方式连接 2 个引脚,并进入伪 Steiner 点的选取.

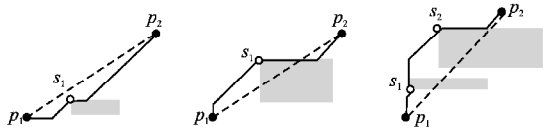
3.4.2 伪 Steiner 点的选取

确定 2 个引脚之间的连线所经过的障碍组的左下角和右上角的坐标,将 2 个引脚直接相连,连线与障碍组之间的联系有以下 3 种:

1)连线未与障碍组相交时,选择与连线的垂直距离最小的障碍组内的障碍顶角作为伪 Steiner 点.

2)连线与障碍组相邻的两边相交时,若连线与障碍组的左边和上边相交,则选取障碍组的左上角为伪 Steiner 点;若连线与障碍组的左边与下边相交,则选取障碍组的左下角为伪 Steiner 点,以此类推. 若上述选取的伪 Steiner 点不属于障碍组内的障碍的顶角,则选择距离该伪 Steiner 点最近的障碍组内的障碍的顶角来代替删除该伪 Steiner 点.

3)连线与障碍组相对的两边相交时,障碍组被连线划分为 2 个部分,选择周长较小的那一部分的 2 个顶角作为伪 Steiner 点. 若选取的伪 Steiner 点不属于障碍组内的障碍的顶角,则选择距离该伪 Steiner 点最近的障碍组内的障碍的顶角来代替删除该伪 Steiner 点.



(a)连线未与障碍组相交时 (b)连线与障碍组相邻的两边相交时 (c)连线与障碍组相对的两边相交时

图 6 伪 Steiner 点的选取

Fig. 6 Selection of pseudo-Steiner points

如图 6(a)所示,p_1 与 p_2 之间的连线未经过障碍,则选取距离连线最近的障碍角点 s_1 为伪 Steiner 点;图 6(b)中 p_1 与 p_2 之间的连线经过障碍组相邻的两边,则选取障碍的左上角 s_1 为伪 Steiner 点;图 6(c)中 p_1 与 p_2 之间的连线经过障碍组相对的两边,则选取障碍组的左上角和左下角为伪 Steiner 点,但障碍组的左上角并非障碍物的角点,因此选择距离障碍组左上角最近的障碍物角点 s_2 为伪 Steiner 点.

3.5 路径精炼策略

在满足避障约束和时序松弛约束后,一些路径在线长上仍可以进一步优化. 本文通过路径精炼策略来增加连线的共享边长度,从而得到拥有更短线长的布线结构. 该策略的实现方式如下:

首先,计算每个引脚的度,并记录与该引脚相连的所有引脚. 如果引脚的度为 D,则列出引脚的 4^D 种布线结构,并选择满足绕障约束并具有最小线长和延迟的连线作为最佳连线. 其次,计算每个布线的共享边长度 WL_{re},并根据共享边的长度按降序排列. 最后,将引脚的初始布线结构替换为其最优布线结构. 遍历结束后,便得到了解决 OAXR-TSC 问题的最终布线结构.

4 实验结果

本文算法用 C/C++语言实现,所有的实验均在1.8GHz CPU 以及 8G 内存的 PC 上单处理器单进程完成,并以避障问题的基准电路作为测试用例,其数据来自同步系统的工业测试用例[43]. 表 2 列出了各测试用例的引脚数量和障碍物数量. 其中,测试用例的引脚数量以及障碍物数量均在 10 ~

表 2 测试样例

Table 2 Test samples

测试样例	引脚数	障碍数	测试样例	引脚数	障碍数
RC01	10	10	RC07	200	500
RC02	30	10	RC08	200	800
RC03	50	10	RC09	200	1000
RC04	70	9	RC10	500	100
RC05	100	10	RC11	1000	100
RC06	100	500			

1000 内,此外,在运行时间充足的情况下,本文算法可以应用于更大规模的测试用例. 表 3 列出了本文的各项参数设置. 本文在严格松弛条件下进行实验,即给定的要求到达时间使所有引脚的松弛值都为负.

表 3　参数信息

Table 3　Parameter information

参数值	d_s	$r_0(w/\mu m)$	$c_0(fF/\mu m)$	w_s	w_e
	0.0075	0.118	0.95	0.40	0.05
参数值	d_e	t_{1s}	t_{1e}	*popsize*	*count*
	0.95	0.90	0.15	100	100

4.1　策略的有效性验证

4.1.1　PSO-RW 算法的有效性验证

为了验证 PSO-RW 算法的有效性,本文在严格松弛条件下,比较了基于不同的更新策略下最终布线树的线长、半径以及 WNS. 表 4 为实验对比的结果,如表 4 所示,改进前的算法未基于遗传算子且适应度值的计算未考虑障碍物的存在,而改进后的算法为本文设计算法. 实验结果证明 PSO-RW 算法相较于改进前可以使线长、半径及 WNS 分别平均优化了 2%、11% 和 8%. 这是由于通过变异操作和交叉操作可以使粒子通过个体变化和种群之间信息交互来实现粒子种群的搜索和更新,从而能够有效解决构建时序松弛约束下绕障 X 结构 Steiner 树这一离散问题. 同时,在适应度值的计算中考虑到障碍物对布线线长的影响,因此对经过障碍的连线增加与经过障碍组周长呈正比的惩罚因子,使布线尽可能少穿过障碍或穿过更小的障碍组来达到减少最终布线线长以及优化半径的目的.

表 4　在严格松弛条件下 PSO 适应度有无考虑障碍物前后对比

Table 4　Fitness of PSO is compared before and after considering obstacles under strict slack conditions

测试样例	改进前			改进后		
	线长	半径	WNS	线长	半径	WNS
RC01	24326	20852	-28459	24326	20852	-28459
RC02	36856	13416	-77018	36765	13416	-76991
RC03	49618	22407	-184529	49463	22407	-179353
RC04	53473	24773	-381385	53291	24220	-378627
RC05	70336	20996	-206181	69762	20324	-205333
RC06	75709	26688	-138101	74435	24840	-136485
RC07	101866	22659	-372308	100664	14182	-187682
RC08	105256	27965	-740100	101927	17540	-244384
RC09	101916	28295	-534118	91642	18812	-306656
RC10	155879	30055	-1312700	155457	30055	-1278691
RC11	224136	51588	-7744300	222799	51588	-7737242
平均值	90852	26336	-1065382	89139	23476	-978173
比率	1.00	1.00	1.00	0.98	0.89	0.92

4.1.2　时序优化策略的有效性验证

为了证明时序优化策略的有效性,本文在严格松弛条件下,比较了有无时序优化策略前后的线长、半径以及 WNS. 实验结果如表 5 所示,时序优化策略在牺牲少量线长的同时使布线的半径及 WNS 分别优化了 19% 和 17%. 由于布线的 WNS 与半径息息相关,这是因为半径越长的布线,处于半径中引脚的实际到达时间也会越大,而引脚的要求到达时间与引脚的其它下游节点有关,从而导致半径中引脚的松弛值会随着半径的增加而更劣. 因此为了优化 WNS,就需要优化布线的半径长度. 而优化半径可能导致布线树的总线长变大,为了不过度牺牲线长代价,本文的时序优化策略选择了平衡线长和半径的布线结果,使得算法可以利用少量线长换取 WNS 得到较大的提升.

表 5　在严格松弛条件下有无时序优化算法前后对比

Table 5　Comparison before and after timing optimization algorithm under strict slack conditions

测试样例	改进前			改进后		
	线长	半径	WNS	线长	半径	WNS
RC01	24326	20852	-28459	25285	15437	-12799
RC02	36856	13416	-77018	38062	13031	-66935
RC03	49618	22407	-184529	49031	22407	-149123
RC04	53473	24773	-381385	53687	20401	-281112
RC05	70336	20996	-206181	71778	14420	-186234
RC06	75709	26688	-138101	74823	24840	-136550
RC07	101866	22659	-372308	103665	20611	-297684
RC08	105256	27965	-740100	105864	23415	-718384
RC09	101916	28295	-534118	102642	18792	-407412
RC10	155879	30055	-1312700	165457	21957	-978113
RC11	224136	51588	-7744300	226816	40498	-6477242
平均值	90852	26336	-1065382	92465	21439	-882872
比率	1.00	1.00	1.00	1.02	0.81	0.83

4.1.3　绕障策略的有效性验证

本文的绕障策略可以使布线完全绕障,同时为了证明绕障策略有效减少了布线线长,本文在严格松弛条件下,比较了改进前后的绕障策略的布线线长、半径以及 WNS. 实验结果如表 6 所示,改进前的绕障策略中连线方式仅选择方式 0 或方式 1,伪 Steiner 点的选取仅选择距离两引脚间线段最近的一个障碍组端点为伪 Steiner 点. 实验结果证明改进后的绕障策略可以使线长、半径及 WNS 分别优化了 3%、3% 和 9%. 这是因为本文绕障策略会在优先选取线长更小的连线方式 0 或连线方式 1 的基础上选择尽可能少穿过障碍或穿过更小障碍组的连线方式,同时,本文中伪 Steiner 点的选取方式可以生成使绕障线长更小的伪 Steiner 点,并进一步利用障碍组内的

表 6　在严格松弛条件下有无选取伪 Steiner 点的绕障策略前后对比

Table 6　Comparison the obstacle-avoiding strategy with or without pseudo-Steiner points under strict slack conditions

测试样例	改进前			改进后		
	线长	半径	WNS	线长	半径	WNS
RC01	24326	20852	-28459	24326	20852	-28459
RC02	36856	13416	-77018	36065	13031	-70532
RC03	49618	22407	-184529	49519	22407	-179352
RC04	53473	24773	-381385	53387	24773	-380663
RC05	70336	20996	-206181	68285	20115	-195510
RC06	75709	26688	-138101	72128	26008	-126564
RC07	101866	22659	-372308	101803	22659	-369682
RC08	105256	27965	-740100	105252	27965	-737212
RC09	101916	28295	-534118	89880	25230	-391254
RC10	155879	30055	-1312700	148783	26918	-964959
RC11	224136	51588	-7744300	223799	51431	-7201100
平均值	90852	26336	-1065382	88476	25581	-967753
比率	1.00	1.00	1.00	0.97	0.97	0.91

布线资源,从而达到优化布线线长的目的.

4.2 本文算法的有效性验证

为了验证本文算法的有效性,本文将 OAXR-TSC 算法与的 OAXSMT 算法,实验结果如表 7 所示,本文的 OAXR-TSC 算法相较于文献[31],在布线线长上平均优化了 6%,在半径上平均优化了 41% 以及在 WNS 上平均分别优化了 59%. 文献[35]是 TDRSMT 算法,实验结果如表 8 所示,本文算法相较于文献[35],在布线线长上平均优化了 2%,在半径上平均优化了 36% 以及在 WNS 上平均分别优化了 40%. 文献[38]是 TDXSMT 算法,实验结果如表 9 所示,尽管本文算法与文献[38]线长相当,但在半径和 WNS 上平均分别优化了 12% 和 51%.

表 7 在严格松弛条件下本文算法与文献[31]的对比

Table 7 Comparison of the proposed algorithm with the literature [31] under strict slack conditions

测试样例	文献[31]			本文算法		
	线长	半径	WNS	线长	半径	WNS
RC01	24931	23278	-105927	25775	15437	-13925
RC02	41422	15208	-146759	36733	13343	-68825
RC03	54103	24129	-502782	49461	22407	-170334
RC04	60228	26910	-515282	53100	20265	-336868
RC05	73815	39416	-1379740	68997	15823	-189333
RC06	80049	30963	-673490	75012	24840	-135673
RC07	108530	25073	-804080	101473	14040	-179452
RC08	118721	30730	-1406950	104804	11812	-294322
RC09	114964	32691	-1575410	101966	17182	-301171
RC10	158140	33614	-2179700	155121	15957	-795069
RC11	222539	55511	-9128020	223005	28498	-5037130
平均值	96131	30684	-1674376	90496	18146	-683827
比率	1.00	1.00	1.00	0.94	0.59	0.41

表 8 在严格松弛条件下本文算法与文献[35]的对比

Table 8 Comparison of the proposed algorithm with the literature [35] under strict slack conditions

测试样例	文献[35]			本文算法		
	线长	半径	WNS	线长	半径	WNS
RC01	25193	21298	-40929	25775	15437	-13925
RC02	38508	13832	-67704	36733	13343	-68825
RC03	49987	24733	-282917	49461	22407	-170334
RC04	54951	26116	-390061	53100	20265	-336868
RC05	74706	36435	-220526	68997	15823	-189333
RC06	76217	28008	-319961	75012	24840	-135673
RC07	103679	23657	-496892	101473	14040	-179452
RC08	105908	27965	-1108350	104804	11812	-294322
RC09	102339	28295	-614755	101966	17182	-301171
RC10	157281	30459	-1324290	155121	15957	-795069
RC11	226065	51645	-7895750	223005	28498	-5037130
平均值	92258	28404	-1160194	90496	18146	-683827
比率	1.00	1.00	1.00	0.98	0.64	0.60

表 9 在严格松弛条件下本文算法与文献[38]的对比

Table 9 Comparison of the proposed algorithm with the literature [38] under strict slack conditions

测试样例	文献[38]			本文算法		
	线长	半径	WNS	线长	半径	WNS
RC01	25231	16735	-71482	25775	15437	-13925
RC02	36129	12582	-192810	36733	13343	-68825
RC03	49859	18373	-713206	49461	22407	-170334
RC04	52837	24773	-627843	53100	20265	-336868
RC05	69720	17267	-361326	68997	15823	-189333
RC06	73506	25774	-878059	75012	24840	-135673
RC07	100215	17625	-865628	101473	14040	-179452
RC08	105166	21627	-1256290	104804	11812	-294322
RC09	100920	23252	-2138900	101966	17182	-301171
RC10	156346	20923	-2438500	155121	15957	-795069
RC11	221823	26758	-5775470	223005	28498	-5037130
平均值	90159	20517	-1392683	90496	18146	-683827
比率	1.00	1.00	1.00	1.00	0.88	0.49

5 总 结

由于考虑时序以及障碍物更贴近实际芯片的设计情况,而 X 结构可以进一步优化线长这一重要指标,因此本文提出了 OAXR-TSC 算法. 首先,根据给定的引脚和障碍物信息进行预处理以加快算法运行速度. 其次,提出了 PSO-RW 算法,在粒子群的迭代过程中综合考虑了线长和半径,有效减少了布线的线长和时延. 在此基础上,为了进一步优化布线结果的 WNS,本文设计了一种局部时序优化策略,对具有 WNS 的引脚所在的最长路径进行优化,并综合考虑线长和 WNS 来更新布线结果,使布线结构尽可能的满足时序松弛约束. 然后,为了使布线满足绕障约束,提出了一种有效的绕障策略,能够充分地利用布线空间资源,尽可能地减少线长和半径. 实验结果表明,本文算法很好权衡线长和半径的关系,有效优化了半径,从而最终带来时序最重要的指标 WNS 的显著优化.

未来的工作包括:1)进一步考虑可靠性、电流密度、噪声干扰或功耗等因素对布线的影响;2)在维持布线质量的同时,提高算法的运行效率.

References:

[1] Andrew B, Jens L, Igor L, et al. VLSI physical design: from graph partitioning to timing closure[M]. Dordrecht: Springer, 2011.

[2] Lavanyashree B, Jamuna S. Design of fault injection technique for VLSI digital circuits[C]//IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology, 2017: 1601-1605.

[3] WANG J Y, ZHU Y H, ZHOU R P, et al. X-architecture steiner minimum tree algorithm based on dynamic particle swarm optimization[J]. Computer Engineering, 2024, 50(9): 226-234.

[4] Jia Y, Mei Y, Zhang M. A bilevel ant colony optimization algorithm for capacitated electric vehicle routing problem[J]. IEEE Transactions on Cybernetics, 2021, 52(10): 10855-10868.

[5] Wang J, Weng T, Zhang Q. A two-stage multiobjective evolutionary algorithm for multiobjective multidepot vehicle routing problem with time windows[J]. IEEE Transactions on Cybernetics, 2018, 49(7): 2467-2478.

[6] Xiao J, Zhang T, Du J, et al. An evolutionary multiobjective route grouping-based heuristic algorithm for large-scale capacitated vehicle routing problems[J]. IEEE Transactions on Cybernetics, 2019, 51(8): 4173-4186.

[7] Tang H, Liu G, Chen X, et al. A survey on Steiner tree construction and global routing for VLSI design[J]. IEEE Access, 2020, 8: 68593-68622.

[8] Liu G, Huang X, Guo W, et al. Multilayer obstacle-avoiding X-architecture Steiner minimal tree construction based on particle swarm optimization[J]. IEEE Transactions on Cybernetics, 2015, 45(5): 1003-1016.

[9] Burman S. Improved global routing using λ-geometry[C]//Annual Allerton Conference on Communication, Control and Computing,

1991:1083-1092.
[10] Li J, Bai Z, Yu R, et al. Mobile location privacy protection algorithm based on PSO optimization[J]. Chinese Journal of Computers, 2018, 41(5): 1037-1051.
[11] LUO Q, JI W D, XU H T, et al. Particle swarm optimization with global best particle dimension-by-dimension mutation[J]. Journal of Chinese Computer Systems, 2020, 41(2): 259-263.
[12] HU K, LI J L, LIN X L, et al. Hybrid particle swarm optimization algorithm combining gravity measure and centroid mutation strategy [J]. Journal of Chinese Computer Systems, 2022, 43(2): 285-292.
[13] 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.
[14] Nath P, Dey S, Nath S, et al. VLSI routing optimization using hybrid PSO based on reinforcement learning[C]//IEEE VLSI Device Circuit and System, 2022: 238-243.
[15] Goh Y, Jung D, Hwang G, et al. Consumer electronics product manufacturing time reduction and optimization using AI-based PCB and VLSI circuit designing[J]. IEEE Transactions on Consumer Electronics, 2023, 69(3): 240-249.
[16] Chu C, Wong Y. FLUTE: fast lookup table based rectilinear Steiner minimal tree algorithm for VLSI design[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2007, 27(1): 70-83.
[17] Latha N, Prasad G. Memory and I/O optimized rectilinear Steiner minimum tree routing for VLSI[J]. International Journal of Electronics, Communications, and Measurement Engineering, 2020, 9(1): 46-59.
[18] Kundu S, Roy S, Mukherjee S. Rectilinear Steiner tree construction techniques using PB-SAT-based methodology[J]. Journal of Circuits, Systems and Computers, 2020, 29(4): 2050057.
[19] Nath S, Gupta S, Biswas S, et al. GPSO hybrid algorithm for rectilinear Steiner tree optimization[C]//IEEE VLSI Device Circuit and System, 2020: 365-369.
[20] Zhu Q, Zhou H, Jing T, et al. Spanning graph-based nonrectilinear Steiner tree algorithms[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2005, 24(7): 1066-1075.
[21] Ho T, Chang C, Chang Y, et al. Multilevel full-chip routing for the X-based architecture [C]//Annual Design Automation Conference, 2005: 597-602.
[22] Arora T, Moses M. Ant colony optimization for power efficient routing in manhattan and non-manhattan VLSI architectures [C]// IEEE Swarm Intelligence Symposium, 2009: 137-144.
[23] Wu H, Xu S, Zhuang Z, et al. X-architecture Steiner minimal tree construction based on discrete differential evolution[C]//International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery, 2020: 433-442.
[24] LIU G G, HUANG Y F, WANG X, et al. Hybrid discrete particle swarm optimization algorithm for X-architecture Steiner minimal tree construction with Slew constraints [J]. Chinese Journal of Computers, 2021, 44(12): 2542-2559.
[25] Ajwani G, Chu C, Mak W. FOARS: FLUTE based obstacle-avoiding rectilinear Steiner tree construction[C]//International Symposium on Physical Design, 2011: 27-34.
[26] Lin K, Lin Y, Li Y, et al. A maze routing-based algorithm for ML-OARST with pre-selecting and re-building Steiner points[C]//The on Great Lakes Symposium on VLSI, 2017: 399-402.
[27] Kundu S, Roy S, Mukherjee S. An efficient obstacle-avoiding rectilinear Steiner tree construction method using PB-SAT[J]. IETE Journal of Research, 2023, 69(6): 3346-3356, .
[28] Wang R, Pai C, Wang J, et al. Efficient multi-layer obstacle-avoiding region-to-region rectilinear Steiner tree construction[C]//Annual Design Automation Conference, 2018: 1-6.
[29] Jing T, Feng Z, Hu Y, et al. λ-OAT: λ-Geometry obstacle-avoiding tree construction with O(*n*log*n*) complexity[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2007, 26(11): 2073-2079.
[30] 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.
[31] Huang X, Guo W, Liu G, et al. FH-OAOS: a fast four-step heuristic for obstacle-avoiding octilinear Steiner tree construction[J]. ACM Transactions on Design Automation of Electronic Systems, 2016, 21(3): 1-31.
[32] Huang X, Guo W, Liu G, et al. MLXR: multi-layer obstacle-avoiding X-architecture Steiner tree construction for VLSI routing[J]. Science China Information Sciences, 2017, 60(1): 1-3.
[33] Lin Y, Chang S, Li Y. Critical-trunk-based obstacle-avoiding rectilinear Steiner tree routings and buffer insertion for delay and slack optimization[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2011, 30(9): 1335-1348.
[34] Zhang Y, Pan D. Timing-driven, over-the-block rectilinear Steiner tree construction with pre-buffering and slew constraints[C]//International Symposium on Physical Design, 2014: 29-36.
[35] Liu G, Zhou R, Xu S, et al. Two-stage competitive particle swarm optimization based timing-driven X-Routing for IC design under smart manufacturing[J]. ACM Transactions on Management Information Systems, 2022, 13(4): 1-26.
[36] Huang H, Chang S, Liu Y, et al. Timing-driven non-rectangular obstacles-avoiding routing algorithm for the X-architecture [C]// World Scientific and Engineering Academy and Society International Conference, Stevens Point, 2009: 31-34.
[37] 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.
[38] Chen X, Zhou R, Liu G, et al. Timing-driven X-architecture Steiner minimum tree construction based on social learning multi-objective particle swarm optimization[C]//Companion Proceedings of the Web Conference, 2021: 77-84.
[39] Elmore W. The transient response of damped linear networks with particular regard to wideband amplifiers[J]. Journal of Applied Physics, 1948, 19(1): 55-63.
[40] Ho T, Chang Y, Chen S. Full-chip nanometer routing techniques [M]. Dordrecht: Springer, 2007.
[41] Tripathi J, Junjariya D, Achar R. Optimization of decoupling capacitors in VLSI systems using granularity learning and logistic regression based PSO[C]//IEEE/MTT-S International Microwave Symposium-IMS, 2023: 159-162.
[42] Liu G, Chen G, Guo W, et al. DPSO-based rectilinear Steiner minimal tree construction considering bend reduction[C]//7th International Conference on Natural Computation, 2011: 1161-1165.
[43] Lin Y, Chang S, Li Y. Critical-trunk-based obstacleavoiding rectilinear Steiner tree routings and buffer insertion for delay and slack optimization[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2011, 30(9): 1335-1348.

附中文参考文献:

[3] 王景熠,朱予涵,周茹平,等. 基于动态粒子群优化的 X 结构 Steiner 最小树算法[J]. 计算机工程,2024,50(9):226-234.
[11] 罗　强,季伟东,徐浩天,等. 一种最优粒子逐维变异的粒子群优化算法[J]. 小型微型计算机系统,2020,41(2):259-263.
[12] 胡　凯,李均利,林秀丽,等. 结合引力测度和质心变异策略的混合粒子群优化算法[J]. 小型微型计算机系统,2022,43(2):285-292.
[24] 刘耿耿,黄逸飞,王　鑫,等. 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[J]. 计算机学报,2021,44(12):2542-2559.