分 类 号 TP302
收藏编号
学校代码 10386

密级
学号 115410002
编号

福州大学

博士研究生学位（毕业）论文

非曼哈顿结构下 VLSI 总体布线算法研究

学 科 专 业：应用数学
研 究 方 向：计算智能及其应用
研 究 生 姓 名：刘耿耿
指 导 教 师 、职 称：陈国龙 教授
所 在 学 院：离散数学与理论计算科学研究中心

答辩委员会主席签名：

二〇一五 年 三 月

一　遵守学术行为规范承诺

本人已熟知并愿意自觉遵守《福州大学研究生和导师学术行为规范暂行规定》和《福州大学关于加强研究生毕业与学位论文质量管理的规定》的所有内容，承诺所提交的毕业和学位论文是终稿，不存在学术造假或学术不端行为，且论文的纸质版与电子版内容完全一致。

二　独创性声明

本人声明所提交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得福州大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。本人完全意识到本声明的法律结果由本人承担。

三　关于论文使用授权的说明

本人完全了解福州大学有关保留使用学位论文的规定，即：学校有权保留送交论文的复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存论文。（保密的论文在解密后应遵守此规定）

本学位论文属于（必须在以下相应方框内打“√”，否则一律按“非保密论文”处理）：

1、保密论文：　□本学位论文属于保密，在_____年解密后适用本授权书。

2、非保密论文：□本学位论文不属于保密范围，适用本授权书。

研究生本人签名：_____　　签字日期：20　　年　　月　　日

研究生导师签名：_____　　签字日期：20　　年　　月　　日

非曼哈顿结构下 VLSI 总体布线算法研究

中文摘要

总体布线是超大规模集成电路(Very Large Scale Integration, VLSI)物理设计中极为重要的一个环节。非曼哈顿结构的提出为物理设计带来诸多性能的提高，但该结构的引入和多层工艺的普及，使得总体布线问题更为复杂，且目前研究工作只就某些局部目标展开，缺乏一种该结构下有效完整的总体布线方案。

作为一类基于种群进化的方法，粒子群优化算法(Particle Swarm Optimization, PSO)是一种强有力的优化工具，且相对其他进化算法而言，具有执行操作简单且有能力快速收敛至一个合理、优秀的解方案。因此，研究非曼哈顿结构下 VLSI 总体布线中存在的若干问题和困难，并根据非曼哈顿结构的性质及该结构下总体布线相关子问题的特点，改造原有的 PSO 算法，从而寻找更为有效的非曼哈顿结构下 VLSI 总体布线算法，具有重要的理论价值和实际意义。

本文工作正是在这样的背景下，基于国家自然科学基金项目《非曼哈顿结构下 VLSI 多层总体布线算法研究》，对非曼哈顿结构 VLSI 总体布线相关问题及 PSO 在其中的应用展开一些研究工作。本文选取 X 结构作为非曼哈顿结构的代表，完成的主要工作如下：

(1) X 结构 Steiner 最小树(X-architecture Steiner Minimal Tree，XSMT)是非曼哈顿结构总体布线算法中多端线网的最佳连接模型，属于 NP 难问题。本文基于 PSO 提出了一种有效的 XSMT 构造算法，以线长为优化目标。该算法的更新方式引入了遗传算法中的交叉和变异算子，并同时采取自适应调整学习因子的策略，以提高 PSO 算法的收敛速度。实验结果表明所提算法能得到较好的 XSMT 解方案，并且可获得多种不同拓扑的 XSMTs，从而有利于 VLSI 总体布线阶段的拥挤度优化。

(2) 构建时延驱动 Steiner 树是 VLSI 性能驱动总体布线阶段中一个非常重要的问题。针对此问题，本文基于多目标 PSO 和 Elmore 时延模型提出了一种构建时延驱动 X 结构 Steiner 树的有效算法，从而有助于性能驱动 X 结构总体布线问题的研究。该算法设计了边变换操作使算法拥有进化到最优解的能力，且引入并查集策略以避免无效解的产生。另外，该算法考虑到拐弯数的优化工作，这是影响芯片可制造性的一个关键指标。本文是第一次在非曼哈顿结构下性能驱动总体布线工作中考虑到拐弯数的减少。相关实验结果表明该算法相对其他算法取得可观的性能优化效果，并且在不同集成电路工艺下算法是稳定且有效的。

(3) 针对多层绕障 X 结构 Steiner 最小树问题，以最小化布线总代价为目标，并同时考虑到通孔数的优化，本文提出了一种基于离散 PSO 算法和惩罚机制的多层绕障 X 结构 Steiner 最小树构建算法。算法在粒子的更新公式中引入边变换操作以增强布线树的绕障能力和线长优化能力，并设计适合 X 结构 Steiner 最小树和多层结构的多层边点对编码策略。通过提出合适的惩罚机制用以融入粒子的适应度函数设计中，该算法能更有效地考虑绕障情况和通孔数。同时，引入预处理策略大大减少了绕障情况的重复判断次数和通孔数的重复计算次数。实验结果表明该算法相关策略的有效性以及该算法相对同类算法能取得可观的布线总代价减少率，是在当前同类算法中具有最强的布线总代价优化能力。而且，针对定向边约束的绕障 X 结构 Steiner 最小树这一更为接近实际芯片设计工艺的问题，本文重新定义布线边的代价，并基于多层绕障 X 结构 Steiner 最小树算法的基础上构建求解该问题的有效算法，进而探讨不同定向边约束模型的优劣，以期为非曼哈顿结构总体布线提供一定的理论支撑。这是第一次开展定向边约束的绕障 X 结构 Steiner 最小树问题的研究。

(4) 针对 X 结构下的总体布线问题，本文提出一种基于整数线性规划(Integer Linear Programming, ILP)模型、划分策略及 PSO 等技术的高质量 X 结构总体布线算法。该算法基于一种新颖的 ILP 模型，可获得更为均为的布线方案，同时设计一种有效 PSO 算法用以求解所提出的 ILP 模型。为了能更为有效地求解 ILP 模型，采用划分策略将大规模的总体布线问题划分为多个中小规模的问题，从总体布线问题中最拥挤的区域开始布线，并不断扩张布线区域直至覆盖整个芯片。在总体布线的后续阶段，设计一种基于新布线边代价的迷宫算法，以线长优化和拥挤均衡为目标，布通一些仍未布线的线网。这是第一次使用并行算法求解 X 结构总体布线问题。实验结果表明该算法不仅在大规模电路中可获得较好解，还可在总线长方面相对现有工作而言取得最佳结果。

关键词：超大规模集成电路，非曼哈顿结构，总体布线，Steiner 树，多层布线，粒子群优化，时延，拐弯数，绕障碍物，拥挤均衡

Research on VLSI Global Routing Algorithms in Non-Manhattan Architecture

Abstract

Global routing is one of the most important steps in very large scale integration (VLSI) physical design. The introduction of non-Manhattan architecture can improve many performance indexes of the chip in physical design. The proposed non-Manhattan architecture and pervasive multilayer technology make the global routing problem be more complex. Furthermore, the relevant research work of the global routing problem in non-Manhattan architecture only focused on certain local goals. However, it still lacks an effective and complete global routing algorithm in non-Manhattan architecture.

As a swarm-based evolutionary method, particle swarm optimization (PSO) has been shown to be a powerful optimization tool. The advantages of PSO over many other optimization algorithms are its implementation simplicity and ability to converge to a reasonably good solution quickly. Thus, it needs to study the issues of VLSI global routing in non-Manhattan architecture and reconstruct our previous discrete PSO algorithm based on both the properties of non-Manhattan architecture and the features of the corresponding global routing problem. As a result, designing a more effective global routing algorithm in non-Manhattan architecture has important theoretical value and practical significance.

In this context, this thesis does some research on the global routing problem in non-Manhattan architecture and the application of PSO based on the National Natural Science Fund Project "Research on VLSI Multilayer Global Routing Algorithms in non-Manhattan Architecture". The X-architecture is selected as the representative of non-Manhattan architecture and the main work is as follows.

First, the X-architecture Steiner minimal tree (XSMT) problem is an NP-hard problem, which is the best connection model for a multi-terminal net in non-Manhattan global routing problem. This thesis presents an XSMT algorithm based on PSO to optimize the wire length. A self-adapting strategy that can adjust the learning factors and the crossover and mutation operators of genetic algorithm are proposed to improve the convergence rate of PSO algorithm. The experimental results show that the proposed algorithm can efficiently provide a solution of XSMT problem with good quality. Moreover, the algorithm can obtain several topologies of XSMTs, which is beneficial for optimizing congestion in VLSI global routing stage.

Second, constructing a timing-driven Steiner tree is very important in VLSI performance-driven global routing stage. In this thesis, an effective algorithm is presented to construct a timing-driven X-architecture Steiner tree for non-Manhattan performance- driven global routing based on the multi-objective PSO and the Elmore delay model. An edge transformation is employed in our algorithm to make the

particles have the ability to achieve the optimal solution, while union-find partition is used to prevent the generation of invalid solution. For the purpose of reducing the number of bends which is one of the key factors of chip manufacturability, we present an edge-vertex encoding strategy combined with the edge transformation. To our best knowledge, no approach has been proposed to optimize the number of bends in the process of non-Manhattan performance-driven global routing. The experimental results indicate that the proposed algorithm has achieved a better delay value than others and is stable and effective under different integrated circuit technologies.

Third, for the multilayer obstacle-avoiding X-architecture Steiner minimal tree (ML-OAXSMT) problem, an effective algorithm based on discrete PSO is presented to construct an ML-OAXSMT. First, an edge transformation strategy is employed to make the particles have the ability to bypass the obstacles and obtain a better wire length. Second, according to the feature of the ML-OAXSMT problem, we design a multilayer edge-vertex encoding strategy. Third, during the design process of the fitness function, a penalty mechanism is proposed to help the particle avoid the obstacles and reduce the generation of vias. Moreover, a pretreatment strategy is presented to reduce the total number of the judgment for the routing conditions around obstacles and the calculation counts of the vias. The experimental results show that the effectiveness of our proposed strategies and our algorithm can achieve the best solution quality among the existing algorithms. Finally, we redefine the routing edge cost and then construct an obstacle-avoiding preferred direction X-architecture Steiner tree (OAPD-XST), which is more in line with the industrial production. To our best knowledge, this is the first work to address the OAPD-XST problem and can offer the theoretical support for the chip design based on non-Manhattan architecture.

Finally, for the global routing problem in X-architecture, this thesis presents a high-quality VLSI global router in X-architecture, called XGRouter, which heavily relies on the integer linear programming (ILP) technique, the partition strategy, and PSO. A new ILP formulation, which can achieve a more uniform routing solution than other formulations and can be effectively solved by the proposed PSO, is proposed. To effectively formulate the new ILP formulation, a partition strategy which decomposes a large-sized problem into some small-sized sub-problems is adopted and the routing region is extended progressively from the most congested region. In the post-processing stage of XGRouter, maze routing based on a new routing edge cost is designed to further optimize the wire length and keep the congestion uniformness. To our best knowledge, XGRouter is the first work to use the concurrent algorithm to solve the global routing problem in X-architecture. The experimental results show that XGRouter can produce the solutions of higher quality than other global routers in the large scale circuit problem.

Keywords: Very Large Scale Integration, Non-Manhattan Architecture, Global Routing, Steiner Tree, Multilayer Routing, Particle Swarm Optimization, Timing Delay, Bends, Obstacle-avoiding, Congestion Uniformness

目录

主要符号表

EDA	Electronic Design Automation
FSTs	Full Steiner Trees
GRG	Global Routing Graph
IC	Integrated Circuit
ILP	Integer Linear Programming
ISPD	International Symposium on Physical Design
ML-OARSMT	Multilayer Obstacle-Avoiding Rectilinear Steiner Minimal Tree
ML-OAXSMT	Multilayer Obstacle-Avoiding X-architecture Steiner Minimal Tree
ML-RSMT	Multilayer Rectilinear Steiner Minimal Tree
MOPSO	Multiple Objective Particle Swarm Optimization
MRMCST	Minimum-Cost Spanning Tree with a Minimum Radius
MST	Minimum Spanning Tree
OAPD-RST	Obstacle-Avoiding Preferred Direction Rectilinear Steiner Tree
OAPD-XST	Obstacle-Avoiding Preferred Direction X-architecture Steiner Tree
OARSMT	Obstacle-Avoiding Rectilinear Steiner Minimal Tree
OASMT	Obstacle-Avoiding Steiner Minimal Tree
OAXSMT	Obstacle-Avoiding X-architecture Steiner Minimal Tree
RSMT	Rectilinear Steiner Minimal Tree
SMT	Steiner Minimal Tree
SOC	System-On-a-Chip
SPT	Shortest Path Tree
TSP	Travelling Salesman Problem
TXST	Timing-driven X-architecture Steiner Tree
VLSI	Very Large Scale Integration
XSMT	X-architecture Steiner Minimal Tree
XST	X-architecture Steiner Tree

第一章 绪论

1.1 引言

大规模集成电路是信息技术与信息产业的硬件核心，其发展水平的高低已成为衡量一个国家科学技术和工业发展水平的重要标志。电子信息产业是以大规模集成电路为基础，其发展对提高我们国家的产业技术创新能力、发展国民经济和加快现代国防建设都具有极其重要的作用[1-3]。当前集成电路产业向超深亚微米工艺不断推进，芯片的集成度进一步提高，一块芯片上所能集成的电路元件越来越多，加上存储空间的局限性和封装工艺的限制，对超大规模集成电路(Very Large Scale Integration, VLSI)设计方法提出了新的挑战。很多超大规模集成电路设计问题的复杂度都随着工艺不断发展呈指数增长，导致现有的电子设计自动化(Electronic Design Automation, EDA)工具难以应付，同时现有的 EDA 工具也缺乏考虑当前工艺下存在的一系列新问题。

由于VLSI物理设计的复杂性，整个物理设计过程被分为以下几个步骤：电路划分、布图规划、布局和布线。在已发布的SRC “Physical Design CAD Top10 Needs”中指出当前物理设计急待解决的十大问题，布线问题首当其冲，在芯片尺寸和容量上，布线工作需要绕线的电路芯片规模达到成千上万个大模块和几百万个小模块，同时要求在合理可行的时间完成布线工作。此外，布线的质量严重影响了设计过程中的其他需求，包括定时和互连线分析。在芯片设计中为了降低布线过程面临的高复杂度问题，通常将布线划分成两个步骤：总体布线和详细布线。其中，总体布线是一种粗略绕线过程，它的功能是将每条待绕线的线网的各部分合理分配到芯片中的各个布线通道区，从而明确定义了各个布线通道区的布线问题。而详细布线则是在总体布线结果的指导下完成各个布线通道区的具体绕线情况。由此可见，VLSI总体布线的质量对详细布线的成功与否以及芯片的最终性能都具有极大的影响[1]。

布线互连结构可分为曼哈顿结构和非曼哈顿结构。曼哈顿结构的布线方向只能是水平走线和垂直走线两种，而非曼哈顿结构的布线方向则更为多样化，主要包括走线方向为 0°，90°和±45°的 X 结构以及走线方向为 0°，60°和 120°的 Y 结构。近年来，ISPD(International Symposium on Physical Design)关于 VLSI 物理设计算法举行了针对性的竞赛，其中 2007 年至 2010 年是针对布线问题。而针对 VLSI 物理设计中的重要组成部分——总体布线，研究人员提出了大量方法[4-18]。大部分总体布线算法都是以曼哈顿结构为模型基础，通过优化 Steiner

树的拓扑、变换线宽、插入缓冲器等方法对线长、时延进行优化，达到优化芯片性能的目的。随着系统级芯片(System-On-a-Chip, SOC)设计概念的出现和制造工艺的不断发展，互连线的延迟对 VLSI 设计的影响越来越大，同时互连线的不断增长会降低芯片的速度、造成过高的功耗以及增大噪声[19]。因而 SOC 设计对互连线线长优化、电路性能提出了更高的要求。但基于曼哈顿结构对芯片进行线长与时延的优化，由于其绕线方向有限不能够充分地利用布线区域，导致互连线资源的过分冗余，故基于曼哈顿结构的相关物理设计阶段在优化互连线线长时，限制了相关策略的优化能力。因此，有必要从根本入手改变传统的曼哈顿结构，一些研究人员开始尝试以非曼哈顿结构为基础模型进行布线，实现芯片整体性能的优化。在非曼哈顿结构下能够进一步减少芯片面积、缩短线长、降低通孔数目和功耗，使物理设计中的很多性能指标得到提高。非曼哈顿结构的提出，使得超大规模集成电路物理设计各个阶段面临新的机遇和挑战，同时引起整个物理设计领域算法的更新，即对以非曼哈顿结构作为基础模型的物理设计过程包括布图规划与布局、布线、参数提取等方面提出新的要求[20,21]，也对 EDA 设计工具的研究提出了巨大挑战，其中，与总体布线的联系紧密。此外，随着集成电路设计工艺的不断发展，允许绕线的布线层数随之增加，大幅度减少了互连线宽度和互连线间距，从而提高了集成电路的性能和密度。于是，多层总体布线应运而生，并且引起了诸多研究机构的广泛关注，包括清华大学、香港科技大学和 IBM，Intel，斯坦福大学等诸多国内外科研机构，已经成为国际上 EDA 领域的研究热点之一[3]。

传统的曼哈顿结构下的总体布线算法虽然可以应用到非曼哈顿结构总体布线，但需要进行特别的处理，使算法变得更加复杂，从而难于直接有效地对该问题进行求解。再者，现有关于非曼哈顿结构的研究工作主要集中在非曼哈顿结构 Steiner 树的构造，对非曼哈顿结构下总体布线算法研究还不是很充分，仍未出现针对非曼哈顿结构特点而与曼哈顿结构总体布线算法相比有较大改进的算法。在非曼哈顿结构和多层设计概念下的总体布线问题变得更为复杂，目前还很难实现理想的总体布线。因此，寻求更为有效的总体布线算法，构造一个高效的非曼哈顿结构总体布线器，具有重要的理论价值和实际意义。

为了便于叙述，下文首先对超大规模集成电路总体布线进行概述，包括总体布线图(Global Routing Graph，GRG)和与本文所研究的非曼哈顿结构总体布线问题相关的两个基础定义进行介绍，然后总结并分析超大规模集成电路总体布线的研究现状，最后阐述论文的主要研究内容和主要贡献，同时给出论文的组织架构安排。

1.2 总体布线概述

作为集成电路设计的一个阶段，物理设计是与EDA设计芯片研制与芯片生产密切相关的，直接影响到芯片设计的生产周期、生产成本和芯片的质量。十几年来，物理设计是电路自动化设计工具中发展最快和自动化程度最高的领域之一。随着集成电路的特征尺寸不断减少，超大规模集成电路中工艺和电路规模以摩尔定律经历了巨大的进步，电路设计中不断增长的复杂性进一步扩大了物理设计中自动化设计问题的难度，并同时迎来一系列新的挑战[3]。鉴于VLSI物理设计的复杂性，作为VLSI设计中最为耗时的一个步骤——物理设计，如图1-1所示，往往被具体细分为电路划分、布图规划和布局、总体布线和详细布线等四个阶段。

VLSI物理设计的总体流程包括[1]：首先，由于VLSI芯片设计中电路元件的规模不断扩大，在物理设计中需要采用电路划分方法将复杂庞大的电路系统分解至合理小的电路子系统。其次，在电路划分后，布图规划和布局步骤则是将不同形状和大小的单元或模块合理地放置到芯片的不同布线区域，同时满足芯片固有的一些相关几何约束。然后，在布局阶段确定完模块和引脚各自的位置后，总体布线将每条待绕线的线网的各部分合理分配到芯片中的各个布线通道区，从而明确定义了各个布线通道区的布线问题。最后，详细布线则是完成各个布线通道区的具体绕线情况。从图1-1可看出，详细布线阶段是以总体布线结果作为指导进行布线工作的。因而，总体布线的结果对详细布线的成功与否起到决定性作用，同时总体布线严重影响最后制造出来的芯片性能。由此可见，总体布线是VLSI物理设计中一个极为重要的步骤。

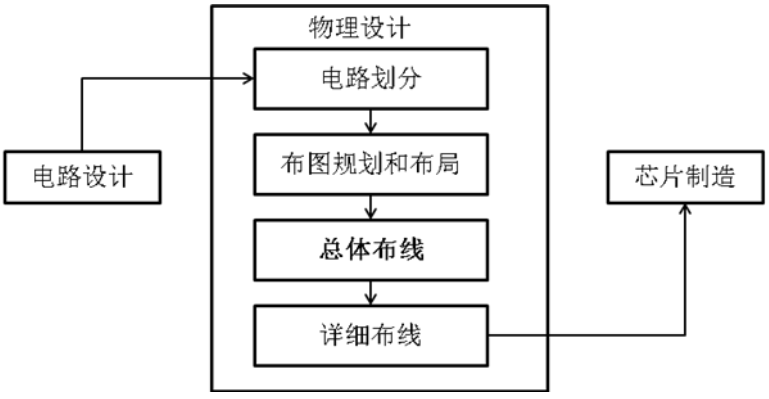


图 1-1 VLSI物理设计的设计过程

1.2.1 总体布线图

总体布线问题是一个典型的图论问题。GRG 将布线区域、每个区域内的布线容量、布线区域内的引脚信息以及不同布线区域之间的相互关系等信息抽象为一

张图。采用不同的设计模式，可对图模型产生很大的影响，从而产生多种布线图模型，常见的布线图模型主要包括网格图模型、布线规划图模型以及通道相交图模型等模型[3]。

(1) 网格图模型。网格图模型是将整个布线区域分为一个行列交错的矩阵形式，每个总体布线单元(Global Routing Cell，GRC)由一个顶点表示，GRC 之间的邻接关系则由水平边和垂直边表示。而对于给定的线网集合，则将它们的引脚集合按照其所在的总体布线单元，映射到该总体布线单元对应的顶点上。此时，VLSI 总体布线问题则是在总体布线图上寻找这些映射后的顶点集合的连接关系。网格图模型适用于门阵列和标准单元布图模式，而且可将网格图模型经过简单的扩展，用于建立多层总体布线问题的网格图模型，有助于探讨多层布线问题的求解。

(2) 布图规划图模型。该模型是基于布局结果的基础上构建的，对于给定的布局方案，布图规划图的一个顶点表示布局方案中的一个模块，模块之间若存在相邻关系，则映射成布图规划图的一条边。类似于网格图模型，布图规划图模型也是将给定线网集合中的引脚映射到布线图的顶点上。该模型适合给模块间的布线容量建模，但存在的缺点是对线长估计的能力不足。

(3) 通道相交图模型。通道相交图模型能够详细准确表达问题信息，跟上述两个图模型一样，是将给定线网集合中的引脚映射到总体规划图相应的边上，形成新的顶点，从而被扩展成为包含总体布线所需信息的总体布线图。该模型适用于积木块自动布图模式。

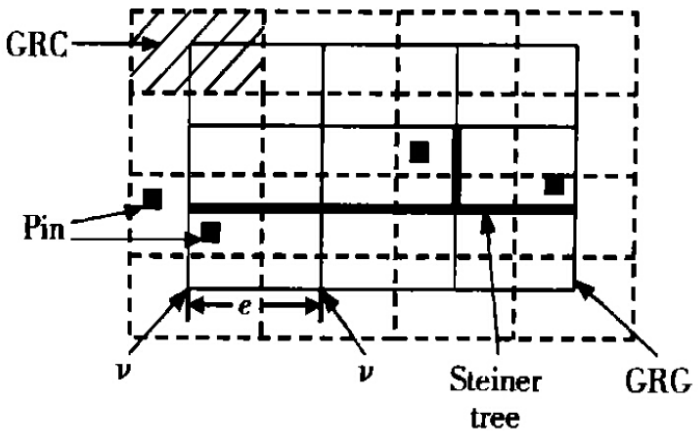


图 1-2 基于网格图模型的总体布线图[20]

给定一个布线图，如图 1-2 所示行列交错的虚线网格集合，即常见的网格图模型，每个虚线网格框表示一个总体布线单元，引脚(Pin)集合根据布局后的结果放置在相应的 GRC 中。GRC 所对应的 GRG 即图 1-2 中行列交错的实线网格集

合，将每个 GRC 映射到 GRG 中作为 GRG 的一个顶点(v)，有邻接关系的两个 GRC 之间则映射为 GRG 的一条边(e)。将 GRC 两两之间的关系按照这种方式映射则变成 GRG，故 GRC 与 GRG 是一一对应的关系。

总体布线问题是在给定 GRG 的基础上，寻找这些引脚集合的布线树，图 1-2 中采用 Steiner 树(Steiner tree)作为布线树模型，该布线树需要满足工艺条件和引脚集合的连接关系。接下来将具体介绍总体布线问题中的相关定义，包括 Steiner 最小树的定义、总体布线的优化目标以及本文所研究的非曼哈顿结构的数学基础知识。

1.2.2 总体布线问题相关定义

VLSI 总体布线问题中，多端线网的总体布线问题是寻找一棵连接给定引脚集合的布线树问题，而 Steiner 最小树则是相对于其他方法所得的布线树来说具有更小的布线树总线长。因此，Steiner 最小树被看做是总体布线问题中多端线网的最佳连接模型，并且在总体布线图上的 Steiner 最小树构建算法是所有总体布线算法的基础。

定义 1.1 (Steiner 最小树)[3] 给定一个带边权的图 $G=(V, E)$和一个节点子集 $R\subseteq V$，选择一个子集 $V'\subseteq V$，使得 $R\subseteq V'$并且 V'构成一个最小费用的树，其中，R 是要求 Steiner 树连接的节点集，$V'-R$ 是 Steiner 点集。

VLSI 总体布线问题最初是以线长最小化为优化目标，但随着制造工艺不断发展和芯片特征尺寸的不断缩小，互连线延迟对芯片性能的影响越来越大，因此，时延和串扰等优化目标也需要在总体布线问题中考虑。同时，影响到芯片的可布性和可制造性的因素，包括溢出数、拥挤度、通孔数等优化目标，也是当今总体布线工作需要优化的指标。

随着集成电路在设计规模和制造工艺方面都迅猛地发展，基于曼哈顿结构的优化策略对互连线的优化能力不能很好地满足 SOC 设计的需求，从而一些学者开始研究基于非曼哈顿结构的物理设计方法，其中，与总体布线联系紧密。非曼哈顿结构的数学基础是 λ-几何学理论，下文描述布线互连结构的 λ-几何学理论的具体定义。

定义 1.2 (布线互连结构的 λ-几何学理论) [20] 在 λ-几何学理论中，布线的方向为 $i\pi/\lambda$，i 为任意整数，λ 为正整数。通过对 λ 和 i 的不同取值，可得到不同的布线方向：

(1) 当 $\lambda=2$ 时，布线的方向为 $i\pi/2$，走线方向包括 0°和 90°，对应传统的曼哈顿结构，亦称为矩形结构；

(2) 当 $\lambda=3$ 时，布线的方向为 $i\pi/3$，走线方向包括 0°，60°和 120°，称为 Y

结构；

(3) 当 $\lambda=4$ 时，布线的方向为 $i\pi/4$，走线方向包括 0°，90°和±45°，称为 X 结构。

布线互连结构可分为曼哈顿结构和非曼哈顿结构，目前在非曼哈顿结构中已开展的研究主要是定义1.2所示的X结构和Y结构。本文工作选取其中得到制造工艺支持的X结构作为非曼哈顿结构的代表展开研究。

1.3 国内外研究现状及发展动态分析

近年来，集成电路在设计规模和制造工艺方面都迅猛地发展，由此为电子设计自动化带来新的挑战。而物理设计中总体布线阶段所面临的问题也是越来越复杂，总体布线问题的研究工作主要集中在曼哈顿结构下布线树的构造和总体布线算法的设计，并取得一些不错的研究成果[22-31]。特别是在 ISPD 所举行的 VLSI 总体布线算法竞赛[32,33]中，涌现出了 FGR[6]，MaizeRouter[7]，Archer[9]，BoxRouter 2.0[11]，Fastroute 4.0[14]，GRIP[15]，NTHU-Route 2.0[17]，NCTU-GR 2.0[34]，GPU-CPU Router[35]，STAIRoute[36]，BOB-router[37]，VFGR[38]等优秀的总体布线器，并在总体布线竞赛中脱颖而出。目前工艺下，金属层数为多层的，以上总体布线器均可进行多层布线，按照多层布线方式不同分为两类：完全 3-D 总体布线和平面总体布线紧接层分配(2.5-D 总体布线)。其中，FGR 和 GRIP 采用完全 3-D 总体布线方式，相对于其他进行平面布线后紧接层分配的 2.5-D 总体布线器，在一些优化目标上取得更为准确的总体布线方案，但是付出了较大的时间和空间复杂度。

以上总体布线器都是基于曼哈顿结构，要求布线走向只能是水平或是垂直走向，并且要求每一金属层布线走向只能为一种，此外，其中大部分布线是要求基于网格结构的。而基于非曼哈顿结构的总体布线中布线走向除了水平或垂直走向，还可以包含±45°或是 60°和 120°，此外，其中部分金属层布线走向可以有多种。因此，原有的方法已经不能直接有效地应用到非曼哈顿结构总体布线中。

现有研究人员对能够提高可观的线长减少量等物理设计指标的非曼哈顿结构[39]已展开一些尝试性的研究，特别是出现专门的工业联盟推广 X 结构[40]，为这样的研究提供实现和验证的基础。洪先龙等指出在这样的背景下非曼哈顿结构的研究将成为国际上布线领域的热点研究方向[20]。文献[39]提出了关于在 X 结构下的布线树和布线算法的一些挑战和机遇，同时给出该结构下良好的展望，并指出在非曼哈顿结构下，Steiner 最小树问题仍是最为关键的问题之一。关于非曼哈顿结构 Steiner 树的研究算法包括精确算法[41,42]和传统的启发式算法[43-47]。文献[41]提出了基于分支界限法构造一棵六边形 Steiner 树，但只适用于小规模问题。文献[42]使用精确算法和一系列的剪枝策略构造非曼哈顿结构树，并相对于最小

矩形 Steiner 树可取得大概 10%的线长减少量，但付出较高的算法复杂度代价。针对精确算法的时间复杂度高，人们开始探索启发式策略在构造非曼哈顿结构 Steiner 树中的应用。文献[43]给出了一个时间复杂度为 O(|*V*|+|*E*|)算法构造非曼哈顿结构 Steiner 树，但只局限于构造同构的非曼哈顿结构 Steiner 树，其中，|*V*|和|*E*|分别代表布线引脚的数量和布线树的边数。为了构造 Y 结构 Steiner 最小树，文献[44]给出了一个时间复杂度为 O(n^3log$n$)的启发式算法求解 Y 结构 Steiner 树，该算法是基于贪心思想。文献[45]提出了两种基于八角形生成图的非曼哈顿结构 Steiner 树构造算法：一种是采用边替换技术，其最坏运行时间是 O($n$log$n$)。另一个则是采用三角收缩技术，在稍微增加时间代价的基础上取得相对比较好的结果，二者同样都是基于贪心思想。文献[46]提出了一种基于划分策略的线长估算技术，可在没有布局和引脚位置信息这些先验知识的前提下产生布线方案，该方法可用于曼哈顿结构和非曼哈顿结构布线。文献[47]针对时延驱动 X 结构 Steiner 树的构建，提出一个修改算法，通过调整最长路径所经过的 Y 型布线，有效降低该最长路径的时延。与文献[47]不同的是，文献[48,49]是在 Hanan 网格和 Elmore 时延模型的基础上，构造 Y 结构时延驱动 Steiner 树，并验证了 Elmore 时延模型相对于线性延迟模型具有较高的计算精度和保真度。文献[50]基于图论，采用 k-IDEA 算法产生多组的绕线方案，并引入了分支界限概念加速求解，可得到一个总线长与拥挤度都较小的结果，但是产生过多的通孔。

构造非曼哈顿结构 Steiner 树是一个 NP 难问题，以上求解非曼哈顿结构 Steiner 树的研究工作都是基于精确算法和传统的启发式算法，而其中精确算法的时间复杂度随着问题规模呈指数增长，传统的启发式算法大多基于贪心策略，易陷入局部极值。二者在构造 Steiner 树时，并未充分利用非曼哈顿结构的几何性质，不能保证 Steiner 树的质量，且主要优化目标集中在线长优化，而较少提供拓扑优化的合适方法，导致在时间效率、线长缩短、时延优化等方面还不能令人满意。

文献[50]阐述了可供选择的 VLSI 布线互连结构，首次采用启发式的非曼哈顿结构布线树构造方法，实现了基于网格的互连布线优化算法。同时还注意到基于非曼哈顿结构布线会带来通孔数的增加，但是其线长减少带来的增益超过通孔数增加的代价。与文献[50]采用启发策略直接构造非曼哈顿结构 Steiner 树不同的是，文献[51]是借鉴已经在曼哈顿结构下取得成功的布线工具和方法应用于非曼哈顿布线算法中，综合两次曼哈顿结构树的构造方法得到所需的非曼哈顿结构树。在此基础上实现的非曼哈顿结构总体布线取得了一定的线长减少量和其他性能的优化，但算法的复杂度加剧。为了进一步简化布线复杂度，文献[21,52,53]提出了一类非曼哈顿结构多级布线器，其在细化阶段做高层块级总体布线时，由

于对底层布线的局部拥挤状况不了解，会导致出现一些上层决策不适合实际布线的问题。文献[54]基于整数线性规划模型和八角形网格模型的基础上，提出一种可同时优化线长、拐弯数、通孔数等目标的模拟布线算法，但未考虑到定向边约束问题和层分配问题。文献[55]只将非曼哈顿结构布线应用于实际芯片设计中部分线网的布线，以最小化对现有计算机辅助设计系统的一些改进，从而对非曼哈顿结构布线在实际生产中的应用做了初步尝试。文献[56]则指出了在非曼哈顿结构层分配中，很难调整布线需求和可得布线资源的关系。Hu 等采用随机子树生长方法在迭代数次后构建相应的布线树，并最终布通整个芯片[57]。Cao 等建立了非曼哈顿结构的总体布线模型，并提出了动态资源调度方法以减少潜在通孔数[58]。然而，这些关于非曼哈顿结构总体布线算法[57,58]是基于早期的 Steiner 树构建算法，导致这些非曼哈顿结构总体布线算法的优化效果不明显，甚至部分结果在线长指标上劣于曼哈顿结构总体布线算法。

非曼哈顿结构虽然给总体布线带来线长、芯片面积的减少，使得芯片性能和密度得到很大提高，但同时带来了通孔数增加的代价，而且直接将传统的总体布线算法应用到非曼哈顿结构总体布线问题中，会使得算法变得更复杂，并存在一定的局限性。目前，非曼哈顿结构总体布线主要集中在线长优化，而要优化芯片的整体性能，需进一步考虑时延、串扰、通孔等目标的优化。非曼哈顿结构的提出，引起整个物理设计领域的更新，特别是对于总体布线阶段，需要重新考虑建立布线边的代价函数问题、更新布线资源分配问题等。

随着集成电路的规模和工艺的不断发展，问题空间维数随之剧增，以及非曼哈顿结构的引入，使得传统的优化算法很难适应集成电路的需要，要么面临计算量爆炸，要么易陷入局部极值，无法接近全局最优解，这也导致了人们开始寻求各种启发式算法[25,59-62]。粒子群优化(Particle Swarm Optimization, PSO)算法作为一种新兴的基于群智能的随机优化算法，具备了其他进化算法所无法相比的简单容易实现和更强的全局优化能力等优势。算法自提出以来引起了众多学者的极大关注，在短短几年内形成了一个研究热点并出现了大量的研究成果[62-68]，大量实验结果也显示出粒子群优化算法确实是有力的优化工具且具有强大的生命力。特别是针对 VLSI 物理设计各个阶段的设计问题，也有相关工作将 PSO 算法应用于这些阶段，并取得一系列较为不错的研究成果[69-76]。针对 VLSI 物理设计中的第一个环节电路划分，文献[69]构造了一种求解电路二划分问题的离散粒子群优化算法；文献[70]进一步考虑时延因素，构造了电路划分的多目标问题模型，设计了一种求解多目标电路划分问题的混合离散粒子群优化算法；针对 VLSI 版图设计中线长估计的一种重要模型——最小生成树，文献[71]提出了一种求解多目标最小生成树问题的有效离散粒子群优化算法；针对布局/布图规划问题，文献[72]

充分考虑布图面积和互连线长度的最小化，刻画了相应的多目标优化数学模型，给出了一种基于 PSO 的智能决策算法，继而文献[73]构建了一种求解布局/布图规划问题的多目标粒子群优化(Multiple Objective Particle Swarm Optimization, MOPSO)算法的初步框架；此外，针对 VLSI 布线中考虑障碍物的最小矩形 Steiner 树问题，文献[74,75]初步提出了一种求解该问题的基于遗传操作的离散粒子群优化算法；针对 VLSI 布线中考虑到拐弯数的减少，文献[76]进一步给出了一种基于离散 PSO 并且同时考虑线长和拐弯数优化的矩形 Steiner 树构造算法。综上所述，这些相关工作已很好地把粒子群优化算法应用于 VLSI 物理设计中相关问题的求解，并刻画出相应的电路性质和数学优化模型。但这些相关工作中关于 VLSI 总体布线问题开展的一些研究都是基于曼哈顿结构，其中算法和模型是初步的，只停留在曼哈顿结构 Steiner 树的构造，尚未涉及到总体布线的完整流程。而相对于曼哈顿结构，非曼哈顿结构下的总体布线及相关问题显得更加复杂，同样是 NP 难问题，需要寻找有效的算法。这些相关工作所强化的粒子群优化算法为本文求解更为复杂的非曼哈顿结构总体布线问题提供一定的支持，且这些相关工作所研究 VLSI 物理设计的图论性质和相应的数学优化模型也为本文工作的开展积累了一定的研究经验。

综上所述，非曼哈顿结构的引入使得物理设计的诸多性能得到提高，比曼哈顿结构具有明显的优势。曼哈顿结构下的布线算法虽然部分可以应用到非曼哈顿结构布线，但需要进行特别的处理，使算法变得更加复杂，从而难以直接有效地对该问题进行求解。同时，已有的解决非曼哈顿结构下总体布线问题的启发式算法，或解决的问题规模小，或只能解决局部阶段工作，或与曼哈顿结构算法相比带来优化程度不大。目前，缺乏一种有效完整的解决非曼哈顿结构下 VLSI 总体布线问题的方案。因此，本文选择获得工业联盟支持的 X 结构作为非曼哈顿结构的代表，将针对 X 结构下 VLSI 总体布线中存在的若干问题和困难展开研究，充分利用 X 结构的几何性质和改造粒子群优化算法以及基于整数线性规划(Integer Linear Programming, ILP)模型，构造求解 X 结构下 VLSI 总体布线相关问题的有效策略，继而构建一个高效的总体布线器，以期可直接用于解决超大规模集成电路的 X 结构总体布线问题，具有重要的理论价值和实际意义。

1.4 论文的研究内容和主要贡献

作为一项迫切的基础性、前瞻性工作，本文正是看到了目前在该领域研究的不足和迫切需求，结合已有的关于粒子群优化算法和 VLSI 物理设计中各阶段相关研究工作，试图从新的角度考虑超大规模集成电路的非曼哈顿结构总体布线问题，以 X 结构作为非曼哈顿结构的代表，重点开展 X 结构 Steiner 最小树

(X-architecture Steiner Minimal Tree, XSMT)构建算法、时延驱动 X 结构 Steiner 最小树(Timing-driven X-architecture Steiner Tree, TXST)构建算法、多层绕障 X 结构 Steiner 最小树(Multilayer Obstacle-Avoiding X-architecture Steiner Minimal Tree，ML-OAXSMT)构建算法、X 结构下超大规模集成电路总体布线算法等方面的研究工作，以克服现有 X 结构下超大规模集成电路总体布线相关工作中存在的问题和不足，从而构建一种有效完整的解决超大规模集成电路 X 结构总体布线问题的方案，具体的研究内容和主要贡献如下：

(1) X 结构 Steiner 最小树构建算法研究

X 结构 Steiner 最小树是 X 结构总体布线算法中多端线网的最佳连接模型，其构建算法是所有总体布线算法研究的基础，该问题是 NP 难问题。因此，本文为了更好地研究超大规模集成电路 X 结构总体布线问题，首先开展 X 结构 Steiner 最小树构建算法的研究，为该 NP 难问题寻找一种有效的算法以寻找接近 Steiner 最小树的连接。

本文基于 PSO 提出了一种有效的 XSMT 构造算法，以线长为优化目标。为了让 PSO 能够用于离散问题的求解和克服高维问题空间中 PSO 算法收敛速度较慢的问题，该算法的更新方式引入了遗传算法中的交叉和变异算子，并同时采取自适应调整学习因子的策略。实验结果表明所提算法能得到较好的 XSMT 解方案，并且可获得多种不同拓扑的 XSMTs，从而有利于 VLSI 总体布线阶段的拥挤度优化工作。

(2) 考虑拐弯数优化的时延驱动 X 结构 Steiner 最小树构建算法研究

近年来，随着芯片特征尺寸的不断缩小，互连线延迟即时延，已经成为纳米领域中 VLSI 布线一个非常关键的指标。因此，时延因素在 Steiner 树的构建问题已不容忽视，本文有必要对以互连线延迟为优化目标的性能驱动总体布线工作进行研究。

针对 X 结构下性能驱动总体布线问题，本文基于 MOPSO 设计一种有效的算法以构建一棵时延驱动的 X 结构 Steiner 树，并同时考虑到拐弯数的优化。本文设计了边变换策略，用以增强 PSO 算法中粒子的进化能力，而同时考虑到 Steiner 点和布线边的变换工作，能够进一步加强布线树的线长优化能力。通过四种边选择方式重新调度 pseudo-Steiner 点的位置，本文提出了一种有效的编码策略。边变换策略可很好结合所提出的边点对编码策略以减少拐弯数。相关实验结果表明本文所构建的时延驱动 X 结构 Steiner 最小树算法相对其他算法取得可观的性能优化效果，并且在不同集成电路(Integrated Circuit, IC)工艺下算法是稳定且有效的。本文工作是非曼哈顿性能驱动总体布线工作中第一次考虑到拐弯数的优化。

(3) 多层绕障 X 结构 Steiner 最小树构建算法研究

近年来超大规模集成电路设计中芯片会存在宏单元、IP 预布好的线网等布线障碍物，在此基础上考虑到障碍物的 Steiner 最小树构建问题受到广泛的关注。同时，随着集成电路设计工艺的不断发展，允许绕线的布线层数随之增加，大幅度减少了互连线宽度和互连线间距，从而提高了集成电路的性能和密度。因此，多层布线应运而生，并且引起了诸多研究机构的广泛关注[3]。超大规模集成电路设计中多层绕障 X 结构 Steiner 最小树问题是考虑到障碍物、X 结构、多层等三个条件的 Steiner 最小树模型。

针对 ML-OAXSMT 问题，本文提出了一种 X 结构下带粒子群优化的多层绕障 Steiner 最小树构建算法。为了提高布线树的绕障能力，算法在 Steiner 点变换操作的基础上加入边变换操作，并设计了基于并查集思想的操作算子以避免生成树中环路的产生，同时设计了适合 X 结构和多层结构的多层边点对编码策略；提出了惩罚函数策略，以达到不违反绕障约束和不鼓励通孔产生，从而优化布线总代价；为了减少冗余的绕障判断次数和通孔计算次数，设计了预处理策略。实验结果表明了本文算法和相关策略的可行性和有效性，并在多层布线中布线总代价这一最重要的优化目标带来显著减少的效果。另外，由于考虑到定向边约束的绕障定向 X 结构 Steiner 最小树问题(Obstacle-Avoiding Preferred Direction X-architecture Steiner Tree，OAPD-XST)更符合 X 结构布线的实际工艺，所以基于 ML-OAXSMT 问题的构建算法，本文设计一种构造 OAPD-XST 的算法，进而分析 OAPD-XST 问题中不同定向边约束模型的优劣性。本文是第一次开展 OAPD-XST 问题的相关分析工作。

(4) X 结构下 VLSI 总体布线算法研究

现有关于 X 结构总体布线算法是基于早期的 Steiner 最小树构建算法，导致这些 X 结构总体布线算法的优化效果不明显，甚至部分结果在线长指标上劣于曼哈顿结构总体布线算法。而且，现有关于 X 结构布线工作属于串行算法，非常依赖于线网的布线顺序或布线代价函数的定义。

针对现有关于 X 结构总体布线算法存在的问题，本文提出了一种 X 结构总体布线器(XGRouter)，主要基于整数线性规划模型、划分策略以及粒子群优化方法。本文所提的整数线性规划模型同时考虑到线长优化和拥挤均衡。基于该模型，XGRouter 能够较好地解决总体布线串行算法存在的布线结果对线网顺序的高度依赖问题，并能得到拥挤相对均匀的布线解。划分策略的基本思想是从最拥挤的布线子区域开始布线，并不断扩张布线区域直至覆盖整个芯片。该划分策略的引入使得 PSO 和 ILP 模型都适合于总体布线问题的求解。从仿真实验结果可看出 XGRouter 可获得相对其他总体布线器而言质量较高的解方案。本文工作是第一

次将并行算法应用于 X 结构总体布线问题的求解，并取得高质量的布线解。

1.5 论文的结构安排

第一章为介绍部分。本章首先介绍本文研究的超大规模集成电路的相关背景，包括研究课题的意义，非曼哈顿结构的背景以及基于非曼哈顿结构下的总体布线所遇到的相关问题。其次，给出了 VLSI 总体布线问题的概述，包括总体布线图和总体布线相关定义的描述。然后，介绍了与本文相关的国内外研究动态及存在的问题，紧接着介绍一些将 PSO 应用于物理设计中各阶段所取得相关研究工作的进展情况，从而阐述了本文研究工作的必要性和重要性。最后，给出本文工作的研究内容及其贡献与创新之处。

第二章是 X 结构 Steiner 最小树构建过程的算法设计与分析。首先，分析了 Steiner 最小树的一些相关工作。接着，介绍了 X 结构 Steiner 最小树的问题模型以及一些相关定义，并给出了本文的算法基础——PSO 算法的基础知识。最后，基于 PSO 的基础上，以线长优化为目标，提出一种构建 X 结构 Steiner 最小树的有效算法。

第三章是时延驱动 X 结构 Steiner 最小树构建过程的算法设计与分析。首先，分析了本章工作涉及到的相关工作。接着，介绍时延驱动 X 结构 Steiner 最小树的问题模型及其涉及到的 Elmore 时延模型和最小线长半径生成树(Minimum-Cost Spanning Tree with a Minimum Radius，MRMCST)模型。最后，在考虑拐弯数减少的基础上，提出了一种有效的时延驱动 X 结构 Steiner 最小树构建算法。

第四章是针对多层绕障 X 结构 Steiner 最小树问题的算法设计与分析。首先，分析了多层绕障 X 结构 Steiner 最小树问题涉及到相关工作的发展历程。接着，介绍多层绕障 X 结构 Steiner 最小树的问题模型及相关定义。然后，基于 PSO 的基础上，引入一些有效的策略，以最小化布线总代价为优化目标，并同时考虑到通孔数的优化，从而提出了一种基于离散 PSO 算法和惩罚机制的多层绕障 X 结构 Steiner 最小树构建算法。最后，进一步引入布线边约束条件，继而构造定向边约束的绕障 X 结构 Steiner 最小树算法，探讨了 X 结构下不同定向边约束模型的优劣。

第五章是针对 X 结构下总体布线问题的算法设计与分析。首先，介绍了本章工作的研究背景。接着，介绍了 X 结构总体布线的问题模型以及相关工作，并提出了 X 结构总体布线问题下一种新颖的整数线性规划模型。最后，提出了一种基于整数线性规划模型、划分策略及 PSO 等技术的高质量 X 结构总体布线算法。

第六章是对本文工作的总结和展望。本章首先总结了全文的主要研究工作，然后对未来的进一步研究方向进行了展望。

第二章 X 结构 Steiner 最小树构建算法

2.1 引言

总体布线是超大规模集成电路物理设计中一个非常重要的步骤。Steiner 最小树(Steiner Minimal Tree, SMT)问题是在给定引脚集合的基础上通过引入一些额外的点(Steiner 点)以寻找一棵连接给定引脚集合的最小代价布线树。因为 Steiner 最小树模型是超大规模集成电路总体布线中多端线网的最佳连接模型[1]，所以 Steiner 最小树构建是超大规模集成电路布线中一个关键环节。目前，大多数布线算法都是基于曼哈顿结构而提出的。随着超大规模集成电路的制造工艺不断发展，互连线效应逐渐成为影响芯片性能的主要挑战。而基于曼哈顿结构的布线模型要求芯片引脚之间的连线只能是水平方向和垂直方向，导致对芯片中互连线的线长优化能力受到限制。因此，能够更充分利用布线资源而带来更强的线长优化能力的非曼哈顿结构吸引越来越多研究人员的兴趣。为了更好地研究非曼哈顿结构结构布线，首要工作是研究非曼哈顿结构 Steiner 最小树的构建问题。非曼哈顿结构 Steiner 最小树主要包含 X 结构 Steiner 最小树[42,43,45]和 Y 结构 Steiner 最小树[41,44]。由于 Steiner 最小树问题是一个 NP 难问题[77]，所以一些在求解 NP 难问题中展现出良好应用前景的进化算法包括 PSO 算法和蚁群算法，被用于求解矩形结构 Steiner 最小树问题(Rectilinear Steiner Minimal Tree, RSMT) [25,61,76]和 X 结构 Steiner 最小树问题[61]。

作为一类基于种群进化的方法，粒子群优化算法于 1995 年由 Eberhart 和 Kennedy 共同提出[78]。PSO 算法相对其他优化算法而言，具有执行操作简单且有能力快速收敛至一个合理、优秀的解方案。近年来，PSO 算法已经广泛并成功应用于很多研究领域，并被许多研究学者通过不断的仿真实验和理论分析等手段证实为一种强有力的优化工具[63-68,79,80]。

基于以上相关研究工作的分析，本章设计并实现一种基于离散粒子群优化的有效算法，用以求解 X 结构 Steiner 最小树的构建问题。首先，为了克服 PSO 算法在应用于高维空间优化问题中收敛速度慢的不足，本章设计一种自适应的参数调节策略用以较好地调整学习因子和惯性权重因子。其次，根据 X 结构 Steiner 最小树问题的特点，结合遗传算法中的变异算子和交叉算子，提出了可用于求解离散问题的 PSO 更新操作策略。然后，设计了一种适合于 X 结构 Steiner 最小树问题的有效编码策略。最后，设计相应的仿真实验，通过实验结果说明本章所提算法的可行性和有效性。

2.2 基础知识

2.2.1 X 结构 Steiner 最小树问题模型

X 结构 Steiner 最小树问题是通过引入 Steiner 点以最小的代价连接给定的引脚集合，这里连接线可走线方向除了包括传统的水平方向和垂直方向，还允许 45°和 135°的绕线方向。

XSMT 问题中给定 $P=\{P_1, P_2, P_3,\ldots,P_n\}$作为待布线网的 n 个引脚集合，且每个 P_i 对应一个坐标对(x_i, y_i)。例如有一个待布线网有 8 个引脚，表 2-1 给出了引脚的输入信息，相应的引脚版图分布如图 2-1 所示，例如，其中编号为 1 的引脚对应的坐标对信息为表 2-1 第二列所示的(33,33)。

表 2-1 待布线网的引脚输入信息

编号	1	2	3	4	5	6	7	8
X 坐标	33	2	42	47	34	38	37	20
Y 坐标	33	9	35	2	1	2	5	4

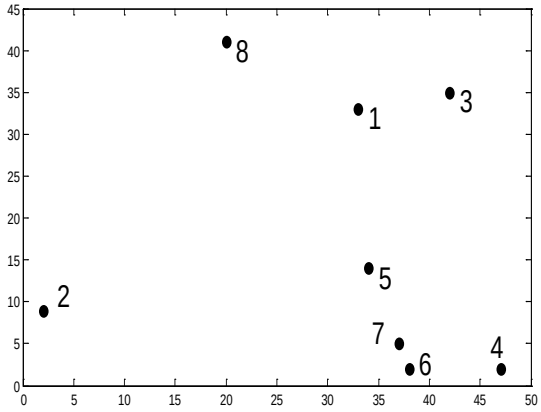


图 2-1 待布线网的引脚分布情况

2.2.2 相关定义

定义 2.1 (pseudo-Steiner 点) 假设除了引脚外的连接点，称为 pseudo- Steiner 点。图 2-2 中 S 为 pseudo-Steiner 点，pseudo-Steiner 点包含 Steiner 点。

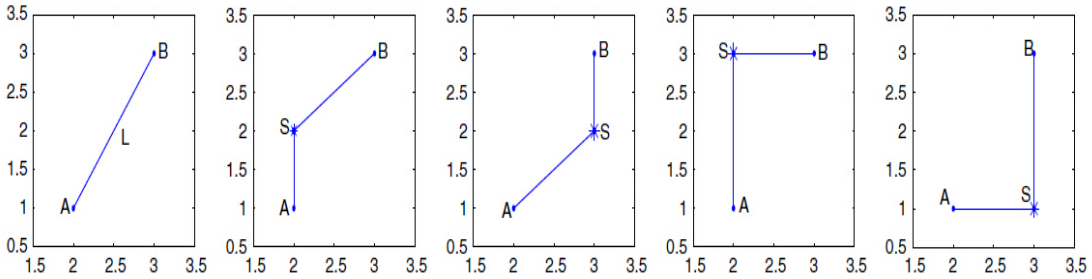
定义 2.2 (0 选择) 如图 2-2(a)所示，(x_1,y_1) 和(x_2,y_2)为边 L 的两个端点 A 和 B 的坐标信息，其中 $x_1<x_2$。边 L 对应的 pseudo-Steiner 点的选择方式如图 2-2 (b)所示，从 A 先引曼哈顿结构边至 S 再从 S 引非曼哈顿结构边至 B，则称作 0 选择。

定义 2.3 (1 选择) 如图 2-2(c)所示，从 A 先引非曼哈顿结构边至 S 再从 S 引

曼哈顿结构边至 *B*，则称作 1 选择。

定义 2.4 (2 选择) 如图 2-2 (d)所示，从 *A* 先引竖直边至 *S* 再从 *S* 引水平边至 *B*，则称作 2 选择。

定义 2.5 (3 选择) 如图 2-2 (e)所示，从 *A* 先引水平边至 *S* 再从 *S* 引竖直边至 *B*，则称作 3 选择。



(a) 线段 *L*　(b) 0 选择　(c) 1 选择　(d) 2 选择　(e) 3 选择

图 2-2 线段 *L* 及其四种边选择方式

2.2.3 基本粒子群优化算法

PSO 算法是一种受到鸟群的社会行为启发的群智能算法，并根据搜索问题产生系列的随机方案作为初始种群。种群中每个方案称为一个粒子，每个粒子可根据自身的情况、粒子历史最佳位置以及整个种群的最佳位置这三方面更新自己的飞行方向，具体的粒子飞行速度更新方式是根据公式(2-1)进行。公式(2-2)是更新粒子的位置，从而飞行至粒子现今所探索到的搜索空间。

$$V_i^{t+1} = w \times V_i^t + c_1 r_1 (P_{pb}^t - X_i^t) + c_2 r_2 (P_{gb}^t - X_i^t) \tag{2-1}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{2-2}$$

其中，V_i^t 表示粒子先前的飞行速度，P_{pb}^t 表示粒子本身历史的最佳位置，P_{gb}^t 表示整个种群搜索至今的最佳位置，t 表示当前的迭代次数，w 是惯性权重参数，c_1 和 c_2 是两个正数(称为加速因子)，r_1 和 r_2 表示取值范围为 0 到 1 的两个随机数。

2.3 基于 PSO 的 X 结构 Steiner 最小树构建算法

由于基本 PSO 算法最初是针对连续问题而设计的，而现实中很多实际应用领域问题都是属于离散问题，所以很多研究学者开始探索和设计能够用于求解离散问题的离散 PSO 算法。这里主要包含三种典型的离散 PSO 算法：(1) Kennedy 和 Eberhart 所提出的二进制离散 PSO 算法[81]，(2) Celec 提出的用于求解旅行商问题(Travelling Salesman Problem，TSP)的离散 PSO 算法[82]，(3) Pan 等提出的用

于流水车间作业调度问题的离散 PSO 算法[83]。此外，文献[71]设计一种有效的离散 PSO 算法用以求解多目标最小生成树问题，并应用于 VLSI 的线长估算。在文献[76]中，提出了一种基于离散 PSO 的矩形 Steiner 最小树构建算法，用以优化布线树的线长，并一定程度上优化拐弯数。基于文献[71]和[76]工作的基础上，本章设计一种构建 X 结构 Steiner 最小树的有效离散 PSO 算法，称为 XSMT_PSO。接下来，本章将从粒子的编码策略、粒子的适应度函数、粒子的更新公式、本章算法的参数设置、本章算法的流程以及本章算法的复杂度分析等六方面详细介绍 XSMT_PSO 算法。

2.3.1 粒子的编码策略

一棵生成树的编码策略通常可包括 Prufer 数编码和边点对编码两种方式。

定义 2.6 (Prufer 数)[84] Prufer 数编码给出了 *n* 个节点的生成树和长度为 *n*-2 的数字串之间的一个对应关系，并习惯地用数字来标志这些节点，所形成的数字串称为 Prufer 数。

Prufer 数编码的长度比边点对编码的长度少三分之二左右，但由于 Prufer 数编码改变编码中一位数字，会对整个生成树的拓扑产生很大的变化，尚不能较好地保留粒子的最优拓扑信息。而边点对编码则能更好地适应粒子群优化算法在迭代过程中保留粒子的最优拓扑信息的特点，且改变边点对编码中一位编码一般不会对生成树的拓扑产生巨大影响，故 XSMT_PSO 算法采用更适合于进化算法的边点对编码方式作为粒子的编码策略。

边点对编码是用一条生成树的边和一位在定义 2.2 至 2.5 中所定义的 pseudo-Steiner 点选择方式表示候选 X 结构 Steiner 最小树的一条边。pseudo-Steiner 点选择方式是将生成树的边转化成 X 结构 Steiner 树(X-architecture Steiner Tree, XST)的 X 结构边。每个 pseudo-Steiner 点选择方式位包含了四种选择方式，即 0 选择、1 选择、2 选择和 3 选择。如果一个待布线网有 *n* 个引脚，则每一棵候选的 XST 包含 *n*-1 条生成树边、*n*-1 位 pseudo-Steiner 点选择方式及一位数字表示粒子的适应度值。又由于一条生成树的边需要两位数字表示该边的两个引脚，所以每个粒子编码的总长度为 3×(*n*-1)+1。例如，一个候选 XST(*n*=8)可用 XSMT_PSO 算法中一个粒子的编码表示，其编码采用如下所示的数字串表示：

7 6 *0* 6 4 *1* 7 5 *1* 5 1 *2* 1 3 *0* 1 8 *1* 5 2 *2* 10.0100

其中，最后一位数字‘10.0100’是表示该粒子的适应度值，而每一位斜体字体的数字则是表示每条边相应的 pseudo-Steiner 点选择方式。第一个数字子串(7 6 *0*)表示一棵 Steiner 树的一条边，该边两个引脚编号为 7 和 6，pseudo-Steiner 点选择方式位是如定义 2.2 所示的 0 选择方式。

2.3.2 粒子的适应度函数

定义 2.7 一颗 X 结构 Steiner 最小树的长度是该布线树中所有边片段的长度总和，其计算方式如下所示。

$$L(T_X)=\sum_{e_i\in T_x} l(e_i) \tag{2-3}$$

其中，$l(e_i)$表示在布线树 T_x 中每个边片段 e_i 的长度。

在计算 XST 中每条边片段的长度总和时，所有边片段分为以下四种类型：水平边片段、垂直边片段、45°斜边片段及 135°斜边片段。算法首先将所有边片段分为以上四种类型，再将其中 45°边片段顺时针方向旋转为水平边，同时将 135°边片段顺时针方向旋转为垂直边，然后将最终的水平边按照其左边引脚的大小从下向上、从左到右的方式排列，同时将垂直边按照其下边引脚的大小从左到右、从下到上的方式排列，最后，XST 的长度即为这些边片段扣除重复部分的总长度之和。

因为粒子群优化算法中粒子的适应度函数值通常是越小代表该粒子越优秀，所以本章算法中的粒子的适应度函数设计如下所示。

$$fitness = 1/(L(T_X)+1) \tag{2-4}$$

其中，分母是布线树的长度加 1，这样是为了防止当布线树的长度可能为 0，即可能两端线网中两个引脚的位置重合这种情况带来的计算无意义问题。

2.3.3 粒子的更新公式

本算法基于遗传算法的交叉和变异操作提出了 XSMT_PSO 算法的粒子更新方式以实现构建 X 结构 Steiner 最小树这一离散问题。粒子的更新公式如下所示。

$$X_i^t = N_3(N_2(N_1(X_i^{t-1},w),c_1),c_2) \tag{2-5}$$

其中，w 是惯性权重因子，c_1 和 c_2 是加速因子，N_1 表示变异操作，N_2 和 N_3 表示交叉操作，假定 r_1, r_2, r_3 都是在区间[0, 1)的随机数。

(1) 粒子的速度更新可表示如下。

$$W_i^t = N_1(X_i^{t-1},w)=\begin{cases} M(X_i^{t-1}), & r_1 < w \\ X_i^{t-1}, & \text{others} \end{cases} \tag{2-6}$$

其中，w 表示粒子进行变异操作的概率。

在公式(2-6)中，变异操作的具体过程：以 2.3.1 小节中 n=8 的候选布线树为例，如图 2-3 所示，取其中的 n-1 位 pseudo-Steiner 点选择方式作为变异序列，即长度为 7 的序列，序列中每个位置只能取 0 至 3 的整数。此时，随机产生一个

0 到 1 区间的数 r_1，若小于变异概率 w，则随机选择序列中一个位置(mp1)，将 mp1 上的值更新为 0 至 3 的整数，但需与 mp1 位置的原值不一样。如图 2-3，mp1 位置上值由 2 更新为 1，此时变异操作结束。

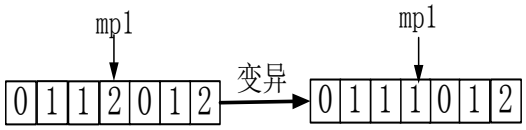


图 2-3 本章算法的变异操作

(2) 粒子的个体经验学习可表示如下。

$$S_i^t = N_2(W_i^t, c_1) = \begin{cases} C_p(W_i^t), & r_2 < c_1 \\ W_i^t, & \text{others} \end{cases} \tag{2-7}$$

其中，c_1 表示粒子与其历史最优方案进行交叉操作的概率。

(3) 粒子与种群其他粒子进行合作学习可表示如下。

$$X_i^t = N_3(S_i^t, c_2) = \begin{cases} C_p(S_i^t), & r_3 < c_2 \\ S_i^t, & \text{others} \end{cases} \tag{2-8}$$

其中，c_2 表示粒子与种群全局最优方案进行交叉操作的概率。

在公式(2-7)和(2-8)中，粒子与其历史最优方案(种群全局最优方案)进行交叉操作的过程：在变异操作后，随机产生一个 0 至 1 的数 r_2(r_3)，若其小于交叉概率 c_1(c_2)，粒子的序列需与其历史最优方案(种群全局最优方案)进行交叉操作。如图 2-4 所示，随机产生两个交叉位置(cp1 和 cp2)，将本粒子中 cp1 和 cp2 的值替换为其历史最优方案(种群全局最优方案)在该区间的值，得到如图 2-4 右半部分所示交叉后的粒子编码情况。

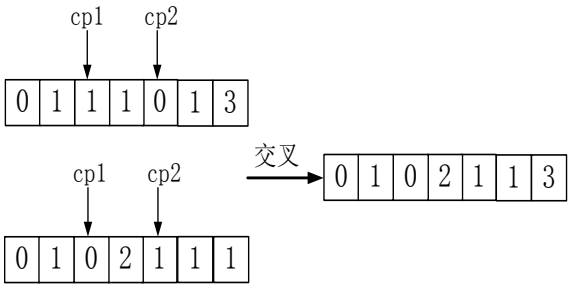


图 2-4 本章算法的交叉操作

2.3.4 XSMT_PSO 算法的参数设置

性质 2.1 惯性权重因子 w 的设置将影响到粒子的局部搜索能力和全局搜索能力的平衡情况。

从粒子的速度更新公式(2-1)中可看出，公式(2-1)的第一部分为粒子提供了在搜索空间的飞行动力，代表粒子在飞行轨道中受先前速度的影响程度。因此，惯性权重因子 w 是反映这种影响程度的具体数值。

性质 2.2 设置值较大的惯性权重因子 w 将使算法拥有更强的全局搜索能力。

从性质 2.1 和公式(2-1)中可看出惯性权重因子 w 是决定粒子保留多少先前的飞行速度。因此一个较大的惯性权重因子将增强粒子搜索未知区域的能力，有助于提高算法的全局搜索能力从而跳出局部最优解。而一个较小的惯性权重因子预示算法将主要集中在当前搜索位置的附近搜索，有助于增强算法的局部搜索能力从而加快收敛速度。

在文献[85]工作中，Shi 和 Eberhart 设计了一种基于线性递减的惯性权重因子调整策略的 PSO 算法。其中，为了确保算法在早期拥有较强的全局搜索能力，采用一个较大的惯性权重因子，同时为了保证后期的收敛速度，即增加局部搜索能力，则后期采用一个较小的惯性权重因子。通过在四种不同的基准测试函数上的实验仿真，最终的实验结果表明基于这种参数调整策略的 PSO 算法能够有效地提高算法的性能。

性质 2.3 值较大的加速因子 c_1 可能导致算法在局部范围内搜索，而值较大的加速因子 c_2 则将使得算法过早收敛至局部最优解。

加速因子 c_1 和 c_2 是用于粒子间的交流学习。Ratnaweera 等提出一种加速因子的调整策略：在算法早期采用值较大的 c_1 和值较小的 c_2，而后期则相反[86]。通过这种方式，可保障算法的早期可在局部范围内进行详细搜索，使其不会在早期直接移动至下一个局部最优位置，同时在后期加快算法的收敛速度。相关实验也表明基于这种调整策略的算法可获得优秀的解方案。

基于以上 PSO 算法的参数性质分析，本章算法的惯性权重因子 w 和两个加速因子 c_1 和 c_2 采用 Shi 等所提的线性递减策略进行设置[85]。具体设置为惯性权重因子 w 从 0.9 线性递减至 0.1，加速因子 c_1 从 0.9 线性递减至 0.2，加速因子 c_2 从 0.4 线性递增至 0.9，每次迭代中相应的参数设置具体按照公式(2-9)，(2-10)和(2-11)进行更新。

$$w = w\_start - \frac{w\_start - w\_end}{evaluations} \times eval \tag{2-9}$$

$$c_1 = c_1\_start - \frac{c_1\_start - c_1\_end}{evaluations} \times eval \tag{2-10}$$

$$c_2 = c_2\_start - \frac{c_2\_start - c_2\_end}{evaluations} \times eval \tag{2-11}$$

其中，*w_start*(c_1*_start*，c_2*_start*)表示参数 *w* (c_1，c_2)迭代开始的初始值，*w_end*(c_1*_ end*，c_2*_ end*)表示参数 *w* (c_1，c_2)的迭代的最终值，*eval* 表示当前迭代次数，*evaluations* 则表示算法的最大迭代次数。

2.3.5 XSMT_PSO 算法的流程

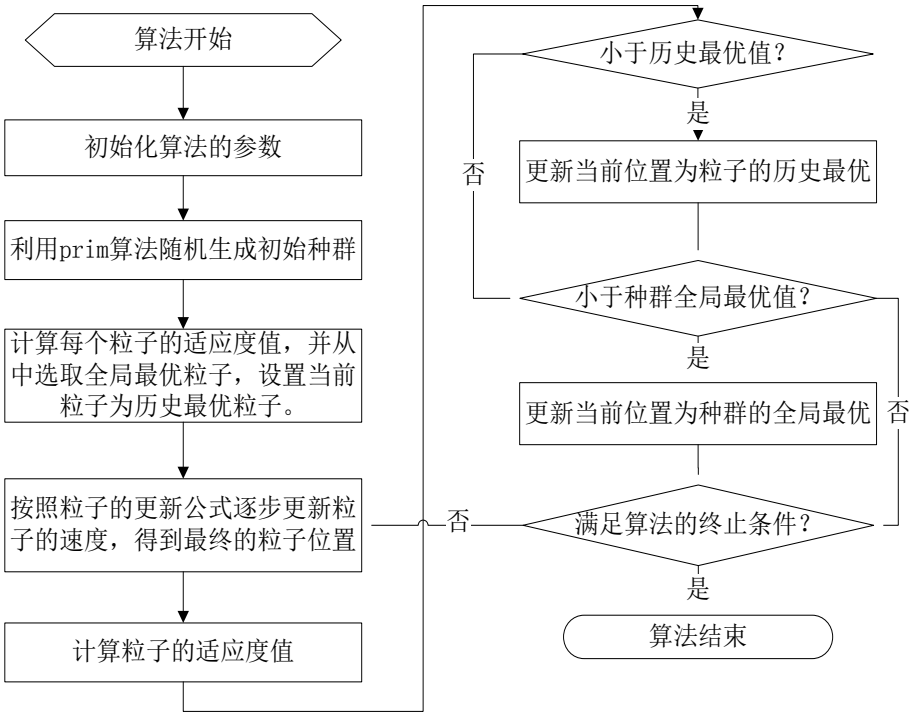


图 2-5 本章算法的流程图

XSMT_PSO 算法的流程图如图 2-5 所示，其详细步骤可概括如下：

步骤 1：初始化 XSMT_PSO 算法中种群大小、迭代次数、惯性权重因子、加速因子等参数的数值，同时随机生成初始种群。

步骤 2：根据公式(2-4)计算每个粒子的适应度值，再从中选择适应度值最小的粒子作为种群的全局最优粒子，并同时将第一代中每个粒子设置为其历史最优粒子。

步骤 3：根据粒子更新公式(2-5)至(2-8)逐步更新粒子的速度，并最终得到更新后的粒子位置。

步骤 4：根据公式(2-4)重新计算每个粒子的适应度值，若当前粒子的适应度值小于其历史最优值，则将更新后的粒子设置为其历史最优粒子。

步骤 5：若更新后的粒子，其适应度值小于种群的全局最优粒子的适应值，

则将更新后的粒子设置为种群全局最优粒子。

步骤 6：检查是否满足算法的终止条件(得到一个足够好的解方案或达到设定的算法最大迭代次数)。如果满足，则算法终止，反之，转到步骤 3。

2.3.6 XSMT_PSO 算法的复杂度分析

引理 2.1 假设种群大小为 p，迭代次数为 $iters$，引脚的个数为 n，则 XSMT_PSO 算法的复杂度为 $O(iters \times p \times n \log n)$。

证明：在 XSMT_PSO 算法的内部循环中，即从步骤 3 至步骤 5，包含变异操作、交叉操作以及适应度值计算操作。在变异和交叉操作中，因为算法只变换 pseudo-Steiner 点的选择方式，时间为常数时间。而在计算适应度的操作中，计算布线树线长的复杂度由排序方法的复杂度决定。因此，算法内部循环的复杂度为 $O(n \log n)$。同时算法外部循环的复杂度主要跟种群的大小以及算法的迭代次数相关，因此 XSMT_PSO 算法的复杂度为 $O(iters \times p \times n \log n)$。

2.4 实验仿真与结果

本章算法的参数设置如下：种群大小为 50，最大迭代次数为 1000。为了验证本章算法的有效性，本章进行两组实验对比，其中本章算法在每个测试实例中各执行 10 次并取最优值。在两组实验的实验结果中，‘RSMT’代表采用文献[76]的算法构造矩形 Steiner 最小树的长度，而‘XSMT’则是采用本章算法构建的 X 结构 Steiner 最小树的长度。在第一组实验中，采用 OR-Library 测试数据[87]用以对比文献[76]和本章算法构造 Steiner 最小树的长度。如表 2-2 所示，给出了在 46 个测试实例中本章算法所构造的 XSMT 的长度值和文献[76]算法所构造的 RSMT 的长度值，以及它们的对比改进情况。如表 2-2 所示，可发现本章算法构造的 XSMT 的长度值相对于文献[76]算法所构造的 RSMT 的长度值取得了平均 9.84% 的优化率。而文献[42]提出了精确算法构建 XSMT，并指出了 XSMT 的长度一般比 RSMT 的长度优化 10%左右，同时文献[42]的测试实例也是来源于 OR-Library 测试数据，并得到 XSMT 长度比 RSMT 长度的优化率范围一般为 $9.75 \pm 2.29\%$。而本章算法的平均优化率为 9.84%是在该优化率范围内。

第二组实验中，本章使用文献[88]中 *rand_points* 产生 10 个随机测试实例，并分别构造相应的 RSMT 和 XSMT。每个测试实例的规模和实验结果分别在表 2-3 中给出，从中可看到本章算法构造的 XSMT 取得了 9.28%的长度优化率。该组实验的平均线长优化率也在文献[42]中给出 $9.75 \pm 2.29\%$的优化率范围内。

表 2-2 在 46 个测试实例中 XSMT 和 RSMT 的长度对比情况

编号	RSMT	XSMT	减少率(%)
1	1.87	1.7646	5.64
2	1.68	1.5664	6.76
3	2.36	2.1855	7.39
4	2.54	2.2382	11.88
5	2.29	2.1385	6.62
6	2.48	2.3285	3.79
7	2.54	2.3475	7.58
8	2.42	2.2685	6.26
9	1.72	1.6543	3.82
10	1.85	1.7057	7.80
11	1.44	1.3511	6.17
12	1.80	1.6828	6.51
13	1.50	1.3243	11.71
14	2.60	2.3657	9.01
15	1.48	1.2895	12.87
16	1.60	1.2485	21.97
17	2.01	1.6920	15.82
18	4.06	4.0566	0.08
19	1.93	1.8128	6.07
20	1.12	1.0685	4.60
21	2.16	1.9188	11.17
22	0.63	0.5363	14.87
23	0.65	0.5277	18.81
24	0.30	0.2680	10.67
25	0.23	0.2097	8.83
26	0.15	0.1290	14.00
27	1.33	1.2070	9.25
28	0.28	0.2156	23.00
29	2.00	1.4728	26.36
30	1.10	1.1000	0.00
31	2.66	2.4870	6.50
32	3.30	3.0381	7.94
33	2.69	2.3317	13.32
34	2.54	2.2682	10.70
35	1.57	1.4243	9.28
36	0.90	0.9000	0.00
37	0.90	0.8070	10.33
38	1.66	1.4808	10.80
39	1.66	1.4808	10.80
40	1.62	1.5204	6.15
41	2.24	2.0634	7.89
42	1.53	1.3833	9.59
43	2.68	2.5443	5.76
44	2.61	2.2910	12.22
45	2.26	2.0881	7.61
46	1.50	1.5000	0.00
AVG			9.84

表 2-3 在随机生成的 10 个测试实例中 XSMT 和 RSMT 的长度对比情况

实例	引脚数	RSMT	XSMT	减少率(%)
1	8	17931	17040	5.0
2	9	20503	18163	11.4
3	10	21910	19818	9.5
4	20	35723	32199	9.9
5	50	53383	47960	10.2
6	70	61987	55980	9.7
7	100	76016	68743	9.6
8	410	156520	142880	8.7
9	500	170273	154290	9.4
10	1000	245201	222050	9.4
AVG				9.28

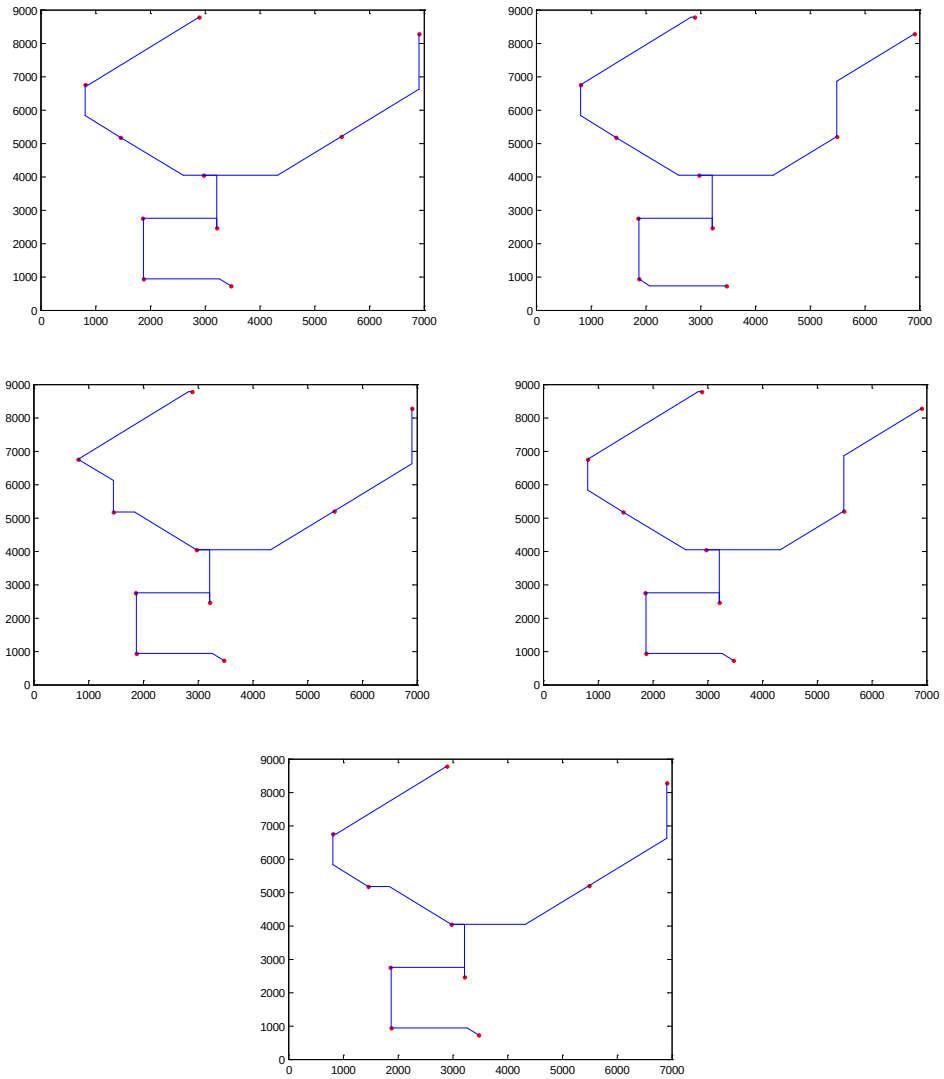


图 2-6 引脚数为 10 的实例可取得线长一样的五种不同拓扑结构

通过实验结果观察到，本章算法在相同或相近最优值的条件下可获得 XSMT 的多种不同拓扑结构。而这种可获得多种不同拓扑结构 Steiner 树的实验效果，有助于在 VLSI 总体布线阶段进行拥挤度优化工作提供不同的拓扑选择方案。图 2-6 给出了一个测试实例，其引脚数为 10 个，通过多次运行本章算法可以得到五种不同拓扑结构(总共可得到拓扑的种类不止五种，在图 2-6 中简单给出五种拓扑)，而且它们的长度值都是一样的。

综上所述，两组实验均可取得跟文献[42]相似的线长优化率，而且本章算法还可得到多种不同拓扑的 RSMTs，这些不同拓扑对 VLSI 总体布线的拥挤度优化工作可带来一定的帮助和指导。

2.5 小结

本章基于 PSO 提出了一种有效的 XSMT 构造算法，以线长为优化目标，相对矩形 Steiner 最小树取得较为不错的线长优化率，并且可获得多种不同拓扑的 XSMTs，从而有利于 VLSI 总体布线阶段的拥挤度优化。最终的实验结果表明本章算法的可行性和有效性。

第三章 时延驱动 X 结构 Steiner 最小树构建算法

3.1 引言

总体布线是 VLSI 物理设计中极为重要的环节之一。Steiner 最小树的构造问题是通过引入一些 Steiner 点用以连接给定的引脚集合，从而达到得到的生成树代价最小。Steiner 最小树是总体布线阶段为芯片中每个线网进行布线的基础模型。近年来，总体布线的优化目标包括互连线的长度、互连线延迟、拥挤度、通孔数等[89,90]。

早期关于 Steiner 最小树的构造方法主要集中在以互连线的线长最小化的优化工作中[91-93]。近年来，随着芯片特征尺寸的不断缩小，互连线延迟已经成为纳米领域中 VLSI 布线的一个非常关键的指标。因此，一些以互连线延迟为优化目标的性能驱动总体布线工作已经开展。Cong 等提出一种效果不错的时延驱动 Steiner 树构造算法，可以达到互连线延迟的优化[94]。Seo 和 Lee 提出一种最小线长半径生成树模型用以构建性能优化的布线树[95]。Hou 等提出了一种性能驱动总体布线模型，其优化目标是满足给定每个漏点的时延约束值，并引入 Steiner 点的变换操作以减少互连线延迟[96]。Yan 基于重新调度 Steiner 点位置的思想，提出了一种有效的调度算法用以满足时延约束[97]。

通常拥挤度和通孔数是影响芯片质量的重要指标，从而影响芯片的可布线性。Sarrafzadeh 等设计了一种单层总体布线器，可以控制拥挤的情况和减少拐弯数[26]。一个拐弯数在层调度或详细布线阶段通常意味着布线层之间的一个转换，从而导致需要使用一个甚至多个通孔进行连接，所以减少拐弯数有利于布线后期通孔数的减少工作。因此，Liang 等提出了一种基于引力方向的矩形 Steiner 最小树构造算法用以减少拐弯数[98,99]。该方法借鉴万有引力计算公式，通过计算一个引脚受到其他所有引脚的引力以判断引脚下一步走线的方向。Bozorgzadeh 等定义了灵活度的概念，并于此提出一种提高芯片可布线性的有效算法，其输入是一棵 Steiner 树，输出是一棵具有更强灵活度的 Steiner 树，从而增强芯片的可布线性[27]。Yan 也探索了在一棵布线树中布线边的灵活度的改进，以期缓和布线拥挤度[97]。

然而以上相关工作均是基于曼哈顿结构 Steiner 树的构造，尚未涉及非曼哈顿结构。随着 VLSI 制造工艺的不断发展，互连线效应成为影响芯片性能的主要因素。曼哈顿结构限制布线方向只能是水平方向和垂直方向，从而影响了布线对线长和时延的优化能力。因此，能够允许更多布线方向的非曼哈顿结构布线工作得

到越来越多的关注和研究，以期能进一步提高芯片的可布线性[42-45,100]。

基于非曼哈顿结构布线的研究工作[42-45,100]，其优化目标大多局限于布线树长度的优化，而对于非曼哈顿结构布线问题，互连线延迟对芯片性能有重大影响，因此有必要构造时延驱动的 Steiner 最小树，但相关研究成果较为匮乏。Yan 提出了一种基于 Steiner 点调度和边变换思想的时延驱动 X 结构 Steiner 最小树构造算法[47]。Samanta 等在 2011 年基于两电极(Two-pole)和 Elmore 时延估算模型构造了一种近似最优的时延驱动 Y 结构 Steiner 最小树[48]。但前者所构造的时延 X 结构驱动 Steiner 最小树，其质量受到 Steiner 点的优化顺序和初始解质量的影响非常大，从而容易陷入局部最优情况。而后者的工作则侧重于时延估算模型的研究，并且是基于 Y 结构。此外，通孔数最小化能减少互连线延迟且可进一步加强芯片的可制造性。而减少拐弯数有助于后期通孔数的优化工作，同时在布线前期减少拐弯数的工作相对于后期详细布线减少通孔数的工作来得容易。因此，在 Steiner 最小树算法的研究中有必要考虑到拐弯数的优化工作。目前关于非曼哈顿结构下时延驱动 Steiner 最小树的相关研究中尚未有考虑拐弯数优化的工作。

综上，本章基于多目标粒子群优化设计一种有效的算法以构建一棵时延驱动的 X 结构 Steiner 树，并同时考虑到拐弯数的优化，该算法称之为 TXST_BR_MOPSO。本章工作的贡献之处如下所示：

(1) 相对于文献[100]中只考虑 Steiner 点的变换，本章算法设计了边变换策略，用以增强 PSO 算法中粒子的进化能力。本章算法同时考虑到 Steiner 点和布线边的变换工作，相对于文献[100]能够进一步加强布线树的线长优化能力，从而取得一定的线长优化。而且边变换策略对 PSO 算法中粒子拥有向 MRMCST 模型进化的能力是必不可少的。

(2) 基于 pseudo-Steiner 点及其灵活度的定义，本章工作通过四种边选择方式重新调度 pseudo-Steiner 点的位置，提出了一种有效的编码策略用以优化布线树的拐弯数。同时边变换策略可很好结合所提出的边点对编码策略以减少拐弯数。本章工作是非曼哈顿性能驱动总体布线工作中第一次考虑到拐弯数的优化。实验结果表明本章算法在引入边变换策略后可取得将近 20%的拐弯数减少率。

(3) 本章工作基于 X 结构布线边的距离计算方式，提出相应的 MRMCST 模型用以寻找并构造 X 结构下的时延驱动布线树。为了能够同时优化 MRMCST 模型中线长和半径两个互为竞争的目标，本章工作基于多目标 PSO 模型设计相应的多目标算法以构造最终的时延驱动 X 结构布线树。实验结果表明最终的布线树相对于 RSMTs[76]和 XSMTs[100]两类算法在最大源漏点时延和总的源漏点时延两个时延指标上能够取得较大的优化率。而且在 0.18μm 的 IC 制造工艺(tost IC 工艺)上，本章算法构造的最终布线树相对于 Yan 所构造的时延驱动 X 结构 Steiner

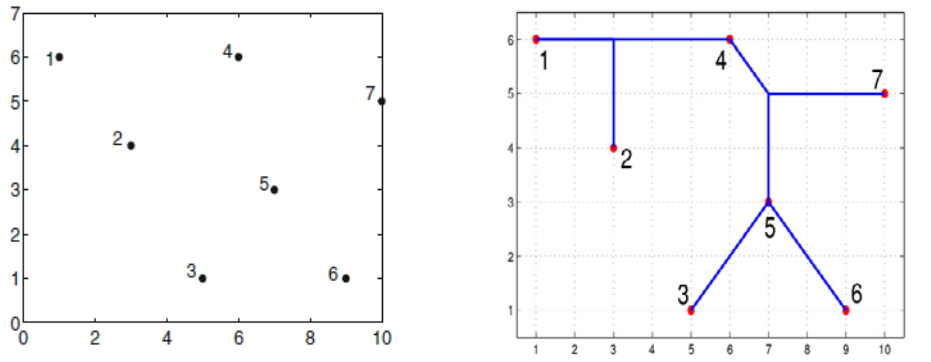
最小树[47]在线长、半径、最大源漏点时延、总的源漏点时延以及拐弯数等五个性能指标上分别能够取得 2.08%、5.22%、4.93%、10.07%、19.67%的优化效果。

3.2 问题模型

3.2.1 时延驱动 X 结构 Steiner 最小树问题

时延驱动 X 结构 Steiner 最小树问题是通过引入 Steiner 点以最小的代价连接给定的引脚集合，连接线可走线方向除了包括传统的水平方向和垂直方向，还允许 45°和 135°的绕线方向，其中 Steiner 树的代价是指时延。本章算法通过优化 Steiner 树的线长和半径以优化最终布线树的最大源漏点时延和所有源漏点时延之和，以获得最终的时延驱动 X 结构 Steiner 最小树。

如图 3-1(a)所示，给出一个初始的引脚分布情况，表 3-1 是对应的各引脚坐标信息，其中编号为 4 的引脚为源点，其他引脚则为漏点，图 3-1(b)则是构建一种时延驱动 XSMT 的布线方案。



(a) 引脚的分布图　　(b) 一种时延驱动 XSMT 的布线方案

图 3-1 时延驱动 XSMT 的问题模型

表 3-1 图 3-1 所示布线实例中引脚的坐标信息

引脚编号	1	2	3	4	5	6	7
X 坐标	1	3	5	6	7	9	10
Y 坐标	6	4	1	6	3	1	5

3.2.2 Elmore 时延模型

本章采用 Elmore 时延模型[101,102]对时延指标进行计算，该时延模型可描述如下。给定一个线网 N 及其相应的一棵布线树 $T(N)$，令 e_i 表示引脚 i 至其所在布线树 $T(N)$中父节点的一条边，其电阻和电容分别用 r_{e_i} 和 c_{e_i} 表示。令 $T(N)_i$ 表示一棵源点为 i 在布线树 $T(N)$ 中的子树，而 C_i 表示集总电容，它是该子树 $T(N)_i$ 中所

有漏点电容和所有边电容的总和。令 n_0 表示源点，其电容和电阻分别用 R_0 和 C_0 表示。最后，在漏点 i 的 Elmore 时延可按照如下公式进行计算。

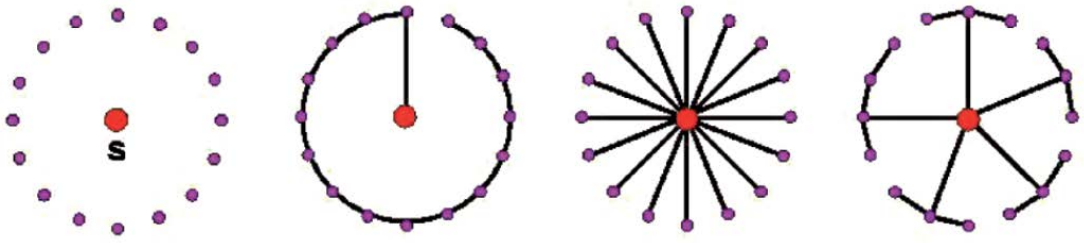
$$t_i = R_0 \times C_0 + \sum_{e_j \in path(s,i)} r_{e_j} \times (c_{e_j} / 2 + C_j) \tag{3-1}$$

其中，$path(s, i)$是表示从源点 s 到漏点 i 路径上的所有边集合。

Elmore 时延模型有如下的优势。首先，估算所有漏点到源点的 Elmore 时延值的时间复杂度为 $O(n)$，其中 n 为引脚个数。因此 Elmore 时延模型的计算量相对较少。其次，Elmore 时延模型的计算准确度高于线性时延模型。最后，基于 Elmore 时延模型所得到的近似最优或最优布线树相对基于其他高阶时延模型具有同等的优越性。因此，本章采用公式(3-1)的 Elmore 时延模型计算布线树中源漏点之间的时延值。

3.2.3 MRMCST 模型

如图 3-2(a)所示，给定一个引脚集合，这些引脚的分布比较特殊，中间有一个源点(称为 S)，引脚集合中其他引脚均匀分布在以源点为中心的一个圆周上，以此例子为基础可形象地区分对比三种不同类型的布线树。如图 3-2(b)所示的树是最小生成树(Minimum Spanning Tree，MST)，拥有最小的线长，而线长是 VLSI 布局和布线的一个重要性能指标。然而最小生成树容易给布线带来冗长的关键路径(称为半径)，从而带来较大的时延，降低芯片的性能。如图 3-2(c)所示是最小路径树(Shortest Path Tree，SPT)，拥有最短的关键路径，但是其线长却远大于 MST。而一棵布线树的半径意味着芯片设计中最大源漏点时延，同时其线长的最小化意味着最小化驱动器的输出电阻和互连线的总电容。因此，线长和半径两个指标对芯片总体性能的优化都有重要的影响，故对芯片性能的改善，需要同时优化这两个指标[89]。



(a) 引脚的分布情况　(b) MST　(c) SPT　(d) MRMCST

图 3-2 给定引脚集合的不同类型连接拓扑[89]

如图 3-2(d)所示的 MRMCST 模型可以很好地权衡线长和半径，给出二者的

折衷方案，因此，MRMCST 模型可用于时延驱动布线树的构建[89]。因为求解 MRMCST 模型是一个 NP 难问题[95]，所以本章工作基于多目标 PSO 方法构建时延驱动 X 结构 Steiner 最小树算法，以同时优化线长和半径，从而为布线获得更好的芯片性能。

3.3 TXST_BR_MOPSO 算法

鉴于 VLSI 物理设计的重要性和 PSO 的优势，已有相关工作将 PSO 应用于电路划分、布图规划以及布线等方面，并取得一定的研究成果。文献[100]也开展关于非曼哈顿结构布线的初步工作，并设计一种有效的 X 结构 Steiner 最小树构建算法，但该方法只对线长这一目标进行优化，并未考虑到时延等一些其他重要的优化指标。此外，文献[103,104]已经成功构建了多目标 PSO 算法的基础框架，并提出了相应的更新操作算子和适应度函数。

基于以上相关工作的基础上，本章设计了一种有效的多目标 PSO 算法，用以求解时延驱动 X 结构 Steiner 最小树问题并同时考虑到拐弯数的优化。在该算法中，表现型共享函数应用于适应度函数的定义以期同时优化线长和半径这两个目标。一种有效的边变换策略能很好结合本章的编码策略，可实现在时延驱动 X 结构 Steiner 最小树构建过程中进一步考虑拐弯数的优化。

3.3.1 性能驱动总体布线的两个重要指标

线长和半径是性能驱动总体布线的两个重要指标。为每个线网构建的布线树，其线长和半径对最后的时延都有直接的影响，因此，本章引入 MRMCST 模型以构建时延驱动 Steiner 最小树，这对性能驱动总体布线工作是非常必要的。不同于 Ho 等是基于 Euclidean 距离构建 MRMCST 模型且未给出相关实验结果[89]，本章提出了 X 结构 MRMCST 模型，其中源漏点的长度计算公式采用 X 结构边的距离进行计算，并给出相应的实验结果。引脚 i 和 j 的 X 结构距离(称为 $Odis(i,j)$)是按照如下公式进行计算。

$$Odis(i,j)=\begin{cases}(\sqrt{2}-1)\min(|x_i-x_j|,|y_i-y_j|)+\max(|x_i-x_j|,|y_i-y_j|), if\ pspc=0\ \text{or}\ 1\\ |x_i-x_j|+|y_i-y_j|, else\end{cases} \tag{3-2}$$

其中，$x_i(x_j)$和 $y_i(y_j)$分别表示引脚 i (j)的水平坐标和垂直坐标。$|x_i-x_j|$($|y_i-y_j|$)表示绝对值，而符号 min(max)表示$|x_i-x_j|$和$|y_i-y_j|$之间的最小值(最大值)。

定义 3.1 X 结构 Steiner 最小树的长度是树中每个 X 结构边片段长度之和，可按照如下公式计算。

$$L(T_X)=\sum_{e_i\in T_x} l(e_i) \tag{3-3}$$

其中，$l(e_i)$表示在布线树 T_x 中 X 结构边片段 e_i 的长度。

定义 3.2 一棵 X 结构 Steiner 最小树的半径是指从源点到各个漏点的 X 结构距离的最大值。

性能驱动总体布线主要的优化目标是时延，而因为 Elmore 时延模型是芯片设计中估算时延值的一种好方法，所以本章是基于 Elmore 时延模型按照公式(3-1)计算布线树中每对源漏点之间的时延值。令 m_j 和 n_j 为布线边 e_j 的两个端点，$r_0(c_0)$表示单位长度的电阻(电容)，则更为详细的计算时延值的公式如下所示。

$$t_i = R_0 \times C_0 + \sum_{e_j \in path(s,i), e_j = (m_j, n_j)} Odis(m_j, n_j) \times r_0 (Odis(m_j, n_j) \times c_0 / 2 + C_j) \qquad (3\text{-}4)$$

线长和半径的优化对最终时延最小化都是同等重要的，然而这两个指标经常是互相矛盾，处于竞争关系[89]。当尝试最小化线长，往往半径变得很大，反之亦然。因此，本章设计多目标 PSO 算法以期更好地同时优化这两个指标。

3.3.2 多目标算法

不管是在曼哈顿结构还是非曼哈顿结构的 VLSI 布线问题都是一个复杂的多目标优化问题。在纳米级 VLSI 布线工作中经常需要同时考虑到一些处于竞争关系且不能同时优化的电学性能指标和几何约束，从而增加纳米级 VLSI 布线的复杂度。因此，如何为纳米级 VLSI 布线问题构建一个有效且稳定的多目标优化算法是一个非常重要的问题。

传统用于多目标优化问题的算法是给每个目标分配一个正系数将多个目标聚合为单个目标，再用现有成熟的单目标优化算法进行求解，其中正系数的大小是由决策者根据实际情况定义的。这些方法包括加权和方法、约束法、目标规划方法等，是以相对较小的计算成本寻找较优解。然则这些方法存在以下不足：(1)因为决策者很难获得关于这些问题的先验知识导致系数设定的困难，从而无法构造合适的算法。(2)使用这些方法求解相关问题，很容易受解空间的非劣前端分布的影响，不能有效解决非劣前端是凹形分布的这类问题。(3)使用这些方法求解多目标问题，每个优化过程中获得非劣解集合是独立的，从而导致决策者很难做出合理的选择。而文献[103,104]所构建的多目标 PSO 框架，则在用于求解多目标优化问题中较好地克服这些问题，取得较为理想的多目标方案。

因此，本章工作将使用多目标 PSO 算法构建时延驱动 X 结构 Steiner 最小树，以优化线长和半径两个目标，同时减少拐弯数，从而解决非曼哈顿结构性能驱动总体布线问题。一些关于多目标 PSO 的基础知识描述如下。

定义 3.3 (目标距离 fd_{ij}) fd_{ij} 表示两个粒子 i 和 j 之间的距离。假设粒子之间的距离包括 m 维，每一维分别标记为 $f_1 d_{ij}, f_2 d_{ij}, \ldots, f_m d_{ij}$，则

$$fd_{ij} = f_1 d_{ij} + f_2 d_{ij} + ... + f_m d_{ij} = \left| f_1(\mathbf{x}^i) - f_1(\mathbf{x}^j) \right| + \left| f_2(\mathbf{x}^i) - f_2(\mathbf{x}^j) \right| + ... + \left| f_m(\mathbf{x}^i) - f_m(\mathbf{x}^j) \right|, \ i \neq j. \tag{3-5}$$

其中，本章算法中考虑到两个优化目标，故 m=2，分别标记为 f_1 和 f_2。其中 f_1 表示布线树的线长函数，f_2 表示布线树的半径函数。

定义 3.4 (支配度量 *D*(*i*)) *D*(*i*)表示在当前种群中支配粒子 *i* 的总粒子数，计算公式如下。

$$D(i) = \sum_{j=1}^{p} nd(i, j) \tag{3-6}$$

其中，*nd*(*i*, *j*)表示如果粒子 *j* 支配粒子 *i* 的话，则值为 1，否则值为 0。

定义 3.5 (共享函数 $sh(fd_{ij})$)

$$sh(fd_{ij}) = \begin{cases} 1, & \text{if } fd_{ij} \le \sigma_s \\ 0, & \text{otherwise} \end{cases} \tag{3-7}$$

其中，σ_s 是共享参数。

定义 3.6 (邻近密度度量 *N*(*i*)) 粒子 *i* 相应的邻近密度度量值 *N*(*i*)的计算公式如下。

$$N(i) = \sum_{j=1}^{p} sh(fd_{ij}) \tag{3-8}$$

定义 3.7 (适应度函数 *F*(*i*))一个给定粒子的适应度函数 *F*(*i*)计算公式如下。

$$F(i) = (1 + D(i))^{\alpha} \times (1 + N(i))^{\beta} \tag{3-9}$$

其中，α 和 β 是非线性参数。

相对于单目标 PSO 算法，在多目标 PSO 算法的搜索过程中，经常可得到不止一个的粒子历史最优方案和种群全局最优方案。类似文献[105]的方法，本算法用 A1 和 A2 两个外部文档来保留这些方案。其中文档 A1 存储当前种群的帕累托前端作为候选集合以指导粒子的飞行，同时 A2 用于存储历代种群的帕累托前端。在更新粒子位置的过程中，一些较优粒子被随机选择用以指导其他粒子的飞行。当每次更新完成后，候选解集合 A1 便会更新。为了能够避免外部文档的规模变得非常大，本章采用 ε-支配方式[106]以控制外部文档的规模大小。

3.3.3 编码策略

本章算法所采用的边点对编码是用一条生成树的边 pseudo-Steiner 点选择方式以表示候选 TXST 的一条边。pseudo-Steiner 点选择方式是将生成树的边转化成 TXST 的 X 结构边。每个 pseudo-Steiner 点选择方式位包含了四种选择方式，

即 0 选择、1 选择、2 选择和 3 选择。如果一个待布线网有 *n* 个引脚，则每一棵候选的 TXST 包含 *n*-1 条生成树边、*n*-1 位 pseudo-Steiner 点选择方式及三位额外的数字，这额外三位数字分别表示粒子适应度值、线长值、半径值。又由于一条生成树的边需要两位数字表示该边的两个引脚，所以每个粒子编码的总长度为 3(*n*-1)+3。例如，一个引脚数为 7 的 TXST 的候选布线树可用 PSO 中一个粒子表示，其编码如下所示的数字串表示。

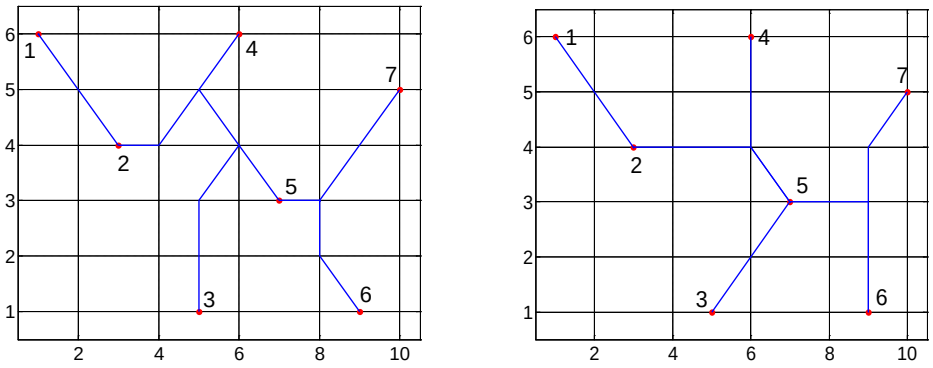
5 7 2	5 4 0	2 4 1	5 3 1	2 1 1	5 6 0	72	18.7279	11.0711

其中，数字‘72’是粒子的适应度值，即目标空间的表现型共享函数值。最后两位数字‘18.7279’和‘11.0711’分别是布线树的线长值和半径值。每个数字子串的第三位数字是 pseudo-Steiner 点选择方式。比如，第一个数字子串(5 7 2)表示布线树的一条边，其两个端点的编号分别为 5 和 7，pseudo-Steiner 点选择方式位是第三位数字‘2’。

定义 3.8 (拐弯点) 布线树中一个点的度数如果大于 1，则定义为拐弯点，拐弯点可能是 Steiner 点、拐角处的点、引脚端点。

近年来随着 VLSI 特征尺寸的不断缩短，通孔数成为总体布线阶段中一个关键的指标。通孔数的优化可减少电路时延并加强芯片的可制造性。而减少拐弯数有助于通孔数的减少，因为在层调度阶段或详细布线阶段中，一个拐弯通常意味着布线层之间的一个转换，这个转换可能需要多个通孔的连接。减少拐弯数相对后续阶段减少通孔数来得容易，因此，需要在 Steiner 树的构建过程中考虑拐弯数的减少。

Yan 在文献[47]中采用了类似本章算法编码中的两种 pseudo-Steiner 点选择方式，包括 0 选择和 1 选择，以表示布线边的走线方式。相比而言，本章算法的编码策略采用四种 pseudo-Steiner 点选择方式拥有对拐弯数进行优化的潜能。这是因为额外引入 2 选择和 3 选择两种方式，有可能与 0 选择和 1 选择两种方式的走线在拐弯点的位置重合，从而有可能进一步优化拐弯数。比如，如图 3-3(b)所示，采用四种 pseudo-Steiner 点选择方式的编码策略令边(4,2)和边(4,5)在拐弯点的位置重合，而如图 3-3(a)所示，采用两种 pseudo-Steiner 点选择方式的编码策略，这两条边都有各自不同位置的拐弯点(编码为 4 的引脚在文献[47]中定义为线网的源点)。而这种情况会在引脚规模较大的线网中多次出现。因此，在大多数电路中，采用如图 3-3(b)所示的编码策略并结合相应的边变换策略，会比采用如图 3-3(a)所示的编码策略在优化拐弯数的能力更强。在本章的后续实验部分会进一步通过实验结果给出验证说明。



(a)文献[47]中两种边选择方式的编码策略　(b)本章算法中四种边选择方式的编码策略

图 3-3 两种不同编码策略的拐弯数情况

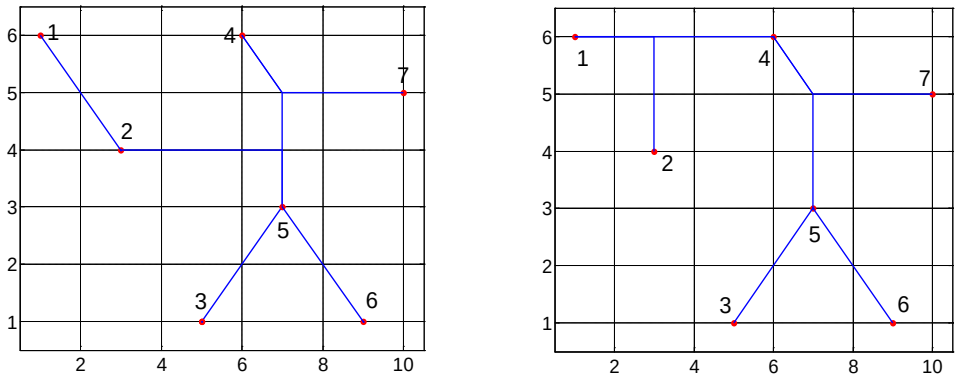
3.3.4 粒子的更新公式

因为 Steiner 树的构建问题是一个离散问题，本章采用一种基于遗传算子的离散位置更新方式，从而提出了 TXST_BR_MOPSO 算法以构建时延驱动 X 结构 Steiner 树，并考虑到拐弯数的优化。

粒子的更新公式具体可表示成如下公式。

$$X_i^t = N_3(N_2(N_1(X_i^{t-1}, w), c_1), c_2) \tag{3-10}$$

其中是 w 惯性权重因子，c_1 和 c_2 是加速因子，N_1 表示变异操作，N_2 和 N_3 表示交叉操作。这里假定 r_1, r_2, r_3 都是在区间[0, 1)的随机数。



(a)布线树拓扑一　(b) 布线树拓扑二

图 3-4 在 MRMCST 构建算法的进化过程中边变换策略的必要性

在构建 X 结构 Steiner 最小树的工作[100]中，更新操作只考虑到 Steiner 点变换。然而，在构建 MRMCST 的过程中，布线树的拓扑是变化的。例如，如图 3-4 所示的两张图分别是构建 MRMCST 的进化过程中存在两种布线树拓扑，两者的拓扑是不一样的，其中边(4,2)存在于如图 3-4(b)所示的布线树中，而在如图 3-4(a)

所示的布线树中却不存在。因此在 MRMCST 构建算法的更新操作中，除了 Steiner 点变换操作外，仍需引入边变换操作。而且边变换操作的引入可进一步增强线长的优化能力，同时在更新操作中引入边变换策略，可保证文献[76]所提算法的搜索空间包含最优解方案。

边变换操作的引入使得在算法迭代过程中布线树可能会出现环的情况，导致无效解的产生，从而破坏粒子编码的强壮性。为了解决边变换操作带来的不足，本章算法将并查集策略[107]融入到如下所示的三个更新操作中。

(1) 粒子的速度更新可表示如下。

$$W_i^t = N_1(X_i^{t-1}, w) = \begin{cases} M(X_i^{t-1}), & r_1 < w \\ X_i^{t-1}, & \text{others} \end{cases} \tag{3-11}$$

其中，w 表示粒子进行变异操作的概率。在粒子的变异过程中，本章算法随机选择准备变异的一条边进行删除，删除之后布线树被分为两棵子树。本章算法使用并查集分别记录两棵子树的引脚集合，并从两个引脚集合中分别选取一个引脚作为变异后新布线边的两个端点。变异的过程如图 3-5 所示，其中边 M_1 表示删除的边，边 M_2 表示变异后的新布线边。变异算子的伪代码如算法 1 所示。

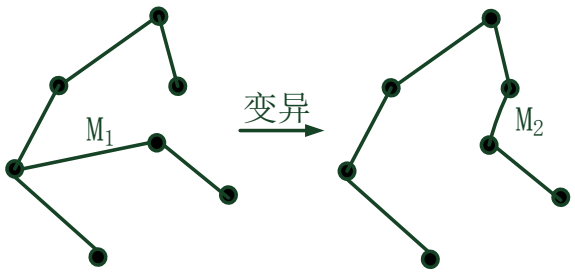


图 3-5 引入并查集策略的变异操作

(2) 粒子的个体经验学习可表示如下。

$$S_i^t = N_2(W_i^t, c_1) = \begin{cases} C_p(W_i^t), & r_2 < c_1 \\ W_i^t, & \text{others} \end{cases} \tag{3-12}$$

其中，c_1 表示粒子与其历史最优方案进行交叉操作的概率。在粒子跟其历史最优位置进行交叉，本章算法选取它们相应的布线树之间的共同边集合作为交叉后新布线树的一部分边，将剩余的边存储在一个集合，称为剩余边集合。然后从剩余边集合中不断随机选取边作为交叉后新布线树的边，直至构成一棵完整的新布线树，这个过程中采用并查集策略避免在随机选取边的过程中，出现环的情况。交叉的过程如图 3-6 所示，其中 C_1，C_2，C_3，C_4，C_5，C_6 表示两个布线树的剩余边集合，C_1，C_3，C_5 表示随机选取后布线树的新边集合。交叉算子的伪代码如算法 2 所示。

算法 1　Mutation operator(*p*)

输入：待变异的粒子 *p*

输出：变异后的新粒子

```
Initialize();//初始化每个粒子的 partition 为单元素集合
r=random(1,n-1);//随机产生 1 到 n-1 的数，n 代表引脚的个数，e_r 表示随机边
for each edge e_i of p do
  if e_i ≠ e_r then
    Union_partition(u,v);//u 和 v 是 e_i 的两个端点，u 和 v 合并到同一个集合中
  end if
end for
while true do
  p_1= random(1,n-1);
  p_2= random(1,n-1);
  if Find_set(p_1) ≠ Find_set(p_2) then
    Union_partition(p_1,p_2);//若 p_1 和 p_2 不在同一集合则合并
    Generate_edge(p_1,p_2);
    break;
  end if
end while
```

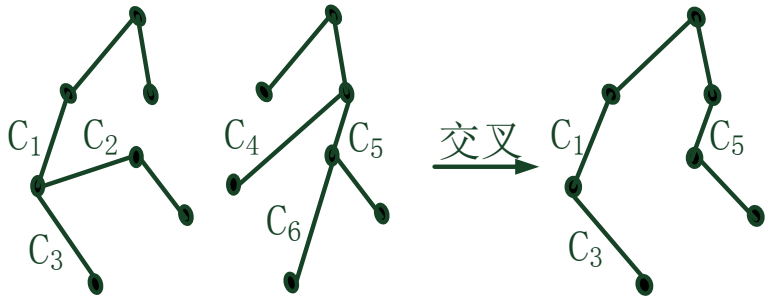


图 3-6 引入并查集策略的交叉操作

(3) 粒子与种群其他粒子进行合作学习可表示如下。

$$X_i^t = N_3(S_i^t, c_2) = \begin{cases} C_p(S_i^t), & r_3 < c_2 \\ S_i^t, & \text{others} \end{cases} \tag{3-13}$$

其中，c_2 表示粒子与种群全局最优方案进行交叉操作的概率。粒子跟种群全局最优位置进行交叉的过程与该粒子跟其历史最优位置交叉的过程类似，这里不再赘述。

算法 2　Crossover operator(*p,q*)

输入：待交叉的粒子 *p,q*

输出：交叉后的新粒子

Initialize();//初始化每个粒子的 partition 为单元素集合
Sort_edge(p,u);//根据边的第一个端点 *u* 的编号给粒子 *p* 中的所有边排序
Sort_edge(p,v);//根据边的第二个端点 *v* 的编号给粒子 *p* 中的所有边排序
Sort_edge(q,u);
Sort_edge(q,v);
Set1=Select_same_edge(p,q);//选择粒子 *p* 和 *q* 的共同边存放在 *Set1*
Set2=Select_different_edge(p,q); //选择粒子 *p* 和 *q* 的不同边存放在 *Set2*
Union_partition(u,v,Set1);//合并 *Set1* 中所有边的端点
New particle=Generate_edge(Set1);将 *Set1* 的边作为新粒子的部分边
while *New particle is not a complete tree* **do**
  L(u,v)=random_select_edge(Set2); //随机挑选 *Set2* 中的边
  if *Find_set(u)*≠*Find_set(v)* **then**
    Add L(u,v) to New particle;
    Union_partition(u,v,L);
  end if
end while

3.3.5 本章算法与同类算法的测试情况

文献[47]第一次提出时延驱动X结构Steiner树的构造算法，大致思想：首先构造一棵矩形Steiner最小树，找出其中的拐点，如图3-7(a)所示；接着将这些拐点和连接的两条直角边去除，并用连接这两个点的斜边代替，如图3-7(b)所示，得到包含Steiner点的布线图；然后重新寻找合适的Steiner点位置，如图3-7(c)所示的*P3*，*P1*，*P2*均为重新寻找到Steiner点的新位置；最后将生成树的边转换成为X结构边，得到如图3-7(d)所示的最终时延驱动X结构Steiner最小树。

文献[47]所构造的时延驱动X结构Steiner最小树的线长为10×sqrt(2)+5=19.142，源点*Source*(*S*)到各漏点(*S1*，*S2*，*S3*，*S4*，*S5*，*S6*)的线长分别为*L* (*S*,*S1*)=2×*sqrt*(2)+1=3.828， *L*(*S*,*S2*)=3×*sqrt*(2)+2=6.243， *L*(*S*,*S3*)=4×*sqrt*(2)+1=6.657，*L*(*S*,*S4*)=3×*sqrt*(2)+1=5.243，*L*(*S*,*S5*)=5×*sqrt*(2)+1=8.071，*L*(*S*,*S6*)=4×*sqrt*(2)+2=7.657。其中，*sqrt*()表示求平方根函数，*L*(*S*,*S5*)是最大源漏点距离，而源点到所有漏点的距离之和为37.699。

本章基于多目标PSO的时延驱动X结构Steiner树的构造算法，所构建的Steiner树如图3-8所示，关于该测试实例的运行结果为：时延驱动X结构Steiner最小树的线长为6×*sqrt*(2)+10=18.485。源点*Source*(*S*)到各漏点(*S1*，*S2*，*S3*，*S4*，*S5*，*S6*)的线长分别为*L* (*S*,*S1*)=2×*sqrt*(2)+1=5，*L*(*S*,*S2*)=3×*sqrt*(2)+2=6.243，*L* (*S*,*S3*)=2×*sqrt*(2) +5=7.828，*L* (*S*,*S4*)=1×*sqrt*(2)+2=3.414，*L* (*S*,*S5*)=2×*sqrt*(2)+5=7.828，*L*(*S*,

S6) = 1× *sqrt*(2)+6=7.414。其中*L*(*S*,*S5*)为最大源漏点距离，而源点到所有漏点的距离之和为 37.727。从中可看出本章算法所构造TXST的树长和半径都比文献[47]的树长和半径分别减少3.43%和3.01%。而且文献[47]中拐弯数为8个，而本章构造的TXST的拐弯数为5个，在拐弯数的优化目标上也相对文献[47]减少3个即减少37.5%。

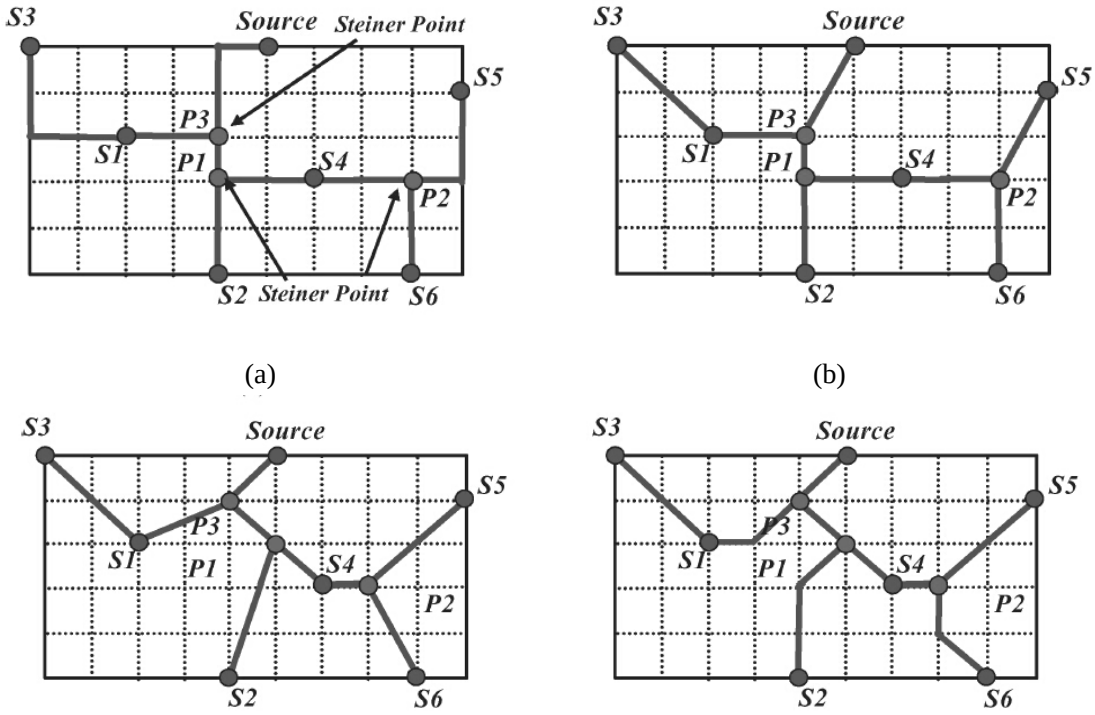


图 3-7 基于 Steiner 点重新调度和路径重构的时延驱动 X 结构 Steiner 树的构建过程[47]

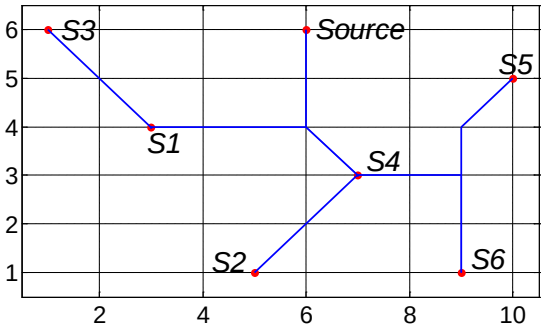


图 3-8 本章算法针对文献[47]的实例所构建的时延驱动 X 结构 Steiner 树

3.3.6 TXST_BR_MOPSO 算法的流程

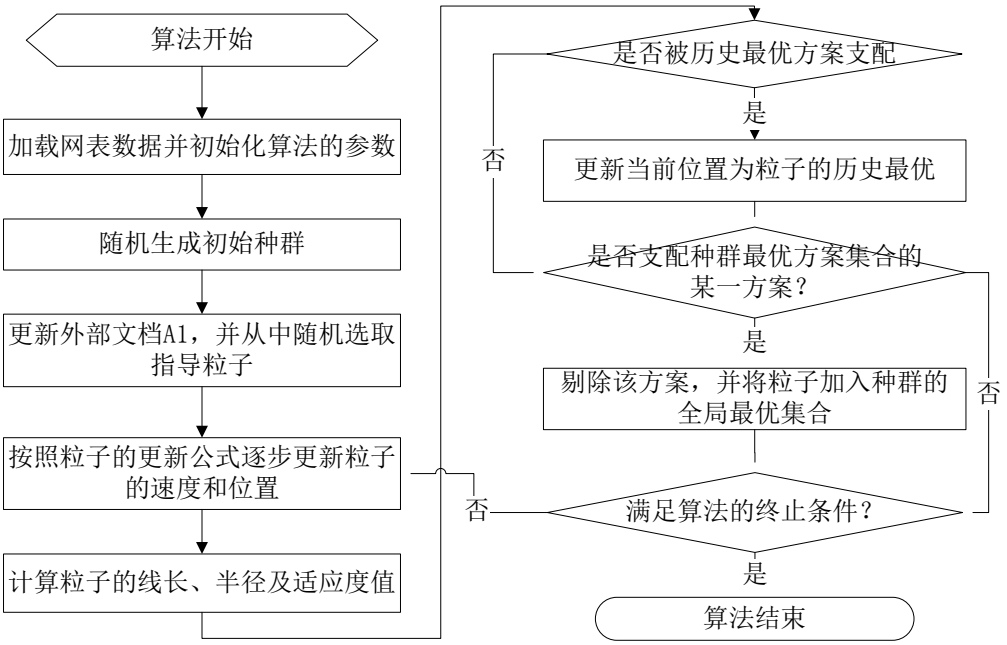


图 3-9 TXST_BR_MOPSO 算法的流程图

TXST_BR_MOPSO 算法的流程图如图 3-9 所示，算法的详细步骤可归纳如下：

步骤 1：加载电路网表数据，并初始化算法中的参数，同时随机产生初始种群。

步骤 2：种群中的每个粒子根据公式(3-9)计算粒子的适应度值。

步骤 3：更新每个粒子的个体最优方案，并存储在外部文档 A1 中，并从中随机选取一个方案作为指导粒子。

步骤 4：根据公式(3-10)至(3-13)更新每个粒子的速度和位置。

步骤 5：重新计算每个粒子的线长值、半径值以及适应度值。

步骤 6：更新种群的全局最优方案集合，并存储至外部文档 A2 中。

步骤 7：检查算法是否满足终止条件(收敛至一个足够好的位置或已达到算法设置的最大迭代次数)。如果满足，则算法终止并输出全局最优方案集合 A2，即算法的最终解方案集合。否则，转到步骤 3。

3.3.7 TXST_BR_MOPSO 算法的复杂度分析

引理 3.1 假设种群大小为 p，迭代次数为 $iters$，引脚的个数为 n，则 TXST_BR_MOPSO 算法的复杂度为 $O(iters \times p \times n^2)$。

证明：在 TXST_BR_MOPSO 算法的内部循环中，即从步骤 3 至步骤 6，包含变异操作、交叉操作以及适应度值计算操作。在变异和交叉操作中，由于并查集策略的时间复杂度略多于线性时间[108]，所以复杂度为 $O(n \log n)$ 的排序方法主导

这些操作的算法复杂度。另外，在计算适应度的操作中，计算布线树半径的复杂度为 $O(n^2)$，而计算布线树线长的复杂度亦由排序方法的复杂度决定。因此，算法内部循环的复杂度为 $O(n \log n + n^2 + n \log n) = O(n^2)$。同时算法外部循环的复杂度主要跟种群的大小以及算法的迭代次数相关，因此 TXST_BR_MOPSO 算法的复杂度为 $O(iters \times p \times n^2)$。

3.3.8 TXST_BR_MOPSO 算法的收敛分析

定义 3.9 (有限马尔科夫链)[109] 令 $X(X = X_k, k = 1,2,\cdots)$ 表示定义在概率空间 (Ω, F, P) 中的离散参数关于有限状态空间 S 的随机过程。如果 X 有马尔科夫性质，即对任意非负整数 k 和状态 $i_0, i_1, \cdots, i_{k-1} \in S$，若 X 是有限马尔科夫链当

$$P(X_{k+1} = i_{k+1} \mid X_0 = i_0, X_1 = i_1, \ldots, X_k = i_k) = P(X_{k+1} = i_{k+1} \mid X_k = i_k) \tag{3-14}$$

$$P(X_0 = i_0, X_1 = i_1, \cdots, X_k = i_k) > 0$$

其中，$P(X_{k+m} = j \mid X_k = i)$ 称为步骤 m 的转换概率 X，它是从状态 i 在时刻 k(步骤 m 之后)到状态 j 在时刻 $k+m$(这里表示为 $P_{ij}(k, k+m)$)。对于 $i, j \in S$，如果 $P_{ij}(k, k+1)$ (简称为 P_{ij})不依赖于时刻 k，则马尔科夫链是齐次的。$P = [P_{ij}]$ 称为转换矩阵，其中 P_{ij} 表示第 i 行第 j 列。齐次有限马尔科夫链的这种长期行为完全取决于其初始分布和第一步的转换概率。

定理 3.1 TXST_BR_MOPSO 算法的马尔科夫链是有限且齐次的。

证明：在算法的全局搜索过程中，TXST_BR_MOPSO 算法是通过随机变异操作和交叉操作获得全局最优方案和粒子的历史最优方案。从全局搜索的过程中判断出，新产生的种群是取决于当前种群的状态，因此从一个状态到另一个具体状态的算法搜索过程的条件概率是满足公式(3-14)，也就是算法满足马尔科夫性质。所以，TXST_BR_MOPSO 算法的马尔科夫链是有限且齐次的。在该算法中，所有种群 $\{\alpha_1, \alpha_2, \cdots, \alpha_m\}$ 组成的集合是有限的。也就是发生在时刻 $k = 0,1,\cdots$ 的事件都属于一个有限可数的事件集合，因此，其马尔科夫链也是有限的[110]。综上，在以上关于马尔科夫链的理论分析和方法均可直接应用到本算法的分析中。

定理 3.2 TXST_BR_MOPSO 算法所构成的马尔科夫链的转移概率矩阵是正定的。

证明：在算法搜索过程中，种群通过一个粒子自身的变异操作和两个交叉操作(粒子与历史最优方案及全局最优方案的交叉操作)从状态 $i_i \in S$ 转移到状态 $i_j \in S$。这三个操作的转移概率分别为 m_{ij}，g_{ij}，d_{ij}，它们相应的转移矩阵分别为 $M = \{m_{ij}\}$，$G = \{g_{ij}\}$，$D = \{d_{ij}\}$。令 $P = MGD$，则

$$m_{ij} > 0, \sum_{i_j \in E} m_{ij} = 1; g_{ij} \geq 0, \sum_{i_j \in E} g_{ij} = 1; d_{ij} \geq 0, \sum_{i_j \in E} d_{ij} = 1$$

因此，M,G,D 都是随机，且 M 是正定的，然后可证明 P 是正定的。令 $B=GD$，对于 $\forall\, i_i \in S, i_j \in S$，存在

$$b_{ij}=\sum_{\lambda_k \in E} g_{ik}d_{kj} \geq 0$$

$$\sum_{\lambda_j \in E} b_{ij}=\sum_{\lambda_j \in E}\sum_{\lambda_k \in E} g_{ik}d_{kj}=\sum_{\lambda_k \in E} g_{ik}\sum_{\lambda_j \in E} d_{kj}=\sum_{\lambda_k \in E} g_{ik}=1$$

因此，B 是随机矩阵，同样可知道 $d_{ij}=\sum_{\lambda_k \in E} b_{ik}m_{kj}>0$。

定理 3.3 (马尔科夫链的极限理论) 假定 P 是齐次马尔科夫链对应的正定随机转移矩阵，则

(1) 存在唯一的概率向量 $\overline{P}^T>0$，其满足 $\overline{P}^T P=\overline{P}^T$。

(2) 对于任意的一个初始状态 $i(e_i^T$ 作为其相应的初始概率)，可得 $\lim_{k\to\infty} e_i^T P^k=\overline{P}^T$。

(3) 概率矩阵的极限为 $\lim_{k\to\infty} P^k=\overline{P}$，其中 $\overline{P}$ 是一个 $n\times n$ 的随机矩阵且其所有行等同于 $\overline{P}^T$。

该极限理论解释了马尔科夫链的长期行为不依赖于其初始状态，并且该理论是算法收敛的基础。

引理 3.2 如果变异概率 $m \succ 0$，则 TXST_BR_MOPSO 算法是一个遍历且不可约的马尔科夫链，其中算法只有一个有限分布且与初始分布无关。另外，在任意随机时刻和任意随机状态中，变异概率是大于 0 的。

证明： 在时刻 t，状态 j，种群的概率分布 $X(t)$ 是

$$P_j(t)=\sum_{j\in S} P_i(1)P_{ij}^{(t)}, t=1,2,\cdots \tag{3-15}$$

根据定理 2，可得如下公式。

$$P_j(\infty)=\lim_{t\to\infty}(\sum_{i\in S} P_i(1)P_{ij}^{(t)})=\sum_{i\in S} P_i(1)P_{ij}^{(\infty)}>0, \forall j\in S \tag{3-16}$$

定义 3.10 假设一个随机变量 $Z_t=\max\{f(x_k^{(t)}(i))\,|\,k=1,2,\cdots,N\}$ 表示种群在状态 i 步骤 t 的粒子历史最优方案，则算法收敛于全局最优方案，当且仅当

$$\lim_{t\to\infty} P\{Z_t=Z^*\}=1 \tag{3-17}$$

其中，$Z^*=\max\{f(x)\,|\,x\in S\}$ 表示全局最优方案。

定理 3.4 对于任意 i 和 j，一个可遍历的马尔科夫链从状态 i 至状态 j，其时间转移矩阵是有限的。

定理 3.5 TXST_BR_MOPSO 算法可收敛于全局最优方案。

证明： 假设对于 $i\in S$，$Z_t<Z^*$，且 $P_i(t)$ 是 TXST_BR_MOPSO 算法在状态 i 步骤 t 的概率矩阵，显然 $P\{Z_t\neq Z^*\}\geq P_i(t)$，因此可知 $P\{Z_t=Z^*\}\leq 1-P_i(t)$。

根据引理 3.2，在 TXST_BR_MOPSO 算法中状态 i 的操作算子的概率是 $P_i(\infty)>0$，则

$$\lim_{t\to\infty} P\{Z_t = Z^*\} \le 1 - P_i(\infty) < 1 \tag{3-18}$$

给定一个新种群，例如 $X_t^+ = \{Z_t, X_t\}, t \ge 1$，其中，$x_{ti} \in S$ 表示搜索空间(是有限集合或可数集合)，跟定义 3.10 一样的 Z_t 表示当前种群的个体最优值，X_t 表示在搜索过程中的种群，可证明群体转移过程 $\{X_t^+, t \ge 1\}$ 仍是一个齐次的且可遍历的马尔科夫链，从而得到

$$P_j^+(t) = \sum_{i\in S} P_i^+(1) P_{ij}^+(t) \tag{3-19}$$

$$P_{ij}^+ > 0 \quad (\forall i \in S, \forall j \in S_0)$$

$$P_{ij}^+ = 0 \quad (\forall i \in S, \forall j \notin S_0)$$

因此

$$(P_{ij}^+)^t \to 0 \quad (t \to \infty)$$

$$P_j^+(\infty) \to 0 \quad (j \notin S_0)$$

$$\lim_{t\to\infty} P\{Z_t = Z^*\} = 1 \tag{3-20}$$

3.4 实验结果

本章所有算法均在 MATLAB R2009a 的 PC 机上运行，环境为微软 XP，CPU 2.00 GHz，RAM 2.00 GB。在文献[72]中关于 Oliver30 TSP 问题已经测试了 567 种加速因子的不同组合。每组实验中算法分别重复运行五次并计算平均值作为相应的实验结果。由最终的实验结果表明 c_1=0.82-0.5 和 c_2=0.4-0.83 这一加速因子组合是最佳的参数组合。本章算法采用文献[85]中的线性递减策略和文献[72]中的最佳参数组合进行加速因子 c_1 和 c_2 的参数更新。此外，算法的其他参数设置如下：种群大小为 200，w 的更新策略跟两个加速因子的更新策略相似，从 0.95 随着迭代次数的增加线性递减至 0.4。本章采用 OR-Library 中的测试数据集[87]进行实验仿真用以验证本章算法的性能，主要包括在 3.4.1 小节至 3.4.4 小节中分别进行的四组相关实验。

3.4.1 边变换策略的有效性验证

为了验证本章算法中边变换策略的有效性，将本章算法与未采取边变换策略的 XSMT 构造算法在线长指标上进行对比。如表 3-2 所示，第一列表示测试实例的编号情况，第二列和第三列分别表示未引入边变换策略(称为 XSMT)和引入边变换策略(称为 EXSMT)的两种算法所构造 X 结构 Steiner 最小树的线长，最后一

列表示 EXSMT 在线长指标上相对 XSMT 的线长减少率。从表 3-2 可看出，引入边变换策略的算法增强了算法在线长指标上的优化能力，相对未引入边变换策略的 XSMT 构造算法[100]取得 1.57%的平均线长减少率。

表 3-2 EXSMT 和 XSMT 的线长对比

实例	XMST 线长	EXSMT 线长	减少率 $\frac{XSMT - EXSMT}{XSMT}$
1	33.0711	33.0711	0.00%
2	27.4853	26.5563	3.38%
3	35.4853	35.3848	0.28%
4	27.0711	26.8995	0.63%
5	26.2426	26.2426	0.00%
6	34.7279	34.7279	0.00%
7	36.4853	35.3137	3.21%
8	34.8995	34.3137	1.68%
9	31.8284	31.2426	1.84%
10	33.1421	32.8995	0.73%
11	28.7279	28.3137	1.44%
12	27.0711	26.8995	0.63%
13	29.0711	27.6569	4.86%
14	32.0711	30.7279	4.19%
15	27.2426	27.0711	0.63%
AVG	30.9748	30.4881	1.57%

3.4.2 本章 MOPSO 算法的性能分析

为了研究本章 MOPSO 算法的性能，将本章的 MOPSO 算法所产生的最优方案集合与 NSGA II [111]和 SPEA 2 [112]这两种典型的多目标进化算法在基准函数 ZDT1，ZDT2 和 ZDT6 上进行对比。这三个基准函数是由 Zitzler 提出来的[113]，代表一些典型的函数类型，在多目标优化问题中拥有不同的非劣前端。其中 ZDT1 是凸函数，ZDT2 是非凸函数，ZDT6 是非连续函数。本章实验采用 Zitzler 等提出的三种量化指标[114]，包括一元附加 EPS 指标 $I^1_{\varepsilon+}$，HYP 指标 I^-_H 和一元 R2 指标 I^1_{R2}，用以验证本章 MOPSO 算法的有效性。

对于三个基准函数 ZDT1，ZDT2 和 ZDT6，在多数情况下，本章 MOPSO 算法可在迭代次数为 100 以内以较少的计算成本收敛于帕累托前端。本章采用 Kruskal-Wallis 统计测试方法[115]对这些算法进行对比。这三种量化指标的 Kruskal-Wallis 统计测试结果如表 3-3 所示。表 3-3 列出了每一组由 Q_R 行的算法和 Q_C 列的算法所构成的 P 值，且相应交替假设算法 Q_R 在各个指标上是优于算

法 Q_C 的，其中显著性水平值为 0.05。方便起见，EPS 表示 $I^1_{\varepsilon+}$，HYP 表示 I^-_H，R2 表示 I^1_{R2}。

表 3-3　EPS, HYP 和 R2 指标的 Kruskal Wallis 统计测试结果

		$I^1_{\varepsilon+}$		
		MOPSO	NSGA II	SPEA 2
ZDT 1	MOPSO	-	1	1
	NSGA II	4.16519E-17	-	1
	SPEA 2	3.48666E-34	4.16519E-17	-
		I^-_H		
	MOPSO	-	1	1
	NSGA II	2.53616E-16	-	1
	SPEA 2	4.00793E-32	4.88893E-15	-
		I^1_{R2}		
	MOPSO	-	1	0.99996
	NSGA II	7.6029E-08	-	0.0551455
	SPEA 2	3.99655E-05	0.944854	-
ZDT 2		$I^1_{\varepsilon+}$		
	MOPSO	-	2.20462E-06	2.16472E-05
	NSGA II	0.999998	-	0.72699
	SPEA 2	0.999978	0.27301	-
		I^-_H		
	MOPSO	-	0.000234484	0.00112884
	NSGA II	0.999766	-	0.689058
	SPEA 2	0.998871	0.310942	-
		I^1_{R2}		
	MOPSO	-	1.18542E-14	7.16123E-14
	NSGA II	1	-	0.658011
	SPEA 2	1	0.341989	-
ZDT 6		$I^1_{\varepsilon+}$		
	MOPSO	-	1	1
	NSGA II	3.27923E-07	-	1
	SPEA 2	1.28526E-20	3.94864E-11	-
		I^-_H		
	MOPSO	-	3.10337E-14	1.21161E-20
	NSGA II	1	-	0.000456755
	SPEA 2	1	0.999543	-
		I^1_{R2}		
	MOPSO	-	8.72107E-20	4.22686E-29
	NSGA II	1	-	1.43107E-07
	SPEA 2	1	1	-

从表 3-3 可知，对于基准测试函数 ZDT2，本章 MOPSO 算法在三个量化指标上是优于其他两种多目标优化算法，从而说明本章 MOPSO 算法特别适用于求

解非凸函数的帕累托前端。而对于基准测试函数 ZDT6，本章 MOPSO 算法在量化指标 R2 和 HYP 取得相对较优的结果，而在 EPS 指标上微弱于其他两种算法。基于文献[99]可知如果多个量化指标的实验结果不一致，则很难判断哪个算法更为优秀。因此，本章 MOPSO 算法可在 ZDT6 这一类非连续函数上取得不错的结果。针对基准测试函数 ZDT1，本章 MOPSO 算法尽管在三种量化指标均劣于 NSGA II 和 SPEA 2，但总体差异不大。在解方案的优劣排序分布中，本章 MOPSO 算法相对其他两种多目标优化算法并未有明显的差异，而且本章 MOPSO 算法能以较少的迭代次数收敛于帕累托前端，因此可说明本章 MOPSO 能够在 ZDT1 这一类具有凸帕累托前端的函数中取得满意的解方案。

通过上述关于三种基准测试函数的综合实验结果分析，可知本章 MOPSO 算法拥有较强的全局搜索能力，且能以较少的计算代价收敛于帕累托前端，并取得较好分布的解方案。因此，本章 MOPSO 算法值得在多目标优化问题领域中进行相关的应用研究。

表 3-4 与两种 SMT 算法(RA;OA)在线长的对比

实例	线长				
	RA (*A1*)	*OA* (*B1*)	*Ours* (*C1*)	减少率 (%) $\frac{A1-C1}{A1}$	减少率(%) $\frac{B1-C1}{B1}$
S0	21.000	19.0711	18.7279	10.82	1.80
1	35.000	33.0711	33.8995	3.14	-2.50
2	31.000	26.7279	30.5563	1.43	-14.32
3	38.000	36.0711	35.9706	5.34	0.28
4	29. 000	27.0711	28.3137	2.37	-4.59
5	28.000	26.2426	26.8995	3.93	-2.50
6	40. 000	34.8995	35.9706	10.07	-3.07
7	40. 000	36.4853	35.3137	11.72	3.21
8	38. 000	34.8995	36.5563	3.80	-4.75
9	33. 000	31.2426	32.1421	2.60	-2.88
10	38. 000	33.1421	34.2132	9.97	-3.23
11	32. 000	29.3137	29.1421	8.93	0.59
12	29. 000	26.8995	28.8995	0.35	-7.44
13	32. 000	29.0711	27.7279	13.35	4.62
14	35. 000	31.6569	31.1421	11.02	1.63
15	29. 000	27.0711	27.0711	6.65	0.00
AVG	33.000	30.1835	30.7841	6.59	-2.07

3.4.3 与两类 SMT 构造算法的对比

为了验证本章算法在布线树的线长和半径两个目标的优化能力，将本章算法(Ours)与两类 SMT 构造算法[76,100]进行比较，这两类算法分别命名为 RA 和 OA。RA 是基于曼哈顿结构，而 OA 是基于 X 结构，它们都只考虑线长这单一个目标的优化。本章算法采用多目标 PSO 优化方式可获得多组解方案，这里选取拐弯数最少的一组解方案作为最终方案在后续实验中进行对比。表 3-4 至表 3-7 给出了本章算法与两类 SMT 构造算法的实验结果比较。其中表 3-4 至表 3-7 中所列出的第一个实例‘S0’，是来自于文献[47]的一个例子，而表 3-4 中的第 5 列和表 3-5 中的第 5 列说明本章算法构造的布线树相对 RA 算法在线长指标取得 6.59%的平均减少率，同时在半径指标取得 24.07%的平均减少率。表 3-4 中的第 6 列和表 3-5 中的第 6 列则说明了本章算法构造的布线树相对 OA 算法在半径指标上取得了 19.30%的平均减少率，而只多付出了平均 2.07%的线长代价。

表 3-5 与两种 SMT 算法(RA;OA)在半径的对比

实例	半径				
	RA (*A2*)	*OA* (*B2*)	*Ours* (*C2*)	减少率 (%) $\frac{A2-C2}{A2}$	减少率(%) $\frac{B2-C2}{B2}$
S0	11.0000	9.4142	6.6569	39.48	29.29
1	19.0000	17.8284	16.6569	12.33	6.57
2	27.0000	22.8995	14.2426	47.25	37.80
3	22.0000	25.0711	17.8995	18.64	28.61
4	15.0000	15.0000	13.2426	11.72	11.72
5	17.0000	16.4142	15.2426	10.34	7.14
6	24.0000	21.6569	19.3137	19.53	10.82
7	27.0000	24.0711	19.0711	29.37	20.77
8	30.0000	28.4853	21.6569	27.81	23.97
9	22.0000	19.4142	17.3137	21.30	10.82
10	30.0000	25.3137	15.4853	48.38	38.83
11	19.0000	19.4853	15.2426	19.78	21.77
12	18.0000	17.6569	16.8284	6.51	4.69
13	24.0000	22.2426	19.8995	17.09	10.53
14	25.0000	23.2426	18.6569	25.37	19.73
15	19.0000	17.8284	13.2426	30.30	25.72
AVG	21.8125	20.3765	16.2907	24.07	19.30

从表3-6和表3-7看出本章算法相对RA算法在最大源漏点时延(max s-t delay)指标取得 32.97%的平均减少率和在总的源漏点时延(sum s-t delay)指标取得 36.13%的平均减少率，同时相对 OA 算法在最大源漏点时延指标取得 24.39%的

平均减少率和在总的源漏点时延指标取得 23.95%的平均减少率。以上实验结果表明基于 X 结构布线相对于曼哈顿结构布线而言拥有更强的线长和时延优化能力，同时相对于同是 X 结构但只优化线长的 OA 算法，本章算法能够在以付出少量的线长代价换取时延指标的较大优化效果。

表 3-6 与两种 SMT 算法(RA;OA)在最大源漏点时延的对比

实例	max s-t delay (10^4ps)				
	RA (*A3*)	*OA* (*B3*)	*Ours* (*C3*)	减少率(%) $\frac{A3-C3}{A3}$	减少率(%) $\frac{B3-C3}{B3}$
S0	3.2104	1.6763	1.5660	51.22	6.58
1	8.4546	7.1853	7.1764	15.12	0.12
2	14.4690	10.7030	5.8080	59.86	45.73
3	14.3870	16.4570	8.6930	39.58	47.18
4	5.3217	5.3149	4.5980	13.60	13.49
5	8.4367	7.9756	6.9760	17.31	12.53
6	12.0040	11.0600	9.9370	17.22	10.15
7	26.2360	21.0310	14.0750	46.35	33.07
8	24.8310	22.9950	14.2810	42.49	37.90
9	12.3400	11.7040	8.4860	31.23	27.49
10	18.5070	14.6500	6.6450	64.09	54.64
11	8.6578	9.3386	6.7213	22.37	28.03
12	8.8019	7.9013	7.7136	12.36	2.38
13	15.8670	14.1490	11.6170	26.79	17.90
14	14.0990	12.6580	10.2200	27.51	19.26
15	10.8980	9.7978	6.4903	40.45	33.76
AVG	12.9076	11.5373	8.1877	32.97	24.39

3.4.4 与时延驱动 X 结构 Steiner 树构造算法的对比

表 3-8 和表 3-9 给出了 TXST_BR_MOPSO 算法与时延驱动 X 结构 Steiner 树构造算法[47](TOA)在线长、半径、最大源漏点时延、总的源漏点时延、拐弯数等五个指标上的实验对比情况。为了有效地与 TOA 算法进行对比，OR-Library 中的所有测试数据都乘以 15 的基础上取整(3.4.3 小节的实验数据同理进行处理)。从表 3-8 可看出在线长和半径两个指标上，TXST_BR_MOPSO 算法相对于 TOA 分别取得 2.08%和 5.22%的平均减少率。而且从表 3-9 可看出在最大源漏点时延、总的源漏点时延和拐弯数三个指标上，TXST_BR_MOPSO 算法相对于 TOA 算法分别取得 4.93%，10.07%和 19.67%的平均优化率。由于 TXST_BR_MOPSO 算法从更为全局的角度进行搜索，以同时优化线长和半径两个对时延有重大影响的指

标，所以 TXST_BR_MOPSO 算法可获得比基于贪心策略且容易陷入局部最优解的 TOA 算法更优秀的时延值。另外，TXST_BR_MOPSO 算法在拐弯数指标取得较大减少的情况，则说明本章编码策略与边变换策略能够很好结合从而有效地减少拐弯数。

表 3-7 与两种 SMT 算法(RA;OA)在总的源漏点时延的对比

实例	sum s-t delay (10^5ps)				
	RA (*A4*)	*OA* (*B4*)	*Ours* (*C4*)	减少率(%) $\frac{A4-C4}{A4}$	减少率(%) $\frac{B4-C4}{B4}$
S0	2.3524	1.1357	0.8036	65.84	29.24
1	5.4829	4.5786	5.1646	5.81	-12.80
2	14.2160	9.2560	5.0689	64.34	45.24
3	13.3370	13.8880	7.2170	45.89	48.03
4	3.7561	3.3623	2.9936	20.30	10.97
5	6.4900	6.2180	5.5906	13.86	10.09
6	9.3503	8.9313	7.4255	20.59	16.86
7	22.8060	18.4760	10.4730	54.08	43.32
8	15.7080	13.7140	10.0090	36.28	27.02
9	9.9499	9.7153	7.3543	26.09	24.30
10	13.0862	10.1170	4.6610	64.38	53.93
11	8.8057	7.7425	5.5512	36.96	28.30
12	7.9341	5.8701	5.8414	26.38	0.49
13	13.2660	9.6916	8.5143	35.82	12.15
14	14.6920	12.9530	9.8520	32.94	23.94
15	8.1944	7.5223	5.8524	28.58	22.20
AVG	10.5892	8.9482	6.3983	36.13	23.95

表 3-10 的前四列给出五种 IC 工艺的具体参数值，而后五列表示在五种不同 IC 工艺下 TXST_BR_MOPSO 算法与 TOA 算法在线长(wire length)、半径(radius)、最大源漏点时延、总源漏点时延及拐弯数(number of bends)这五个指标上的平均减少率。因为一个数据集合的标准差可反映该数据集合的波动性，从图 3-10 可发现每个指标在五种 IC 工艺下所取得的平均减少率的标准差均非常小且小于 0.014，所以从中可表明本章算法在不同的 IC 工艺下是稳定且有效的。

如图 3-11 给出本章算法在编号为 1 和 2 两个测试实例上的收敛曲线图。从图 3-11 中可看出本章算法在两个测试实例上均能在较少的迭代次数内（总迭代次数为 200）收敛到一个较为不错的解方案。由于其他测试实例的收敛曲线图也跟这两个测试实例的收敛曲线图类似，这里就不一一给出。

表 3-8 与同类 TXST 算法(TOA)在线长、半径的对比

实例	线长			半径		
	TOA (*D1*)	*Ours* (*C1*)	减少率 (%) $\frac{D1-C1}{D1}$	*TOA* (*D2*)	*Ours* (*C2*)	减少率(%) $\frac{D2-C2}{D2}$
S0	19.1421	18.7279	2.16	8.0711	6.6569	17.52
1	33.4853	33.8995	-1.24	17.8284	16.6569	6.57
2	31.1421	30.5563	1.88	16.2426	14.2426	12.31
3	37.7279	35.9706	4.66	18.4853	17.8995	3.17
4	29.6569	28.3137	4.53	15.0000	13.2426	11.72
5	26.4853	26.8995	-1.56	15.8284	15.2426	3.70
6	36.9706	35.9706	2.70	19.8995	19.3137	2.94
7	36.4853	35.3137	3.21	19.0711	19.0711	0.00
8	37.9706	36.5563	3.72	21.6569	21.6569	0.00
9	34.7279	32.1421	7.45	17.6569	17.3137	1.94
10	34.6274	34.2132	1.20	17.4853	15.4853	11.44
11	29.5563	29.1421	1.40	15.0711	15.2426	-1.14
12	27.0711	28.8995	-6.75	18.2426	16.8284	7.75
13	29.3137	27.7279	5.41	21.0711	19.8995	5.56
14	31.8995	31.1421	2.37	18.6569	18.6569	0.00
15	27.6569	27.0711	2.12	13.2426	13.2426	0.00
AVG	31.4949	30.7841	2.08	17.0944	16.2907	5.22

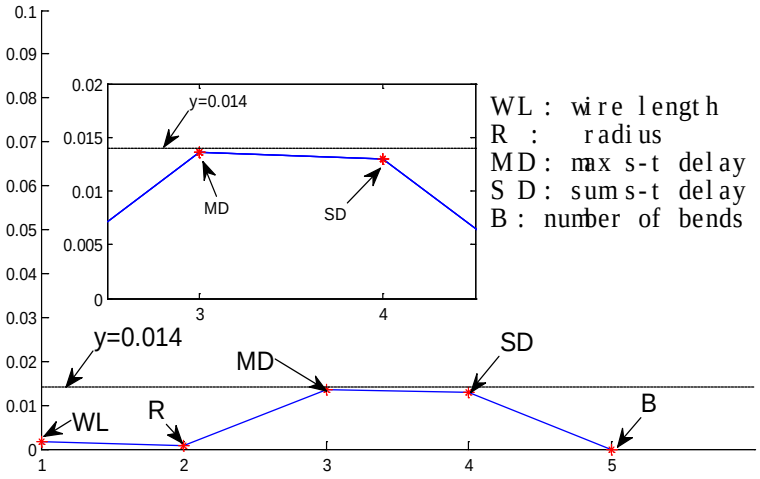


图 3-10 表 3-10 所示的五种性能指标改进率集合的标准差

表 3-9 与 TXST 算法(TOA)在最大源漏点时延与总的源漏点时延的对比

实例	max s-t delay (10^4ps)			sum s-t delay (10^5ps)			拐弯数		
	TOA (*D3*)	*Ours* (*C3*)	减少率 (%) $\frac{D3-C3}{D3}$	*TOA* (*D4*)	*Ours* (*C4*)	减少率 (%) $\frac{D4-C4}{D4}$	*TOA* (*D5*)	*Ours* (*C5*)	减少率 (%) $\frac{D5-C5}{D5}$
S0	1.8003	1.5660	13.01	1.3516	0.8036	40.54	8.0000	5.0000	37.50
1	7.5840	7.1764	5.37	5.2654	5.1646	1.91	10.0000	7.0000	30.00
2	8.7020	5.8080	33.26	6.6822	5.0689	24.14	12.0000	10.0000	16.67
3	8.3574	8.6930	-4.02	8.2290	7.2170	12.30	10.0000	9.0000	10.00
4	5.1806	4.5980	11.25	3.1631	2.9936	5.36	9.0000	8.0000	11.11
5	7.2874	6.9760	4.27	5.5906	5.5906	0.00	10.0000	8.0000	20.00
6	8.3432	9.9370	-19.10	6.6707	7.4255	-11.32	14.0000	10.0000	28.57
7	14.5640	14.0750	3.36	13.6470	10.4730	23.26	12.0000	10.0000	16.67
8	14.2820	14.2810	0.01	10.8670	10.0090	7.90	10.0000	8.0000	20.00
9	10.1480	8.4860	16.38	9.4058	7.3543	21.81	8.0000	8.0000	0.00
10	7.7577	6.6450	14.34	5.4999	4.6610	15.25	14.0000	11.0000	21.43
11	6.2610	6.7213	-7.35	5.0439	5.5512	-10.06	12.0000	10.0000	16.67
12	8.1007	7.7136	4.78	6.4449	5.8414	9.36	10.0000	8.0000	20.00
13	13.3840	11.6170	13.20	9.4219	8.5143	9.63	12.0000	9.0000	25.00
14	9.2866	10.2200	-10.05	10.1720	9.8520	3.15	10.0000	7.0000	30.00
15	6.5065	6.4903	0.25	6.3513	5.8524	7.86	9.0000	8.0000	11.11
AVG	8.5966	8.1877	4.93	7.1129	6.3983	10.07	10.6250	8.5000	19.67

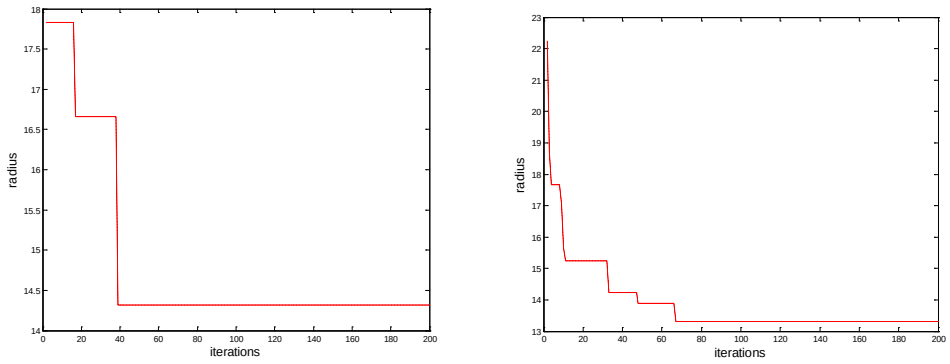
表 3-10 与 TXST 算法(TOA)在不同 IC 工艺下的对比

r_d	r_0	c_0	c_k	IC 工艺	线长	半径	max s-t delay	sum s-t delay	拐弯数
180.0	0.0075	0.118	23.4	tost	2.08%	5.22%	4.93%	10.07%	19.67%
164.0	0.0330	0.234	5.7	IC1	1.95%	5.23%	5.09%	12.45%	19.67%
212.1	0.0730	0.083	7.1	IC2	2.16%	5.10%	6.31%	13.24%	19.67%
270.0	0.1120	0.039	1.0	IC3	2.16%	5.10%	3.70%	11.87%	19.67%
25.0	0.0080	0.060	1000.0	MCM	2.42%	5.22%	7.24%	13.20%	19.67%
				STD	0.0017	0.0007	0.0136	0.0130	0.0000

3.5 小结

构建时延驱动 Steiner 树是 VLSI 性能驱动总体布线阶段中一个非常重要的问题。针对此问题，本章基于 MOPSO 和 Elmore 时延模型提出了一种构建时延驱动 X 结构 Steiner 树的有效算法，从而有助于 X 结构性能驱动总体布线的研究。本章算法设计了边变换操作和并查集策略以使算法的进化过程更为有效。另外，本章算法考虑到拐弯数的优化工作，而拐弯数是影响芯片可制造性的一个关键指

标且对 VLSI 布线有重要意义。本章算法是第一次在非曼哈顿结构性能布线工作中考虑到拐弯数的减少。相关实验结果表明 TXST_BR_MOPSO 算法相对其他算法取得可观的性能优化效果，并且在不同 IC 工艺下算法是稳定且有效的。



(a) 编号为 1 的实例　　(b) 编号为 2 的实例

图 3-11 两个测试实例的算法收敛情况

第四章 多层绕障 X 结构 Steiner 最小树构建算法

4.1 引言

超大规模集成电路设计中多层绕障 X 结构 Steiner 最小树问题是给定布线层上一系列布线引脚和障碍物集合，通过 X 结构边连接每个布线层上的引脚且布线层之间借助通孔连接，在布线边和通孔不穿越障碍物的约束下，构建布线总代价最小的 Steiner 树。

ML-OAXSMT 问题是考虑到障碍物、X 结构、多层等三个条件的 Steiner 最小树模型。首先，作为 ML-OAXSMT 问题的基础模型，Steiner 最小树是布线中多端线网连接的最佳模型。近年来超大规模集成电路设计中芯片会存在宏单元、IP 预布好的线网等布线障碍物，在此基础上考虑到障碍物的 Steiner 最小树(Obstacle-Avoiding Steiner Minimal Tree，OASMT)问题受到广泛的关注。其次，目前关于布线树的相关研究工作主要集中在曼哈顿结构，但基于曼哈顿结构进行线长与时延的优化，由于其布线走向有限，不能够充分地利用布线区域，其优化能力受限。一些研究人员开展了能为布线带来线长、通孔、功耗等优化指标提高的非曼哈顿结构的相关研究工作，但非曼哈顿结构下绕障工作研究相对较少[116-118]。最后，随着集成电路设计工艺的不断发展，允许绕线的布线层数随之不断增加，大幅度减少了互连线宽度和互连线间距，从而提高了集成电路的性能和密度。因此多层布线应运而生，并且引起了诸多研究机构的广泛关注[3]。

针对 ML-OAXSMT 问题，本章提出了一种 X 结构下带粒子群优化的多层绕障 Steiner 最小树算法，称之为 ML-OAXSMT-PSO。为了提高布线树的绕障能力，算法在 Steiner 点变换操作的基础上加入边变换操作，并设计了基于并查集思想的操作算子以避免生成树中环路的产生，同时设计了适合 X 结构和多层结构的多层边点对编码策略；提出了惩罚函数策略，以达到不违反绕障约束和不鼓励通孔产生，从而优化布线总代价；为了减少冗余的绕障判断次数和通孔计算次数，设计了预处理策略。实验结果表明了本章算法和相关策略的可行性和有效性，并在多层布线中布线总代价这一最重要的优化目标带来显著减少的效果。另外，由于考虑到定向边约束的绕障定向 X 结构 Steiner 最小树问题更符合 X 结构布线的实际工艺，所以基于 ML-OAXSMT-PSO 算法，本章设计一个构造 OAPD-XST 的算法，进而分析 OAPD-XST 问题中不同定向边约束模型的优劣性。

4.2 相关研究工作

Steiner 最小树问题是布线中线网连接的重要模型，而 Steiner 最小树及其变形，可用于布局、总体布线及详细布线等 VLSI 物理设计阶段进行线长、时延、拥挤度等方面的估计。因此，在这些方面有大量研究工作，本节分别介绍与本章相关的三类工作。

4.2.1 X 结构 Steiner 最小树

洪先龙等指出非曼哈顿结构下的物理设计相关研究将成为国际上布线领域的热点研究方向[20]。文献[39]提出了在 X 结构下的布线树和布线算法的一些挑战和机遇，同时指出在 X 结构下，Steiner 最小树问题仍是最为关键的问题之一。X 结构 Steiner 最小树的相关工作在第一章中已经展开论述。此外，一些以时延为优化目标的非曼哈顿结构 Steiner 最小树的工作也相应开展，包括构建 X 结构时延驱动 Steiner 最小树[47,119]和 Y 结构时延驱动 Steiner 最小树[48,49]。

4.2.2 单层绕障 Steiner 最小树

以上相关研究虽然都是在非曼哈顿结构下开展的 Steiner 最小树构建工作，并取得一系列的研究成果，但均忽略布线过程中经常存在障碍物的情况。本小节将介绍考虑障碍物的单层 Steiner 最小树的相关工作，包括曼哈顿结构与非曼哈顿结构两方面。

单层绕障 Steiner 最小树的构建方法主要包含四类：先构造再替换法、不确定性算法、基于生成图的方法、精确算法。第一种方法主要是在不考虑障碍物的情况下先构建布线端点集合的 Steiner 最小树，然后对其中穿过障碍物的边替换成经过障碍物边界的布线边[120]，该类算法过程简单，但容易获得较低质量的布线方案。不确定性算法是基于一些元启发式策略的[118,121]。文献[118]基于粒子群优化算法设计了绕障 X 结构 Steiner 最小树(Obstacle-Avoiding X-architecture Steiner Minimal Tree，OAXSMT)的构造方法。文献[121]采取基于局部搜索的蚁群算法构造考虑障碍物的矩形 Steiner 最小树(Obstacle-Avoiding Rectilinear Steiner Minimal Tree，OARSMT)问题，并可处理形状较为复杂的障碍物。很多绕障算法都属于基于生成图的方法，其中生成图一般包含引脚端点和部分障碍物端点，在一定程度上减低了问题求解空间的复杂度，并在此基础上取得线长与运行时间较为折中的方案。Zhou 等引入一般化的连接图，并提出了构建 OARSMT 的有效算法，但只适合于三端点的线网[122]。文献[116]设计考虑障碍物的三角收缩技术，从而提出三阶段方法以求解不同互连结构的绕障 Steiner 最小树问题。

Ajwani 等借助他们前期所构建 RSMT 方法，进一步考虑障碍物存在的情况，设计求解 OARSMT 的有效方法[123]。文献[124]引入 Steiner 点位置概念，通过选取合适的 Steiner 点集合以获得较为有效的解方案。文献[125]提出了一种带启发策略的迷宫算法用以求解 OARSMT，与上述属于串行算法的绕障工作不一样，文献[125]还借助基础图形处理单元设计高效的并行算法。第四种方法是能够得到准确方案的精确算法[126, 127]。文献[126]是基于 GeoSteiner 方法的两阶段算法，首先构造考虑障碍的完全 Steiner 树(Full Steiner Trees, FSTs)，继而构建整数规划模型并利用分支定界策略从中选取若干 FSTs 用于构建最后的 OARSMT。文献[127]进一步设计求解 OARMST 问题的准确算法，从而可处理形状更为复杂的直线型障碍物。

4.2.3 多层 Steiner 最小树

随着制造工艺的不断进步，多层布线已成为 VLSI 自动化设计领域的研究热点之一，因而多层布线算法显得十分重要[3]。而上述两个小节的相关研究成果均是在平面下进行单层 Steiner 最小树的构造，尚未涉及到多层芯片下的 Steiner 最小树构造问题。本小节将介绍多层 Steiner 最小树的构造问题，包括未考虑障碍物和考虑到障碍物存在两方面工作。

文献[128]提出一种并行遗传算法用于构造多层矩形 Steiner 最小树(Multilayer Rectilinear Steiner Minimal Tree, ML-RSMT) 问题，并优化了会妨碍到线长和时延估计的拐弯数。Lin 等进一步考虑障碍物情况，首次构建了多层绕障矩形 Steiner 最小树 (Multilayer Obstacle-Avoiding Rectilinear Steiner Minimal Tree, ML-OARSMT)模型[129]，但并未考虑到实际芯片设计中更为重要的布线约束——定向边(Preferred Direction)约束。Yildz 等考虑到定向边约束，提出了定向边 Steiner 树布线模型，但尚未考虑障碍物[130]。Liu 等构建了绕障定向矩形 Steiner 树问题(Obstacle-Avoiding Preferred Direction Rectilinear Steiner Tree，OAPD-RST)，可处理更接近实际设计条件的布线模型，并取得可接受的布通率[131]。文献[132]将基于 Steiner 点位置选取策略的单层 OARSMT 构建方法扩展到多层绕障定向 Steiner 树模型的求解中。文献[133]首先确定所有布线引脚的连接顺序，再采取贪心策略依次连接相邻引脚，从而提出一种有效的绕障定向矩形 Steiner 树的求解方法。

上述多层 Steiner 最小树工作都是基于曼哈顿结构，即求解多层矩形 Steiner 最小树的构建问题。而对于非曼哈顿结构下多层绕障 Steiner 最小树的构建，文献[134]开展了相关工作。该方法考虑到线长和通孔数的优化，分别对每个布线层进行绕障 Steiner 最小树的构建工作，再为每两个毗邻布线层寻找最小的连接

路径，是首次考虑到 X 结构下多层绕障 Steiner 最小树的构建问题。但该方法将多层绕障 Steiner 最小树问题转换为多个单层绕障 Steiner 最小树问题，未能从多层结构的全局角度寻找解方案，很大程度影响布线解的质量，至于如何影响及其造成布线质量的退化程度将在后续的算法性质分析和实验结果分析进行详细描述。

OASMT 问题在曼哈顿结构和 X 结构的发展历程如图 4-1 所示。随着矩形结构、多层布线及定向边约束的引入，OASMT 问题进一步发展为 OARSMT，ML-OARSMT 及 OAPD-RST 等问题，而对于 X 结构布线，则依次发展为 OAXSMT，ML-OAXSMT 及 OAPD-XST 等问题。目前对如图 4-1 中虚线框所表示的 OAPD-XST 问题仍未开展相关研究工作。

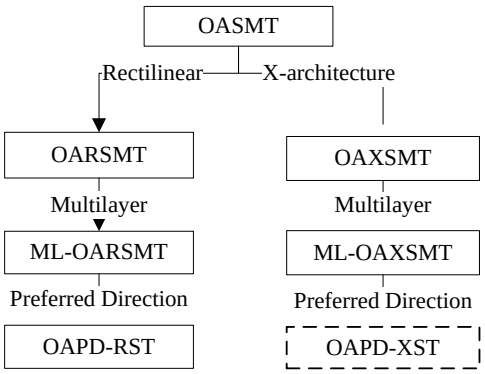


图 4-1 OASMT 问题在曼哈顿结构(Rectilinear)和 X 结构(X-architecture)下的发展历程

鉴于 ML-OAXSMT 问题的重要性和现有相关工作的局限性，本章基于多层结构的全局角度并以优化布线总代价为目标，构建求解 ML-OAXSMT 问题的有效算法。另外，OAPD-XST 问题的研究也在本章工作中开展，这是第一次开展 OAPD-XST 问题的相关分析工作。

4.3 问题模型及定义

4.3.1 ML-OAXSMT 问题

定义 4.1 (障碍物) 障碍物的形状通常是矩形，也有一些是直线边界的多边形。在布线问题中，任意两个布线障碍物都不能互相重叠，但可以点与点接触或是边与边接触。

定义 4.2 (引脚) 引脚端点可以是在任意布线层上的端点。引脚端点不能处于任意障碍物里面，但可以分布在障碍物的边界或端点处。

定义 4.3 (通孔) 层 z 的通孔是一条连接布线层 z 和 z+1 的边，其两个端点的三维坐标是(x，y，z)和(x，y，z+1)。通孔跟引脚一样也不能处于任意障碍物内，

但可处于障碍物的边界或端点处。

ML-OAXSMT 问题是给定通孔代价 C_v 和布线层数 N_l，集合 $P=\{P_1,P_2,P_3,\ldots,P_n\}$ 作为 n 个芯片引脚的集合，集合 $O=\{O_1, O_2, O_3, \ldots, O_m\}$ 作为 m 个障碍物的集合，在端点集合 P 的基础上，引入一些合适的 Steiner 点，并连接所有的芯片引脚，同时满足(1)连接树的每条边都是 X 结构边且边不穿越任意一个障碍物 O_i，(2)不同布线层采用通孔连接，通孔也不在任意障碍物 O_i 内，从而达到布线总代价的最小化。

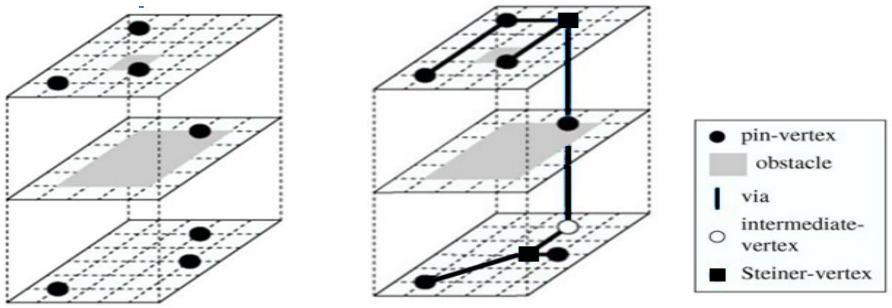
如图 4-2 所示，给出一个 ML-OAXSMT 问题的实例，包含 3 个布线层(N_l = 3)，7 个引脚(n=7)以及两个矩形障碍物(m=2)。图 4-2(b)给出一种 ML-OAXSMT 的解方案，包含 2 个通孔。如图 4-2(a)所示，假设选取引脚集合中的左下端点作为坐标系统的原点，建立相应的三维坐标系统，则引脚集合和障碍物集合的输入信息分别如表 4-1 和表 4-2 所示。表 4-1 的第一行列出了引脚的编号，2 至 4 行则列出了引脚在坐标系统中的 X 轴，Y 轴，Z 轴坐标(Z 轴坐标表示引脚所在布线层的编号)。表 4-2 的第一列列出了障碍物的编号，而 2 至 7 列则分别列出了每个障碍物中左下端点和右上端点在坐标系统中的 X 轴，Y 轴，Z 轴坐标(Z 轴坐标表示障碍物所在布线层的编号)。

表 4-1 ML-OAXSMT 的引脚输入信息

引脚编号	P_1	P_2	P_3	P_4	P_5	P_6	P_7
X 坐标	1	4	3	3	1	1	3
Y 坐标	1	3	5	5	1	5	2
Z 坐标	1	1	1	2	3	3	3

表 4-2 ML-OAXSMT 的障碍物输入信息

障碍物编号	左下端点 X 坐标	左下端点 Y 坐标	左下端点 Z 坐标	右上端点 X 坐标	右上端点 Y 坐标	右上端点 Z 坐标
O_1	1	1	2	4	5	2
O_2	2	2	3	3	3	3



(a) 引脚和障碍物的分布图[129]　　(b) 构建的 ML-OAXSMT

图 4-2 ML-OAXSMT 问题模型

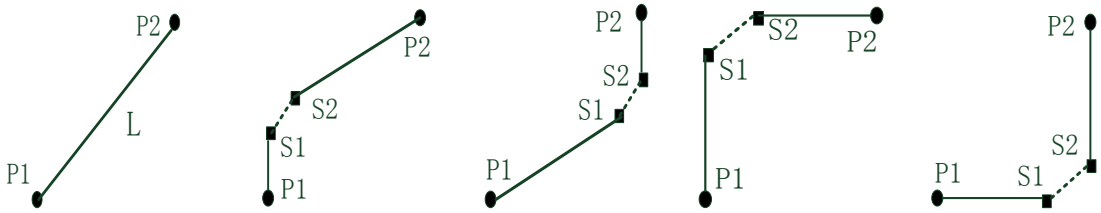
4.3.2 相关定义

定义 4.4 (0 选择) 如图 4-3 (a)所示，(x_1，y_1，z_1)和(x_2，y_2，z_2)为边 *L* 的两个端点 *P1* 和 *P2*，其中 $x_1<x_2$。边 *L* 对应的 pseudo-Steiner 点的选择如图 4-3 (b)所示，从 *P1* 先引曼哈顿结构边至 *S1*，再由通孔连接到 *P2* 所在布线层上的 *S2*(若是 *P1* 和 *P2* 在同一布线层，则不需要引入通孔，此时 *S1* 和 *S2* 是重合的，下述定义中同理)，最后，从 *S2* 引 X 结构边至 *P2*，则称作 0 选择。

定义 4.5 (1 选择) 如图 4-3(c)所示，从 *P1* 先引 X 结构边至 *S1*，再由通孔连接到 *P2* 所在布线层上的 *S2*，最后，从 *S2* 引曼哈顿结构边至 *P2*，则称作 1 选择。

定义 4.6 (2 选择) 如图 4-3 (d)所示，从 *P1* 先引竖直边至 *S1*，再由通孔连接到 *P2* 所在布线层上的 *S2*，最后，从 *S2* 引水平边至 *P2*，则称作 2 选择。

定义 4.7 (3 选择) 如图 4-3 (e)所示，从 *P1* 先引水平边至 *S1*，再由通孔连接到 *P2* 所在布线层上的 *S2*，最后，从 *S2* 引竖直边至 *P2*，则称作 3 选择。



(a) 线段 *L*　(b) 0 选择　(c) 1 选择　(d) 2 选择　(e) 3 选择

图 4-3 线段 *L* 及其四种边选择方式

4.4 算法的设计与实现

现有文献将 PSO 应用于离散问题，主要有将速度作为位置变化的概率、直接将连续 PSO 用于离散问题的求解以及重新定义 PSO 操作算子三种策略。为解决实际工程应用问题，一些研究人员一直跟踪 PSO 的研究进展[72,76,118, 135,136]。为此，根据这些相关工作的基础上，本章重新设计了适合 ML-OAXSMT 问题的有效离散 PSO 算法，即 ML-OAXSMT-PSO 算法。

4.4.1 编码策略

ML-OAXSMT-PSO 算法设计了适合 X 结构 Steiner 最小树和多层布线结构的多层边点对编码策略。多层边点对编码采用多层布线树的边集合编码相应的 Steiner 树，每条边的编码采用四位数字串表示。前两位表示边所连接两引脚的引脚编号，第三位走线方式位 *pspc* 表示边的 pseudo-Steiner 点选择方式，其中 *pspc* 的取值包含 0、1、2、3，分别表示 0 选择、1 选择、2 选择、3 选择等四种方式。

编码的最后一位是走线状态位 *label*，记录布线边走线是否穿越障碍物及其产生的通孔数，正数代表未穿越障碍物，负数代表穿越障碍物，数值大小则代表通孔数。如果一个线网有 *n* 个引脚，其对应的生成树就有 *n*-1 条边，*n*-1 位走线方式位，*n*-1 位走线状态位，加上 1 位的适应度函数值。另外，一条边需要两位引脚编号表示，故相应粒子的总长度为 4(*n* -1)+1。

假设一棵候选布线树(*n*=7)的粒子编码可表示如下数字串：

1 2 1 0	3 4 3 1	5 6 2 0	6 7 2 0	1 3 1 0	4 6 1 1	19.8284

其中，数字串的末位‘19.8284’ 表示粒子的适应度值，每四位数字子串编码一条边。在数字串中第 2 条边的编码为(3 4 3 1)，该四位数字子串表示引脚 3 和引脚 4 以 3 选择方式连接，第四位‘1’ 表示该边未穿越障碍物，但穿过 1 个布线层，即通孔数为 1。

4.4.2 适应度计算函数及惩罚机制

ML-OAXSMT 问题中布线树的代价除了包含布线树的长度，还需计算通孔的布线代价。在本章的编码策略下，两个引脚 *i* 和 *j* 的距离计算如下所示。

$$Odis(i,j)=\begin{cases}(\sqrt{2}-1)\min(xy)+\max(xy), \text{if } pspc=0 \text{ or } 1\\ \left|x_i-x_j\right|+\left|y_i-y_j\right|, \quad \text{else}\end{cases} \tag{4-1}$$

其中，$\min(xy)=\min(|x_i-x_j|,|y_i-y_j|)$，$\max(xy)=\max(|x_i-x_j|,|y_i-y_j|)$，$x_i$ (x_j) 和 y_i (y_j)分别表示引脚 *i*(*j*)的水平坐标和垂直坐标。$|x_i-x_j|$($|y_i-y_j|$)表示相应的绝对值，min(max)则表示$|x_i-x_j|$和$|y_i-y_j|$之间的最小值(最大值)。

通孔代价 $C_{via}(i,j)$ 则表示为$|z_i-z_j|\times C_v$，其中 z_i(z_j) 表示引脚 *i*(*j*)所在布线层的 Z 轴坐标，C_v 的取值一般为 3 或 5[129]。因此，布线树 *T* 的布线总代价的计算如下所示。

$$\cos t(T)=\sum_{(i,j)\in T}\cos t(i,j)=\sum_{(i,j)\in T}(Odis(i,j)+C_{via}(i,j)) \tag{4-2}$$

布线树的边如果穿越障碍物，即违反 ML-OAXSMT 问题的约束，属于不可行解，但不可行解中可能会携带部分最优子结构信息，不能随意删除。为了解决这个问题，本章设计了合适的惩罚函数，用以惩罚布线边穿越障碍物的情况，以一定概率保留这些不可行解，从而确保最终解方案的质量。同时，通孔数是 VLSI 芯片设计中考虑到可布性和可制造性的一个重要优化指标，而在多层布线环境下，通孔的产生是不可避免的，但可采取有效的措施以减少通孔数。因此，本章在适应度计算函数中引入的惩罚机制，对产生通孔情况也施加一定的惩罚力度。

惩罚函数是用来对优化问题中出现不满足约束条件的解方案施以相应的惩罚，即惩罚目标函数，达到从目标函数上一定程度标识违反约束的程度，并同时保留这类解方案不被随意删除，在后续的进化过程中有机会呈现该类解方案所保留的局部最佳信息，增强种群的多样性。基于此，本算法的适应度函数为

$$F(X)=\sum_{e\in T} penalty(e)\times \cos t(T) \tag{4-3}$$

其中，e 代表属于布线树 T 的边， $penalty(e)$ 表示边 e 对应的惩罚因子。

在实验测试中， $penalty(e)$ 的计算公式依次选取 f_1，f_2 和 f_3 三种类型函数，如公式(4-4)至(4-6)所示。

$$f_1(x)=\begin{cases} lb, & x>0 \\ ub, & x<0 \\ 1, & x=0 \end{cases} \tag{4-4}$$

$$f_2(x)=\begin{cases} \dfrac{lb-1}{l\max}x+1, & x>0 \\ \dfrac{ub-1}{-l\max}x+1, & x\le 0 \end{cases} \tag{4-5}$$

$$f_3(x)=\begin{cases} \dfrac{1}{\dfrac{1}{lb}+\dfrac{1}{1+e^{x}}}, & x>0 \\ \dfrac{1}{\dfrac{1}{ub}+\dfrac{1}{1+e^{-x}}}, & x\le 0 \end{cases} \tag{4-6}$$

其中，lb 和 ub 是用户自定义且值大于 1 的参数，本章算法取 lb=10/9，ub=2。x 表示边 e 编码中走线状态位的数值，$lmax$ 代表布线层的最大编号。此外，为了区别同一层中绕过障碍物与穿过障碍物两种情况，本章设置在同一布线层不违反绕障约束的边 e 的走向状态位值为 0，而违反绕障约束的边走线状态位值为-0.9。

4.4.3 粒子的更新公式

本章算法引入了遗传算法中的交叉和变异算子，并结合并查集思想，构造了适合 ML-OAXSMT 问题的离散型 PSO 算法的操作算子。本章粒子的更新公式如下所示。

$$X_i^t=N_3(N_2(N_1(X_i^{t-1},w),c_1),c_2) \tag{4-7}$$

其中，w 是惯性权重，c_1 和 c_2 是加速因子， N_1 表示变异算子，N_2 和 N_3 表示交

叉算子。这里假设 r_1，r_2，r_3 是[0，1)的随机数。本章算法的惯性权重 w 和加速因子 c_1，c_2 的取值均采用线性递减的策略[85]进行更新。

文献[100]构造的 X 结构 Steiner 最小树工作中，更新操作只考虑到 Steiner 点的变换，使得 Steiner 树在绕障能力方面受限。例如在图 4-4(a)所示的例子，边 $\overline{P1P3}$ 违反绕障约束，在文献[100]中只考虑 Steiner 点变换的更新操作，可通过将边 $\overline{P1P3}$ 的选择从 0 方式变换为 2 方式，则可得到如图 4-4(b)所示的不违反绕障约束的布线树，即边 $\overline{P1P3}$ 不穿越障碍物。而在图 4-4(c)所示的情况，如果单纯考虑 Steiner 点的变换，只能得到类似图 4-4(d)所示的违反绕障约束的 Steiner 树。从而造成布线树在进化过程中不具备绕开障碍物的可能，而需改变布线树的拓扑，即在更新操作中引入边变换策略，将边 $\overline{P2P3}$ 的连接转换为边 $\overline{P1P3}$ 的连接，可得到如 4-4(b)所示的绕障 Steiner 树。因此，本章算法在求解 ML-OAXSMT 问题中引入边变换策略。具体的变异算子、交叉算子及其边变换策略的设计过程跟第三章的设计过程类似，这里不再赘述。

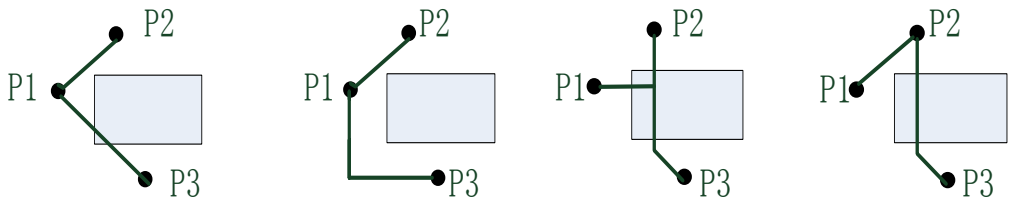


图 4-4 Steiner 点变换策略与边变换策略

4.4.4 预处理策略

ML-OAXSMT-PSO 算法设置了走线状态位用以记录走线是否穿越障碍物及其产生的通孔数，走线状态位的计算复杂度是影响算法速度的主要因素之一。如果未采取任何措施，需要在迭代中每次计算粒子适应度函数值时计算走线状态位，而这就需要重新判断粒子解码后相应布线树的绕障情况及其通孔数的计算。这样将导致重复判断很多布线边的绕障情况和通孔数的重复计算，从而增加算法的计算时间。为此，本章算法引入计算走线状态位的预处理策略，在算法未进入迭代过程前，计算所有可能走线边的走线状态位并存储在数组中，算法在进行迭代时直接查询所需要的走线状态位的数值，从而大大降低了走线状态位的计算次数，避免一些冗余计算。具体计算次数的减少程度将在实验部分给出。

4.4.5 ML-OAXSMT-PSO 算法的流程

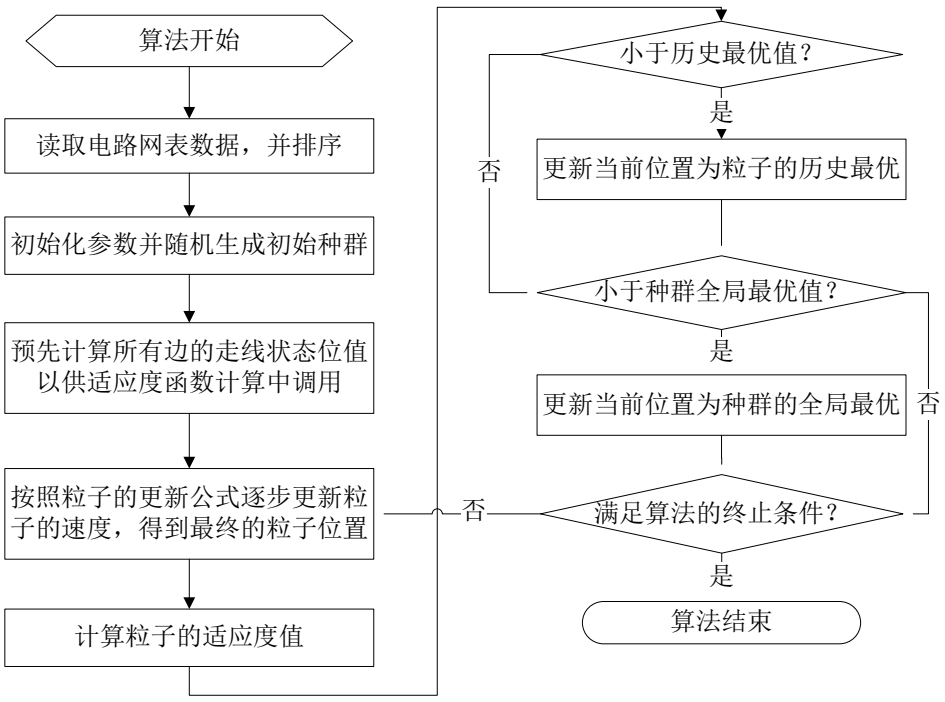


图 4-5 本章算法的流程图

本章算法的流程图如图 4-5 所示，其执行步骤可归纳如下：

步骤 1：读取基准测试电路网表数据，并按层数和坐标大小进行升序排序；

步骤 2：初始化种群的大小、迭代次数及惯性权重、加速因子等参数，并随机产生初始种群；

步骤 3：预先计算所有边走线方式的走线状态位值，用以衡量绕障情况和穿层情况，并生成相应的存储数组以供算法主循环中计算适应度函数值时直接查询；

步骤 4：根据公式(4-7)更新每个粒子的位置和速度；

步骤 5：根据公式(4-3)计算每个粒子的适应度值；

步骤 6：检查每个粒子的适应度值，如果优于它的历史最佳位置，则更新为该粒子的历史最佳位置；

步骤 7：检查每个粒子的适应度值，如果优于整个种群的全局最佳位置，则更新为种群的全局最佳位置；

步骤 8：检查是否满足终止条件。如果满足，算法结束，输出最优布线树，否则转到步骤 4。

4.4.6 ML-OAXSMT-PSO 算法的性质分析

目前关于 ML-OAXSMT 问题所提出的算法，是将多层绕障 Steiner 最小树问题转换为多个单层绕障 Steiner 最小树问题[134]，这类方法称为基于单层结构的贪心方法。而本章算法是从多层结构的全局角度考虑多层绕障 Steiner 最小树问题，能够有效地利用合适的通孔，从而能够搜索到更为优秀的问题解空间，以达到更强的布线总代价优化能力。本小节将通过三个性质分析本章算法与基于单层结构的贪心方法在布线总代价方面的优化能力以及布线模型区别。

性质 4.1 对于有障碍物存在的多层 Steiner 最小树问题，本章算法相对基于单层结构的贪心方法能够取得布线总代价的优化率为[0%，100%)。

在图 4-6 中考虑 *P1*、*P2* 和 *P3* 的最短连接路径，如果采用基于单层结构的贪心方法进行布线，如图 4-6(a)所示，首先需要为 *P2* 和 *P3* 所在的布线层以及 *P1* 所在的布线层分别进行绕障布线，再引入合适的通孔以完成最后的多层布线，所需的布线总代价为：$2r+C_v+b$。而本章基于多层结构的全局角度考虑布线问题，如图 4-6(b)所示，则所需的布线总代价为：$b+2C_v$。在如图 4-6 所示这一类问题中，本章算法相对基于单层结构的贪心方法，)布线总代价减少率为$(2r-C_v)/(2r+C_v+b)$。当 $2r$ 的长度值远大于 C_v 和 b 时，本章算法所取得布线总代价的减少率将接近 100%。而在所有基准测试电路中，存在电路的局部问题所对应 $2r$ 的长度值远大于 C_v 和 b，所以该情况下所取得布线总代价的优化率为[0%，100%)。

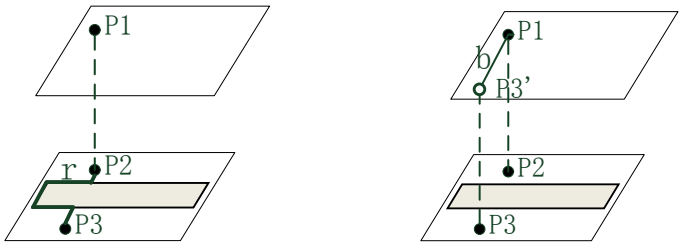
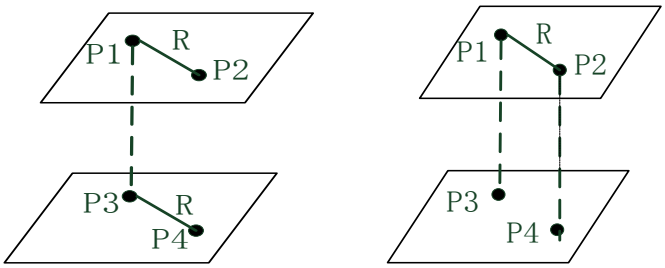


图 4-6 两类算法在有障碍物存在的多层 Steiner 最小树问题中布线总代价优化能力对比

性质 4.2 对于无障碍物存在的多层 Steiner 最小树问题，本章算法相对基于单层结构的贪心方法能够取得布线总代价的优化率为[0%，50%)。

在图 4-7 中考虑 *P1*、*P2*、*P3* 和 *P4* 的最短连接路径，如果采用基于单层结构的贪心方法进行布线，如图 4-7(a)所示，先为 *P1* 和 *P2* 所在的布线层以及 *P3* 和 *P4* 所在的布线层分别进行布线，再引入合适的通孔以完成最后的多层布线，需要的布线总代价为 $2R+C_v$。而本章基于多层结构的全局角度考虑布线问题，如图 4-7(b)所示，则需要的布线总代价为 $R+2C_v$。在如图 4-7 所示这一类问题中，本章算法相对基于单层结构的贪心方法，布线总代价减少率为$(R-C_v)/(2R+C_v)$。

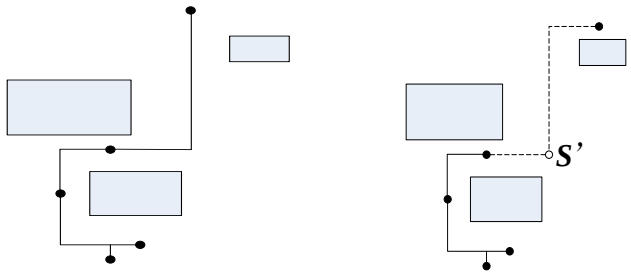
当 R 的长度值远大于 C_v 时，本章算法所取得布线总代价的减少率接近 50%。而在大多数的基准测试电路，R 的长度值是远大于通孔代价 C_v，所以该情况下所取得布线总代价的优化率为[0%，50%)。



(a) (b)

图 4-7 两类算法在无障碍物存在的多层 Steiner 最小树问题中布线总代价优化能力对比

文献[74]定义了关于曼哈顿结构下的绕障 Steiner 最小树问题的连通模型和非连通模型，本章将这两种模型的定义进一步扩展到 X 结构下。



(a)连通模型 (b)非连通模型

图 4-8 绕障问题的两类情况

定义 4.8 (连通模型) 在布线问题中，每个引脚都可跟其他引脚中的至少一个引脚只采用一次的边选择方式(0 选择或 1 选择或 2 选择或 3 选择)便可在不违反绕障约束的条件下进行连接，该布线问题属于连通模型。图 4-8(a)给出的例子是连通模型。

定义 4.9 (非连通模型) X 结构下绕障 Steiner 树问题中，由于在一些情况下，pseudo-Steiner 点会处于障碍物内，这些 pseudo-Steiner 点不能够被选取，从而导致在布线问题中存在至少一个引脚不能通过只采用一次的边选择方式连接到其他引脚集合中的任意一个。图 4-8(b)给出的例子是非连通模型，其中，S'的引入表示右上角的引脚需要通过至少两次的边选择方式才能连接到其他引脚中的一个。

性质 4.3 在本章多层布线的编码策略下，使得在单层绕障 Steiner 最小树问题中部分非连通模型可变为连通模型。所有基准测试电路在本章多层布线的编码

策略下均是连通模型。

在多层绕障 X 结构 Steiner 最小树问题中，本章基于多层结构的全局角度考虑多层布线问题，可通过引入合适的通孔，绕开障碍物。具体过程可参照性质 4.1 中关于图 4-6 的分析。若采用基于单层结构的贪心方法考虑该问题，对于图 4-6(a)中的 *P2* 和 *P3* 两引脚之间因为障碍物的存在，在该层成为非连通模型，导致了无法通过多层边点对的编码策略寻找有效解。而本章算法基于多层结构的全局角度考虑多层绕障 Steiner 最小树问题，可通过引入额外的通孔，同时引入该布线层的上一层布线路径，寻找到如图 4-6(b)所示的可行解。本章算法通过在多层布线问题中所有基准测试电路[129]的实验仿真，发现所有基准测试电路均可通过引入合适的通孔成为连通模型。

4.4.7 ML-OAXSMT-PSO 算法的复杂度分析

引理 4.1 假设种群大小为 *pop*，迭代次数为 *iters*，引脚个数为 *n*，障碍物的个数为 *m*，则未加入预处理策略的算法复杂度为 $O(pop\times iters\times m\times n)$，而加入预处理策略的算法复杂度为 $O(m\times n^2+pop\times iters\times n\log n)$。

证明：(1) 未加入预处理策略，如本章算法执行步骤 4 和步骤 5 所示，算法的内部循环包括变异操作，交叉操作和适应度值计算操作。在变异和交叉操作中，因为其中的并查集策略所需时间略高于线性时间[108]，所以其中复杂度为 $O(n\log n)$ 的排序步骤主导这两个操作的算法时间复杂度。而计算适应度值的过程，复杂度主要取决于计算走线标志位 *label* 的值，即判断走线绕障碍物的次数的复杂度 $O(m\times n)$。由于如图 4-9 所示，在所有基准测试电路中 *m* 大于 $\log n$，因此，算法内部循环中三个主要操作的时间复杂度为 $O(n\log n+m\times n)=O(m\times n)$。而在算法的外循环阶段的时间复杂度是跟种群大小 *pop* 和迭代次数 *iters* 相关。综上，本章算法未加入预处理策略的时间复杂度为 $O(pop\times iters\times m\times n)$。

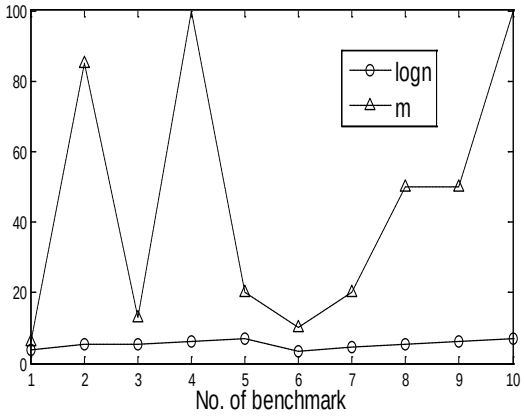


图 4-9 所有基准测试电路中 *m* 与 $\log n$ 的大小关系

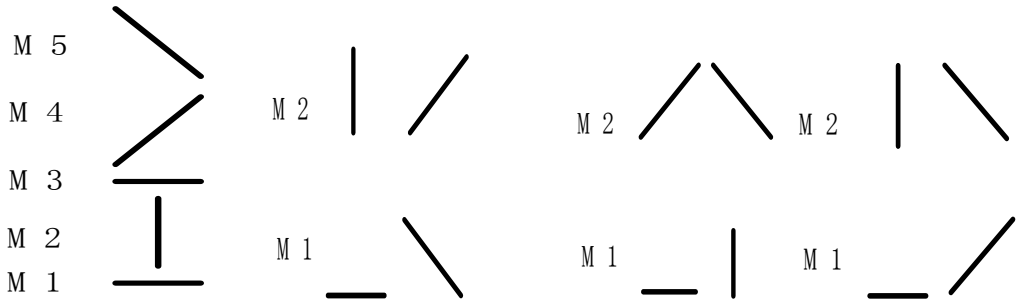
(2) 加入预处理策略，在算法未进入算法迭代过程之前，本章算法预计算所有可能走线边的走线标志位 *label* 的值，并存储在数组中。该预处理策略需要计算 $n\times(n-1)$ 条边绕 m 个障碍物的情况，故预处理策略时间复杂度为 $O(n^2\times m)$。同时，算法进入迭代阶段中，可直接查询所需 *label* 值，故算法主阶段的时间复杂度为 $O(pop\times iters\times n\log n)$。因此，本章算法加入预处理策略的复杂度为 $O(n^2\times m+pop\times iters\times n\log n)$。从后续表 4-3 的实验结果中可发现在所有基准测试电路中加入预处理策略的 $O(n^2\times m+pop\times iters\times n\log n)$ 远快于未加入该策略的 $O(pop\times iters\times m\times n)$。

4.5 OAPD-XST 问题模型及算法设计

4.5.1 OAPD-XST 问题的定向边约束模型

关于曼哈顿结构下的 OAPD-RST 问题已经开展了相关研究，其定向边约束模型有两种：(1) 在奇数层只能走垂直方向的布线边，而在偶数层只能走水平方向的布线边；(2) 在奇数层只能走水平方向的布线边，而在偶数层只能走垂直方向的布线边。然而关于 X 结构 OAPD-XST 问题中的定向边约束模型及其问题研究尚未开展。本章工作将设计一种用于求解 OAPD-XST 问题的算法以探索 OAPD-XST 问题的不同定向边约束模型的优劣性，。

OAPD-XST 问题的定向边约束模型主要可包括以下四种类型：(1) 每五层交替走线模型，即这五层依次的走线方向分别为 0°，90°，0°，45°和 135°，如图 4-10(a)所示，称为模型 *a*；(2) 奇数层的走线方向只能是 0°和 135°，而偶数层的走线方向只能是 45°和 90°，如图 4-10(b)所示，称为模型 *b*；(3) 奇数层的走线方向只能是 0°和 90°，而偶数层的走线方向只能是 45°和 135°，如图 4-10(c)所示，称为模型 *c*；(4) 奇数层的走线方向只能是 0°和 45°，而偶数层的走线方向只能是 90°和 135°，如图 4-10(d)所示，称为模型 *d*。



(a)模型 *a*　　(b) 模型 *b*　　(c) 模型 *c*　　(d) 模型 *d*

图 4-10 OAPD-XST 问题中四种定向边约束模型

综上所述的四种模型的优劣性可通过通孔数、违反约束数(定义 4.11)以及布线总代价三个指标衡量，并从中得到 OAPD-XST 问题的最佳定向边约束模型，以期为非曼哈顿结构的芯片设计提供一定的理论支持。

4.5.2 OAPD-XST 算法

OAPD-XST 算法的流程可概括如下。

步骤 1：根据定义 4.10 重新定义每个布线边在不同布线层的布线代价。

定义 4.10 (OAPD-XST 问题中边的布线代价) 给定布线层 *LE* 上的一条边 *e*，如果该布线边 *e* 的走线方向满足布线层 *LE* 的定向边约束，则该边 *e* 的代价(*costpd*(*e*))定义为其在多层边点对编码策略下的原始距离(*olength*)。如果不满足，*costpd*(*e*)则定义为 *olength*×*a*。参数 *a* 是自定义的大于 1 的数，在本章实验中，参数 *a* 的值设定为 3。

步骤 2：调用 ML-OAXSMT-PSO 算法，并重新按照公式(4-8)定义其适应度值函数。

$$C(T_x)=\sum_{e_i\in T_x}(\cos tpd(e_i)+\Delta Z\times C_v) \tag{4-8}$$

其中，*costpd*(e_i)是布线 e_i 边的布线代价，该布线代价可根据定义 4.10 进行计算，C_v 是一个通孔的布线代价，ΔZ 表示由边 e_i 产生的通孔数。

步骤 3：按照重新定义的适应度函数，调用 ML-OAXSMT-PSO 算法后，依次计算所构造的 OAPD-XST 的布线总代价、通孔数和总违反约束数。其中总违反约束数按照定义 4.11 进行统计。最终，算法可获得 OAPD-XST 的布线总代价、通孔数和总违反约束数。

定义 4.11 (违反约束数) 每个 OAPD-XST 定向边约束模型中都有不同的走线方向约束。如果布线边 *e* 的走线方向与相应布线层的定向边约束不一致，则该布线边产生一个违反约束数。

4.6 算法仿真与结果分析

所有算法均使用 C/C++语言在 3.30GHz CPU 和 8.00GB RAM(Windows XP 系统)平台上进行编程运行。用以验证算法性能的基准测试电路来自于文献[129]。为了研究相关策略的有效性和本章算法的性能，相关实验仿真和算法对比将在 4.6.1 至 4.6.5 节详细给出，包括本章所提出预处理策略的有效性验证、本章算法基于三种不同类型惩罚机制的比较、本章算法与三种多层绕障矩形 Steiner 最小树布线算法的比较、本章算法与两种多层绕障 X 结构 Steiner 最小树算法的比较以及 OAPD-XST 问题中不同模型优劣性的比较。

文献[72]已经在 Oliver30 TSP 最小化问题中进行大量参数组合测试，本章参考这种测试策略对所有测试电路进行参数选择。本章采用线性递减的策略和最优参数组合用以更新本算法的加速因子和惯性权重。本章所采用的基准测试电路包括 10 组，分别是 5 组(ind1-ind5)来自 Synopsys 和 5 组(rt1-rt5)来自文献[129]产生的测试电路。这些测试电路的名称和参数情况如表 4-3 中第 1-2 列所示，其中 n 是引脚个数，m 是障碍物个数，N_l 是芯片的总层数。从表中可看出基准测试电路的规模包括引脚个数从 25 到 1000、障碍物数目从 6 到 100、布线层数从 5 到 10。

4.6.1 预处理策略的有效性验证

本章算法引入计算走线状态位的预处理策略，大大降低了走线状态位的计算次数，避免一些冗余计算，具体计算次数的减少程度如表 4-3 所示。在所有基准测试电路中进行测试，采取预处理策略(WP)相对未采取预处理策略(WOP)走线状态位的计算次数平均减少 95.14%，从实验结果上说明了预处理策略的有效性。

表 4-3 预处理策略带来计算次数的减少情况

实例	$n/m/N_l$	WOP	WP	减少率(%)
ind1	50/6/5	2299890	1854	99.98
ind2	200/85/6	3806905	37481	99.75
ind3	250/13/10	502468	46287	97.70
ind4	500/100/5	9447739	748815	98.02
ind5	1000/20/5	487215	260445	86.64
rt1	25/10/10	89628	183	99.95
rt2	100/20/10	403012	6190	99.62
rt3	250/50/10	1551069	74281	98.80
rt4	500/50/10	1075891	284737	93.38
rt5	1000/100/5	2418451	2169159	77.58
AVG		2208227	362943	95.14

4.6.2 基于三种不同类型惩罚机制的比较

针对 ML-OAXSMST 问题，本章设计了基于惩罚机制的适应度函数，该惩罚机制同时考虑到障碍物出现的情况以及通孔数的产生，以较好地评估可行解的优劣性，从而将 ML-OAXSMT 问题转换成无约束问题。三种惩罚函数 f_1，f_2 和 f_3 依次选取为适应度函数中的惩罚机制以验证算法的有效性，并从中选取一个最佳函数作为本章算法的惩罚机制。从图 4-11 可看出三种惩罚函数 f_1，f_2 和 f_3 的三个极值点分别是(0，1)，(-10，2)和(10，10/9)，而三者最大的区别之处是它们的斜率不一样。

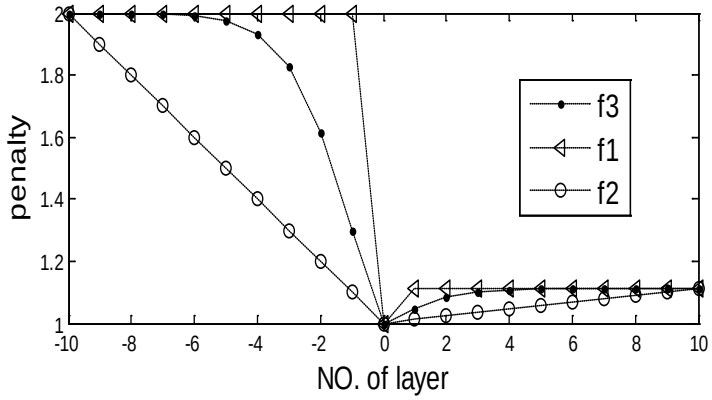
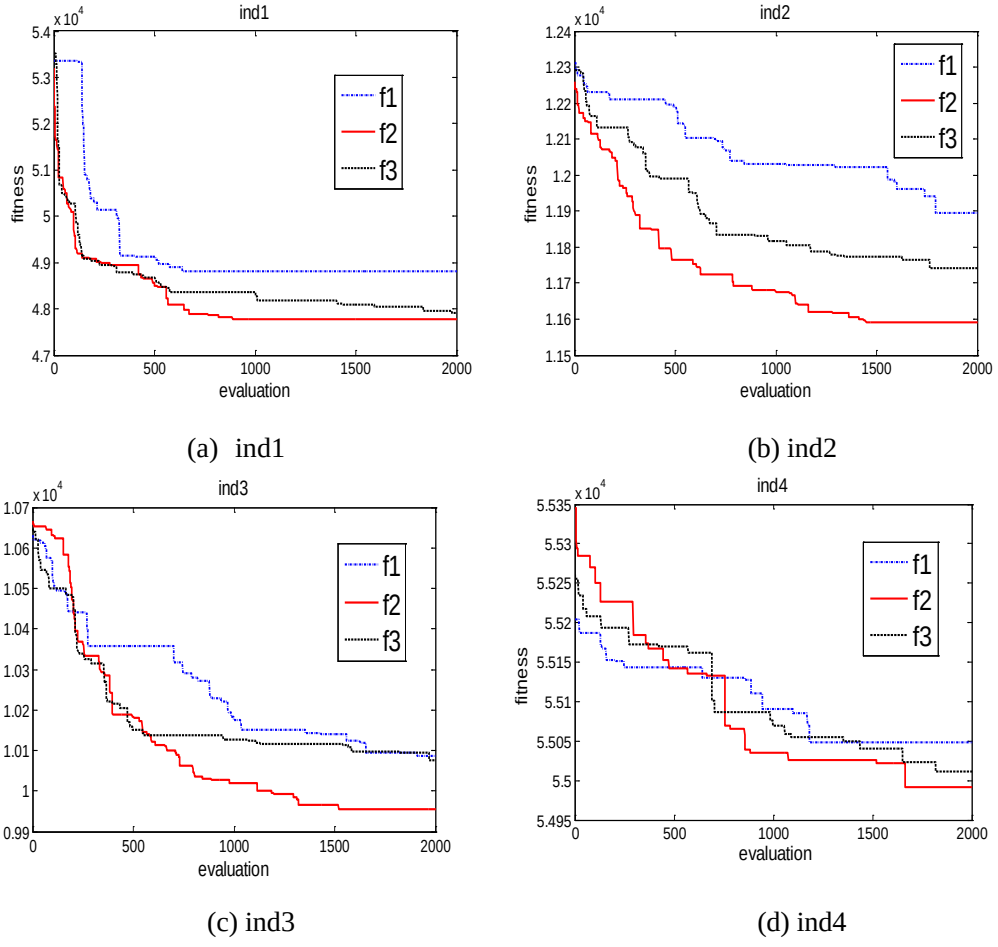


图 4-11 三种不同的惩罚函数 f_1，f_2 和 f_3



(a) ind1　　(b) ind2

(c) ind3　　(d) ind4

图 4-12 算法基于三种惩罚函数的收敛情况

基于不同惩罚函数的 ML-OAXSMT-PSO 算法所对应的实验结果如图 4-12 所示，算法分别在 ind1，ind2，ind3，ind4 这些基准电路中进行测试。从图 4-12 可看出，基于 f_2 的算法收敛情况优于基于 f_1，f_3 的算法收敛情况。另外，从图 4-12 所示的四组收敛曲线的波动性可看出基于 f_2 的 ML-OAXSMT-PSO 算法相对其他两种具有更好的种群多样性。基于 f_1，f_2 和 f_3 的 ML-OAXSMT-PSO 算法在其他

基准测试电路中，包括 ind5，rt1-rt5，所获得的收敛曲线情况与 ind1-ind4 的相似。因此，ML-OAXSMT-PSO 算法选取相对较优的 f_2 作为惩罚函数，并应用于后续实验的 ML-OAXSMT-PSO 算法中。

4.6.3 与三种多层绕障矩形 Steiner 最小树算法的比较

为了验证本章算法的有效性，本章算法(OURS)将在基准测试电路上的实验结果跟文献[129]中提出的先构造后修正的方法(CC)、基于生成图的 ML-OARSMT 算法(SG)以及文献[132]中提出的 ML- OARSMT 算法(VG)进行对比，对比结果如表 4-4 至 4-7 所示。

表 4-4 通孔代价 C_v=3 时与三种 ML-OARSMT 算法在布线总代价方面的对比

实例	$n/m/N_l$	布线总代价				(X-OURS)/X (%)		
		CC	SG	VG	OURS	X=CC	X=SG	X=VG
ind1	50/6/5	82556	55537	54207	48996	40.65	11.78	9.61
ind2	200/85/6	17568	12512	12008	11652	33.67	6.87	2.96
ind3	250/13/10	17837	10973	10555	9654	45.88	12.02	8.54
ind4	500/100/5	273235	77033	77292	54965	79.88	28.65	28.89
ind5	1000/20/5	23314944	14515511	14599961	14356523	38.42	1.10	1.67
rt1	25/10/10	5095	4334	4169	3855	24.33	11.05	7.53
rt2	100/20/10	12885	9434	9132	8774	31.90	6.99	3.92
rt3	250/50/10	23233	15569	14750	14084	39.38	9.54	4.52
rt4	500/50/10	29464	22034	21013	21070	28.49	4.38	-0.27
rt5	1000/100/5	38702	27890	26970	27410	29.18	1.72	-1.63
AVG						39.18	9.41	6.57

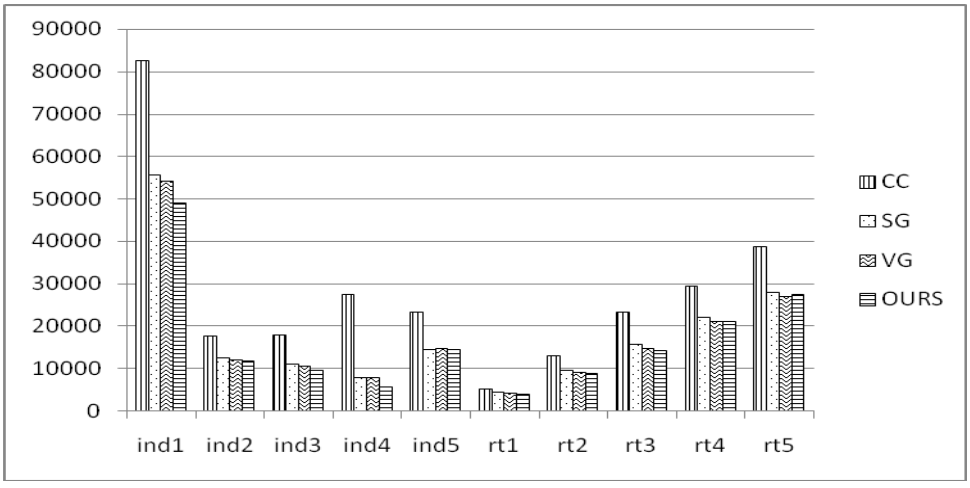


图 4-13 四种算法在通孔代价 C_v=3 时在布线总代价方面的对比柱状图

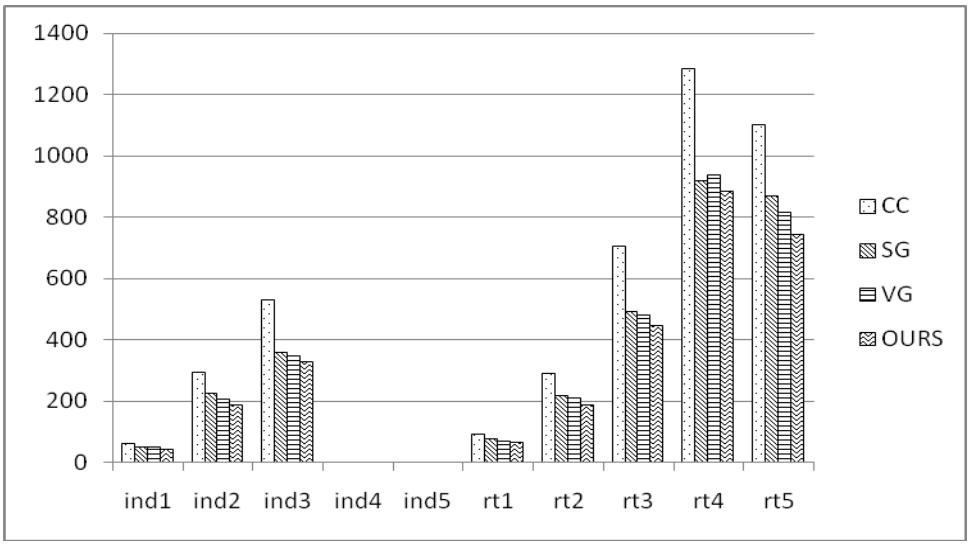


图 4-14 四种算法在通孔代价 C_v=3 时在通孔数方面的对比柱状图

表 4-4 是在通孔代价为 3 时，本章算法与其他算法在布线总代价方面的对比情况。表 4-4 中第 1 列是基准测试电路的名称，第 2 列列举了基准测试电路的引脚数、障碍物数目、总的布线层数，第 3-6 列为三种 ML-OARSMT 算法与本章算法的布线总代价值，最后 3 列为本章算法在布线总代价方面相对三种 ML-OARSMT 算法的改进情况。从表 4-4 可看出，本章算法在 C_v=3 时相对 CC、SG、VG 三种 ML-OARSMT 算法在布线总代价分别取得 39.18%、9.41%、6.57% 的平均减少率，最高的减少率甚至达到 79.88%(在 ind4 上)，说明本算法由于引入 X 结构并且基于多层结构的全局角度开展多层布线，从而相对目前相关工作而言具有较强的布线总代价优化能力。

表 4-5 通孔代价 C_v=3 时与三种 ML-OARSMT 算法在通孔数方面的对比

实例	$n/m/N_l$	#via				(X-OURS)/X (%)		
		CC	SG	VG	OURS	X=CC	X=SG	X=VG
ind1	50/6/5	59	49	49	43	27.12	12.24	12.24
ind2	200/85/6	293	224	206	188	35.84	16.07	8.74
ind3	250/13/10	529	359	348	326	38.37	9.19	6.32
ind4	500/100/5	0	0	0	0	0.00	0.00	0.00
ind5	1000/20/5	0	0	0	0	0.00	0.00	0.00
rt1	25/10/10	91	76	70	66	27.47	13.16	5.71
rt2	100/20/10	290	215	209	187	35.52	13.02	10.53
rt3	250/50/10	705	490	478	445	36.88	9.18	6.90
rt4	500/50/10	1282	918	936	883	31.12	3.81	5.66
rt5	1000/100/5	1102	869	814	744	32.49	14.38	8.60
AVG						26.48	9.11	6.47

表 4-6 通孔代价 C_v=5 时与三种 ML-OARSMT 算法在布线总代价方面的对比

实例	$n/m/N_l$	布线总代价				(X-OURS)/X (%)		
		CC	SG	VG	OURS	X=CC	X=SG	X=VG
ind1	50/6/5	82674	55635	54305	49082	40.63	11.78	9.62
ind2	200/85/6	17389	12899	12420	12028	30.83	6.75	3.15
ind3	250/13/10	17599	11698	11251	10368	41.09	11.37	7.85
ind4	500/100/5	273235	77033	77292	54965	79.88	28.65	28.89
ind5	1000/20/5	23314944	14515511	14599961	14356523	38.42	1.10	1.67
rt1	25/10/10	5277	4486	4309	3987	24.44	11.12	7.47
rt2	100/20/10	13874	9812	9550	9148	34.06	6.76	4.21
rt3	250/50/10	22727	16384	15706	14974	34.12	8.61	4.66
rt4	500/50/10	30542	23883	22885	22836	25.23	4.39	0.22
rt5	1000/100/5	43775	29598	28598	28898	33.99	2.37	-1.05
AVG						38.31	9.36	6.74

表 4-5 是在通孔代价 C_v=3 时本章算法与三种 ML-OARSMT 算法在通孔数方面的对比情况。从表 4-5 可以看出，本章算法在 C_v=3 时，相对三种 ML-OARSMT 算法在通孔数(#via)方面分别取得 26.48%、9.11%、6.47%的平均较少率。其中，由于在 ind4 和 ind5 两组基准测试电路的障碍物是分布在多个布线层，而引脚均在单个布线层上，因而所有算法在这两组基准测试电路中均优化到未产生通孔的情况，即相应的通孔数均为 0。由于本章算法相对三种 ML-OARSMT 算法能取得可观的通孔减少率，说明本章设计的惩罚机制带来了一定的通孔优化能力。此外，为了更形象地阐述文中的实验效果，图 4-13 和图 4-14 分别给出表 4-4 和表 4-5 关于四种算法在通孔代价 C_v=3 时在布线总代价和通孔数两方面的对比数据柱状图。为了使所有柱状图显示在较为统一数量级的坐标上，不妨将图 4-13 中四种算法在测试实例 ind4 的布线总代价值上均除以 10 以及在测试实例 ind5 的布线总代价上均除以 1000（后面的图 4-15 亦采取这种方式）。

表 4-6 和表 4-7 是在通孔代价 C_v=5 时本章算法与三种 ML-OARSMT 算法在布线总代价和通孔数方面的对比情况。本章算法在 C_v=5 时相对 CC、SG、VG 三种 ML-OARSMT 算法在布线总代价分别取得 38.31%、9.36%、6.74%的平均减少率，最高减少率达到 79.88%(在 ind4 上)。而且，本章算法在 C_v=5 时，相对三种 ML-OARSMT 算法在通孔数方面分别取得 22.91%、5.09%、6.47%的平均较少率。因此，实验结果进一步表明本章算法相对三种 ML-OARSMT 算法在不同的通孔代价(C_v=3 和 C_v=5)均能取得布线总代价和通孔数两方面较高的优化率。同样，为了更形象地阐述文中的实验效果，图 4-15 和图 4-16 分别给出表 4-6 和表 4-7

关于四种算法在通孔代价 C_v=3 时在布线总代价和通孔数两方面的对比数据柱状图。

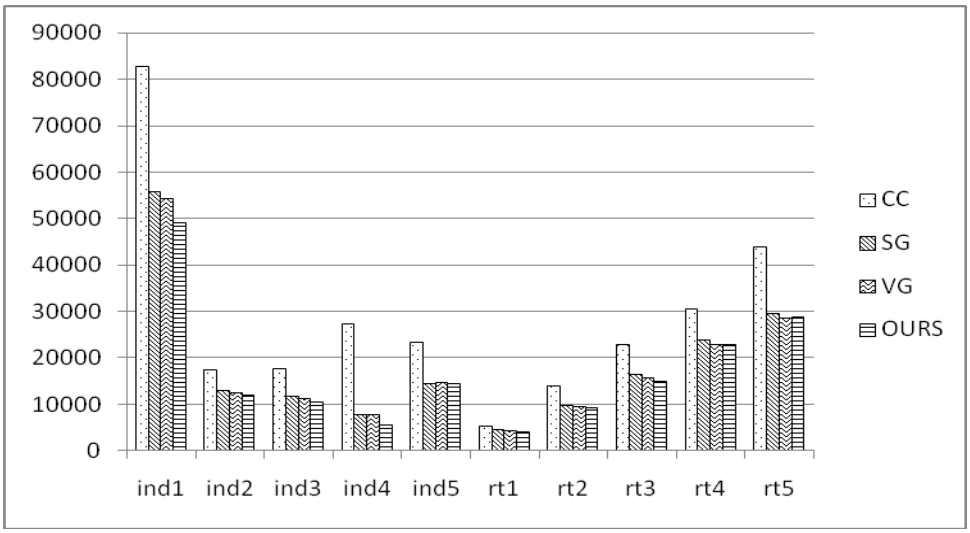


图 4-15 四种算法在通孔代价 C_v=5 时在布线总代价方面的对比柱状图

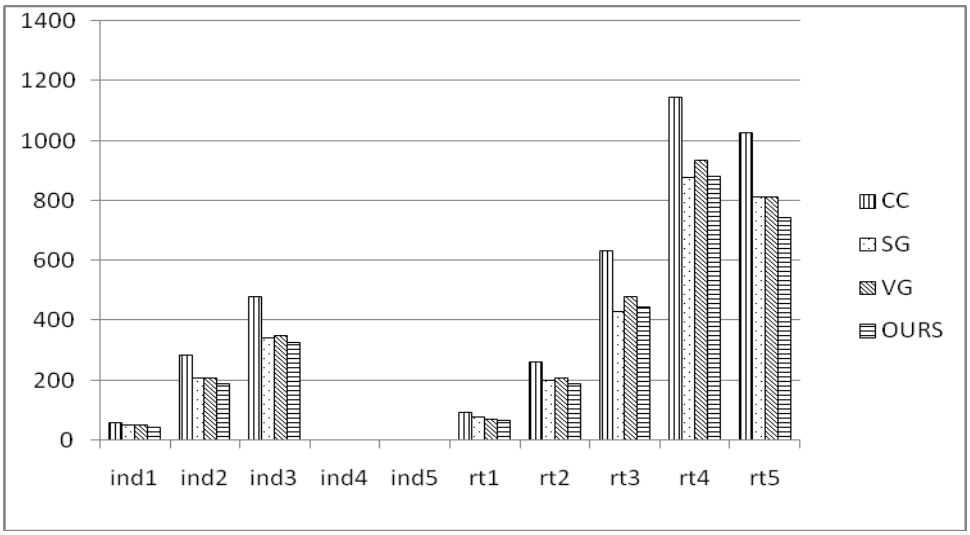


图 4-16 四种算法在通孔代价 C_v=5 时在通孔数方面的对比柱状图

4.6.4 与两种多层绕障 X 结构 Steiner 最小树算法的比较

表 4-8 是本章算法与文献[134]算法(X1)在布线总代价方面的对比情况，其中‘*’代表文献[134]算法在该组基准测试电路上得不到有效解。另外，类似文献[134]的思想，本章也将单层绕障 X 结构 Steiner 最小树构造方法[118]分别应用到多层布线的每一层绕障 X 结构 Steiner 最小树构造，再依次连接层与层的布线路径，继而构造相应的多层布线算法(X2)，从而完成最后的多层布线问题，相关实验结果也在表 4-8 的第 4 列和第 7 列中分别给出。X1 和 X2 算法均是基于单层结构的贪心方法。从表 4-8 可看出，本章基于多层结构的全局方法相对基于单层结构的贪

心方法，在布线总代价上均取得较大的改进率(分别取得平均 42.22%和 33.41%的减少率)。取得如此大改进率的原因，主要有以下两点。

表 4-7 通孔代价 C_v=5 时与三种 ML-OARSMT 算法在通孔数方面的对比

实例	$n/m/N_l$	#via				(X-OURS)/X (%)		
		CC	SG	VG	OURS	X=CC	X=SG	X=VG
ind1	50/6/5	59	49	49	43	27.12	12.24	12.24
ind2	200/85/6	283	208	206	188	33.57	9.62	8.74
ind3	250/13/10	480	343	348	326	32.08	4.96	6.32
ind4	500/100/5	0	0	0	0	0.00	0.00	0.00
ind5	1000/20/5	0	0	0	0	0.00	0.00	0.00
rt1	25/10/10	91	76	70	66	27.47	13.16	5.71
rt2	100/20/10	263	200	209	187	28.90	6.50	10.53
rt3	250/50/10	633	429	478	445	29.70	-3.73	6.90
rt4	500/50/10	1144	879	936	883	22.81	-0.46	5.66
rt5	1000/100/5	1025	814	814	744	27.41	8.60	8.60
AVG						22.91	5.09	6.47

首先， X1 算法是将多层布线问题转换成多个单层布线问题，分别对布线问题中每一层的引脚集合构建其绕障 Steiner 最小树，之后再为两两布线层寻找一条最小的连接路径。这样导致 X1 算法未能从多层结构的全局角度考虑布线总代价的优化，从而使得算法缺乏搜索较优解的能力甚至搜索不到有效解。同样，X2 算法亦是如此。而在单层绕障 Steiner 最小树构造中两个引脚之间的最短距离只需考虑引脚和障碍物的端点即可构造出引脚之间的最小路径，而在多层绕障 Steiner 最小树问题的大多数情况不行。根据性质 4.1 的分析过程和结论，若跟本章算法一样从多层结构的全局角度考虑多层绕障 Steiner 最小树问题，除了考虑引脚和障碍物的端点，还需考虑图 4-6 中 *P3* 在其上一层的映射点 *P3'*，从而构建 *P1*，*P2*，*P3* 之间的最小路径，从而相对于 X1 和 X2 这一类基于单层结构的贪心方法可带来布线总代价的优化率为[0%，100%)。这是本章算法相对上述基于单层结构的贪心方法(X1 和 X2)能够取得如此大改进率的原因之一。

其次，X1 和 X2 算法更侧重于通孔数的优化，也导致其布线总代价的优化能力受限。根据性质 4.1 和性质 4.2 的分析过程和结论，若跟本章算法一样从多层结构的全局角度考虑多层绕障 Steiner 最小树问题的求解，通过在图 4-6 和图 4-7 分别引入通孔 $\overline{P3P3'}$ 和通孔 $\overline{P2P4}$，可带来布线总代价的优化率分别为[0%，100%)和[0%，50%)。这是本章相对基于单层结构的贪心方法(X1 和 X2)能够取得如此大改进率的第二个原因。

因此，X1 和 X2 算法并没有从多层结构的全局角度考虑多层绕障 X 结构 Steiner 最小树问题的求解。而本章算法的设计是从多层结构的全局角度考虑问题，更全面地搜索问题的解空间和更有效地利用通孔的优势，具有更好的全局性，从而相对 X1 和 X2 算法带来较为明显的布线总代价优化程度。

表 4-8 与两种 ML-OAXSMT 算法在布线总代价方面的对比

实例	$n/m/N_l$	布线总代价(#via)			(X-OURS)/X (%)	
		X1	X2	OURS	X=X1	X=X2
ind1	50/6/5	96780(4)	61392(4)	48996(43)	49.37	20.19
ind2	200/85/6	*	17569(5)	11652(188)	*	33.68
ind3	250/13/10	25569(9)	17700(9)	9654(326)	62.24	45.46
ind4	500/100/5	73588(0)	55154(0)	54965(0)	25.31	0.34
ind5	1000/20/5	14625950(0)	14368566(0)	14356523(0)	1.84	0.08
rt1	25/10/10	8206(9)	6488(9)	3855(66)	53.03	40.58
rt2	100/20/10	*	17783(9)	8774(187)	*	50.66
rt3	250/50/10	*	32876(9)	14084(445)	*	57.16
rt4	500/50/10	54872(9)	34142(9)	21070(883)	61.60	38.29
rt5	1000/100/5	59238(4)	52326(4)	27410(744)	42.15	47.62
AVG					42.22	33.41

4.6.5 OAPD-XST 问题中不同模型的比较

表 4-9 和表 4-10 给出了关于 OAPD-XST 问题中不同定向边约束模型所对应的通孔数、违反约束数(称为 VO)及布线总代价(称为 TC)等指标情况。因为模型 *a* 的循环周期是五层，而其他模型的循环周期是两层，所以本章选取布线层数 N_l=10 的基准电路作为 4.6.5 小节中实验的测试电路。从图 4-17 可看出模型 *c* 在违反约束数与布线总代价两个指标是明显优于模型 *a*，*b*，*d*，而所有模型在通孔数指标方面拥有类似的结果。如图 4-17 所示，水平坐标上值为 1，2，3，4 分别表示模型 *a*，*b*，*c*，*d*。在表 4-11 中，本章选取较优模型 *c* 作为对比模型，将其他模型与模型 *c* 进行对比。从中可看出，模型 *c* 的通孔数略少于模型 *a* 和模型 *b*(分别减少 0.85%和 1.87%)，而略多于模型 *d*(2.73%)。然而，对于模型 *c*，在 VO 指标上，相对模型 *a*，*b*，*d* 分别取得 45.70%、20.75%及 21.00%的减少率。另外，在布线总代价的指标上，模型 *c* 相对模型 *a*，*b*，*d* 分别取得 26.61%、13.55%及 13.56%的减少率。综上所述，将这四种模型应用于 OAPD-XST 问题中，其优先顺序为模型 *c*，*b*，*d*，*a*。其中，模型 *c* 是优于其他模型，而模型 *b* 和 *d* 的优劣程度区别不大。

表 4-9 OAPD-XST 问题中不同定向边约束模型的对比(1)

实例	$n/m/N_l$	*a*			*b*		
		#via	VO	TC	#via	VO	TC
ind3	250/13/10	262	315	22636	248	253	20209
rt1	25/10/10	22	33	10195	22	19	8694
rt2	100/20/10	131	140	21915	142	93	18369
rt3	250/50/10	309	351	37639	332	240	31736
rt4	500/50/10	733	712	52503	701	480	43000
AVG		291	310	28978	289	217	24402

表 4-10 OAPD-XST 问题中不同定向边约束模型的对比(2)

实例	$n/m/N_l$	*c*			*d*		
		#via	VO	TC	#via	VO	TC
ind3	250/13/10	243	196	18118	248	233	19568
rt1	25/10/10	23	11	6057	22	22	8865
rt2	100/20/10	130	76	16844	123	89	18527
rt3	250/50/10	315	216	29015	309	239	31892
rt4	500/50/10	713	428	38606	689	503	43198
AVG		285	185	21728	278	217	24410

表 4-11 OAPD-XST 问题中不同定向边约束模型的对比(3)

实例	(*a*-*c*)/*a* (%)			(*b*-*c*)/*b* (%)			(*d*-*c*)/*d* (%)		
	#via	VO	TC	#via	VO	TC	#via	VO	TC
ind3	7.25	37.78	19.96	2.02	22.53	10.35	2.02	15.88	7.41
rt1	-4.55	66.67	40.59	-4.55	42.11	30.33	-4.55	50.00	31.68
rt2	0.76	45.71	23.14	8.45	18.28	8.30	-5.69	14.61	9.08
rt3	-1.94	38.46	22.91	5.12	10.00	8.57	-1.94	9.62	9.02
rt4	2.73	39.89	26.47	-1.71	10.83	10.22	-3.48	14.91	10.63
AVG	0.85	45.70	26.61	1.87	20.75	13.55	-2.73	21.00	13.56

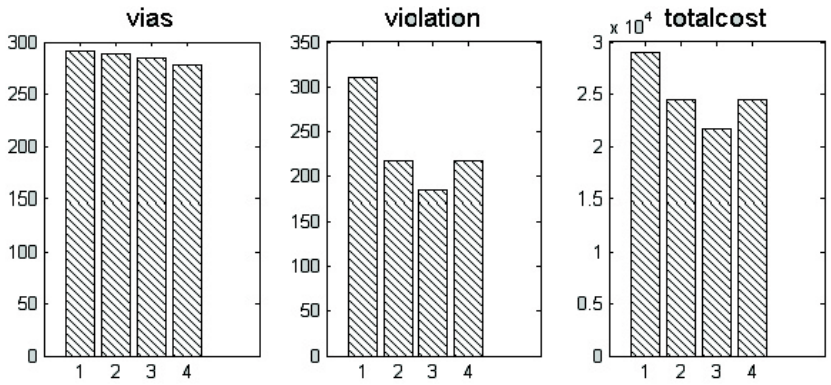


图 4-17 四种不同定向边约束模型在通孔数、违反约束数及布线总代价的对比

4.7 小结

针对多层绕障 X 结构 Steiner 最小树问题，本章以最小化布线总代价为目标，并同时考虑通孔数的优化，提出了一种基于离散 PSO 算法和惩罚机制的多层绕障 X 结构 Steiner 最小树构建算法。算法在粒子的更新公式中引入边变换操作以加强布线树的拓扑变化能力，进一步增加布线树的绕障能力和线长优化能力，并设计适合 X 结构 Steiner 最小树和多层结构的多层边点对编码策略。本章通过提出合适的惩罚机制用以融入粒子的适应度函数设计，更有效地考虑绕障情况和通孔数，并从基于多层结构的全局角度考虑多层布线问题的求解空间，从而达到更有力优化布线总代价的目的。同时，引入预处理策略大大减少了绕障情况的重复判断次数和通孔数的重复计算次数。实验结果表明算法相关策略的有效性以及本章算法相对同类算法能取得可观的布线总代价减少率，是在当前同类算法中具有最强的布线总代价优化能力。

此外，由于 OAPD-XST 问题模型更符合 X 结构布线的实际工艺，所以基于 ML- OAXSMT-PSO 算法，本章设计了一种 OAPD-XST 问题的构造算法，进而分析在 OAPD-XST 问题中不同定向边约束模型的优劣性。本章工作是第一次开展 OAPD-XST 问题的研究，相关研究工作可为基于非曼哈顿结构的芯片设计提供一定的理论支持。

第五章 基于 ILP 和划分策略的 X 结构总体布线算法

5.1 引言

随着半导体和互连线的大小不断缩小，互连线严重影响很多关键的设计指标，因此，作为组织互连线的实际走线位置的布线阶段在现今 VLSI 物理设计中变得尤为重要。复杂的布线过程是由总体布线和详细布线两阶段构成的。在总体布线中，每个线网的走线被分配到各个通道布线区域中，每个通道区域的布线问题得到明确定义。而详细布线则给出了每个线网在通道区域的具体位置。因此，总体布线的质量严重影响了详细布线的成功率，进而对整个芯片的性能起到决定性的作用。

总体布线是超大规模集成电路物理设计中极为重要的一部分，因而学者们提出了很多有效的算法和总体布线技术，主要可分为串行算法和并行算法两种，特别是以串行算法为代表的总体布线方法能够处理大规模的问题[4-6,9,10,17,18,34,137]。但串行算法对线网的布线顺序或是布线代价的定义具有严重的依赖性，这一潜在特性严重影响了总体布线的质量。而以整数线性规划为代表的并行算法能够减少布线结果对线网顺序的依赖性，取得质量较好的总体布线方案[11,15,138-143]。这些以整数线性规划为代表的部分方法在求解整数线性规划模型是先将其松弛为线性规划模型以减少求解规划模型的时间复杂度，再利用随机取整的方法将线性解转换为非线性解，这样导致随机取整的过程所取得的布线方案可能严重偏离真正的解方案。

大部分总体布线算法都是以曼哈顿结构为模型基础开展相关工作，而基于曼哈顿结构的优化策略在进行互连线线长优化时，由于绕线方向只能为水平和垂直，其优化能力受限。因此，研究人员开始尝试以非曼哈顿结构为基础模型进行布线，实现芯片整体性能的优化。目前关于 X 结构布线的研究主要包括两方面：(1)X 结构布线树的构造[100,144-147]和 X 结构总体布线算法设计[39,52-58]。随着 X 结构的引入，布线算法设计变得更为的复杂。然而，一些关于 X 结构布线的研究[39,52-58]主要集中在线长优化，这些算法是基于早期的 Steiner 树构造方法，从而导致了这些 X 结构布线算法相对于曼哈顿结构布线算法的优化效果不明显，甚至部分工作的布线结果相对曼哈顿结构的布线结果在线长优化能力方面显得更差。

为此，本章提出了一种 X 结构总体布线器(XGRouter)，主要基于整数线性规划模型、划分策略以及粒子群优化方法。本章提出的整数线性规划模型同时考虑

到线长优化和拥挤度均衡。而 XGRouter 的性能基本不受基准测试电路的属性影响，可产生较高质量的解，具有良好的鲁棒性。

本章工作的贡献如下。

(1) 本章提出的 ILP 模型(O-ILP 模型)能够较好地解决串行布线器存在的布线结果对线网顺序的高度依赖问题，并能得到拥挤度相对均匀的布线解，且相对传统的 ILP 模型[139]具有更好的扩展性。

(2) XGRouter 是第一次将并行算法应用于 X 结构总体布线问题的求解，并取得较高质量的布线解。

(3) 部分并行算法在求解曼哈顿结构总体布线问题时，是将总体布线问题转换为 ILP 模型，并将其松弛为线性规划问题，并使用随机取整策略求解，从而导致最终的解方案产生偏差。为此，本章算法设计了重定义的离散 PSO 算法用以求解 O-ILP 模型，以更好地预测和协调区域内所有线网的布线情况。

(4) 本章算法采用划分策略以减少所建立 ILP 模型的规模，其中 XGRouter 首先从最拥挤区域开始布线，并逐步扩展布线区域直至覆盖整个芯片。该划分策略的引入使得 PSO 和 O-ILP 模型都适用于总体布线问题的求解。从仿真实验结果可看出 XGRouter 可获得相对其他总体布线器而言质量较高的解方案。

5.2 相关研究工作

布线通道拥挤度的不可预见性和对线网顺序的依赖性是目前总体布线算法面临的两大问题。为了有效地解决这两个问题，串行算法和并行算法相继提出。其中，串行算法采用迷宫布线进行初始布线，再采用基于不同的代价函数设计相应的拆线重布策略，对不满足约束的线网重布，以进一步优化布线解的质量。这些代价函数包括基于离散拉格朗日乘数[6]以及基于不同协商机制的拥挤度函数[5,8,9, 13,17]。随着现今芯片设计变得更为复杂，设计一种好的代价函数变得越来越困难。因此，对于串行算法，拥挤度预测变得非常困难且线网的布线顺序对最后的总体布线质量影响极大。

部分学者针对总体布线问题建立 ILP 模型，并使用相应的线性规划方法或流算法对相应的问题进行求解，这是一类典型的并行算法。Vannelli 在文献[141]中提出一种线性规划算法。首先，为每个线网寻找一些可能的连接树并计算相应每一棵候选树的布线代价，再建立带通道容量约束的线性方程，最后，求解相应的线性规划模型以获得最优解。但该算法的复杂度非常高，因此，Behjat 等针对总体布线问题提出了一种整数线性规划模型，并采用线性松弛的方法来降低模型的计算复杂度[142,143]。不同于线性规划模型，Albrecht 是采取多商品流算法求解总体布线问题[139]。以上并行算法克服了串行算法所得解方案的质量受线网布线顺

序影响的问题，但只适合于一般规模的布线问题。为此，Cho 等提出一种新型 ILP 模型(M-ILP)，该模型相对于传统的 ILP 模型(T-ILP)[139]显得更为快速且扩展性良好，并采用划分策略进一步改善运行时间[11,138]。Wu 等也采用了划分思想，将芯片区域分解为多个子区域以求解多层布线问题[15]。但是现有这些并行算法均基于曼哈顿结构，且部分算法采用线性松弛方法求解 ILP 模型，存在偏差问题。

随着集成电路规模的集成度不断增加以及制造工艺的发展，VLSI 物理设计的互连结构已经发生变化。以互连线为中心及其非曼哈顿结构的提出，使得非曼哈顿结构布线算法面临新的机遇和挑战，已成为 VLSI 物理设计的研究热点。

在总体布线图上的 X 结构 Steiner 树生成算法是所有 X 结构总体布线的算法基础[3]。为了能够研究 X 结构总体布线，首先需要建立 X 结构布线树[42,43,45,46]，X 结构布线树的构建是 X 结构总体布线的的基础问题[39]。Koh 和 Madden 描述了多种 VLSI 互连结构，并针对非曼哈顿结构 Steiner 树的构建问题，提出了相应的启发式算法[50]。针对图形 Steiner 树问题，文献[144]提出一类新颖的启发式算法，包括用于求解曼哈顿结构布线中的绕障矩形 Steiner 树问题与 X 结构布线中的绕障 X 结构 Steiner 树问题。Hung 等基于生成图的概念，提出了一种多步骤的绕障 X 结构布线树构建算法[145]。文献[100]提出一种基于离散 PSO 的 X 结构 Steiner 最小树构建算法，以优化线长为目标。Liu 等进一步考虑时延目标的优化，提出一种时延驱动 X 结构 Steiner 最小树构建算法并考虑到拐弯数的优化工作[119]。

针对 X 结构布线算法设计，一种多级布线框架在文献[52,53]中提出，采用先自底向上粗化再自顶向下细化的两阶段策略。Hu 等采用随机子树生长方法在迭代数次后构建相应的布线树，并最终布通整个芯片[57]。Cao 等建立了 X 结构的总体布线问题模型，并提出了动态资源调度方法以减少潜在通孔数[58]。然而这些关于 X 结构布线算法[52,53,57,58]是基于早期的 Steiner 树构建算法，导致这些 X 结构布线算法的优化效果不明显，甚至部分结果在线长指标上劣于曼哈顿结构布线算法。而且以上 X 结构布线算法均属于串行算法，非常依赖于线网的布线顺序或布线代价函数的定义。

本章所开展的工作是第一次使用并行算法处理 X 结构总体布线问题。为了能够使拥挤度更为均衡，提出了一种新型的 ILP 模型，其优化目标包括线长最小化和拥挤度均衡。本章采用划分策略以减少针对总体布线问题所构建 ILP 模型的复杂度。

5.3 基础知识

5.3.1 问题模型

本章工作所使用符号的标记情况如表 5-1 所示。

给定一个总体布线图 G，包含一个引脚集合 V，边集合 E，一个线网集合 $N=\{N_1, N_2, \ldots, N_n\}$，其中线网 N_k，$1 \le k \le n$，包含一系列引脚集合，则以线长为优化目标的总体布线问题可描述如下。

$$\text{Minimize} \quad \sum_{k=1}^{n} L(T_k)$$
$$\text{S.T.} \quad U(e_i) \le C(e_i), \ \forall e_i \in E$$

其中，$L(T_k)$表示布线树 T_k 的长度，布线边的最大容量 $C(e_i)$表示该边 e_i 能够允许的最大走线数。$U(e_i)$表示总体布线问题中经过边 e_i 的次数，可表示为

$$U(e_i) = \sum_{k=1}^{n} x_{ik}$$

其中，n 是布线树的总数，X_{ik} 是一个二进制数。当布线树 T_k 的走线经过边 e_i，则 X_{ik} 的值为 1，否则为 0。

表 5-1 本章工作所使用符号的标记情况

符号标记	在本章所代表的意思
G_i	总体布线单元 i
N_k	总体布线问题中的一个线网
T_j	线网对应的一棵布线树
e_i	总体布线图中的一条边
$C(e_i)$	边 e_i 的布线容量
$U(e_i)$	总体布线问题中经过边 e_i 的次数
X_{ik}	记录布线树 T_k 是否经过边 e_i

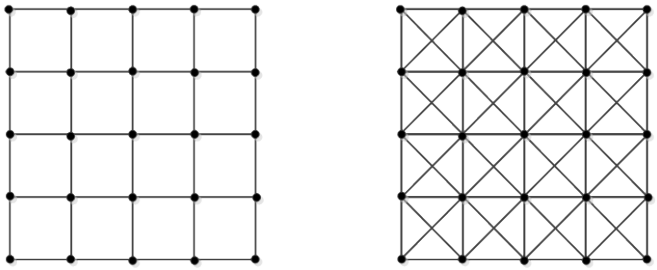
定义 5.1 (网表)[3] $N=\{N_1, N_2, \ldots\}$，其中每个线网 N_k 是一系列引脚的集合，属于同一个线网的引脚集合将由布线工具把它们连接一起。网表是提供电路的互连信息。

定义 5.2 (布线容量)[3] 在布线区域内所能容下最大的走线数，它是由设计规则、布线区域的大小和线网的宽度所决定的。

定义 5.3 (溢出边) 当一个边的布线资源需求量大于可用的布线资源，则该边称为溢出边。

图 5-1 分别给出了基于曼哈顿结构和 X 结构的总体布线图。从中可看出由于 X 结构布线设计允许相对更多的走线方向，所以基于 X 结构的 GRG 图相对于基

于曼哈顿结构的 GRG 图更为复杂。



(a) 曼哈顿结构设计　　(b) X 结构设计

图 5-1 总体布线图

5.3.2 网格的 X 结构距离

X 结构网格是由水平线、垂直线、45°的对角线、135°的对角线交叉组成的。两引脚 P_i 和 P_j 之间在 X 结构网格下八角形距离用边 $e(P_i, P_j)$的长度表示，令 P_i=(X_i,Y_i)和 P_j=(X_j,Y_j)分别表示平面上两引脚的坐标。

引理 5.1 如果(X_i == X_j)且(Y_i!= Y_j)，那么两引脚 P_i 和 P_j 之间的距离 $e(P_i, P_j)$ =$|y_i - y_j|$。两引脚沿垂直方向走线。

引理 5.2 如果(X_i != X_j)且(Y_i== Y_j)，那么两引脚 P_i 和 P_j 之间的距离 $e(P_i, P_j)$ =$|x_i - x_j|$。两引脚沿水平方向走线。

引理 5.3 如果(X_i != X_j)和(Y_i!= Y_j)并且(X_i - X_j)= (Y_i -Y_j)与(X_i - X_j)= - (Y_i -Y_j)之一成立，那么两引脚 P_i 和 P_j 之间的距离 $e(P_i, P_j) = \sqrt{2}\times|x_i - x_j|$。两引脚位于网格的对角线上，因此两引脚的走线方向为 45°对角线或 135°对角线。

引理 5.4 如果(X_i != X_j)和(Y_i!= Y_j)并且$|x_i - x_j| < |y_i - y_j|$，那么两引脚 P_i 和 P_j 之间的距离 $e(P_i, P_j) = (\sqrt{2}-1)\times|x_i - x_j| + |y_i - y_j|$。两引脚的走线方向包括对角线方向和垂直方向。

引理 5.5 如果(X_i != X_j)和(Y_i!= Y_j)并且$|x_i - x_j| > |y_i - y_j|$，那么两引脚 P_i 和 P_j 之间的距离 $e(P_i, P_j) = (\sqrt{2}-1)\times|y_i - y_j| + |x_i - x_j|$。两引脚的走线方向包括对角线方向和水平方向。

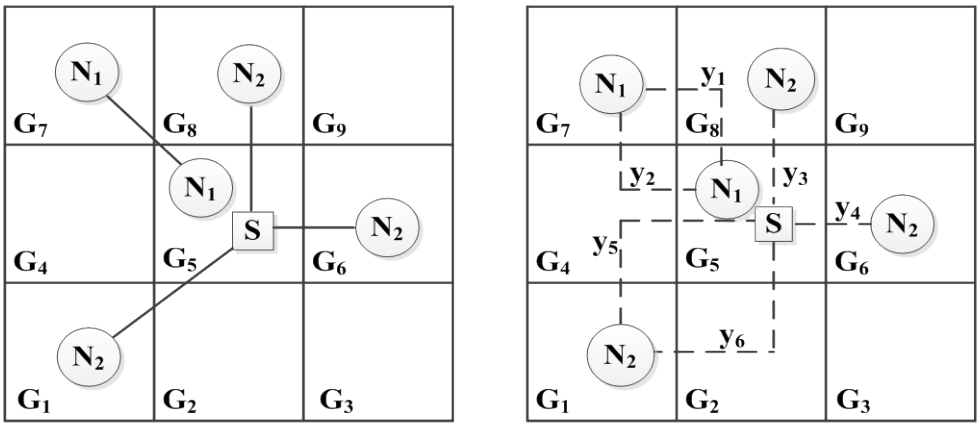
5.4 ILP 模型

很多并行算法常用 ILP 模型求解总体布线问题，并可取得较为不错的布线方案。通常，基于 ILP 模型的方法，首先，寻找所有可能的布线树作为候选解，并计算每个布线树的代价，建立相应的带容量约束的线性方程组，最后，求解该线性方程组以获得最终的总体布线方案。为了更好地求解总体布线问题，一些学者提出了不同类型的 ILP 模型，它们具有各自的优缺点。本小节将详细介绍两种 ILP

模型并提出本章用于求解 X 结构总体布线问题的新 ILP 模型。

5.4.1 T-ILP 模型

T-ILP 模型是在文献[139]中提出的，其优化目标是最小化最大拥挤度。图 5-2(a)给出了准备布线的两个线网 N_1 和 N_2 以及最大通道容量为 2。本章通过 Steiner 最小树算法将多端线网 N_2 分解为 3 个两端线网(SG_8，SG_6 和 SG_1)，并为所有两端线网建立相应的布线候选解，如图 5-2(b)所示。图 5-2(c)给出基于传统 ILP 模型得到的布线方案。针对图 5-2(b)所示的例子，建立相应的 T-ILP 模型，具体的模型如图 5-3 所示。图 5-4 给出了 T-ILP 模型的一般形式。



(a)分布在布线图上的两个待布线网　(b)N_2的分解及为每个两端线网构造候选解

(c)第一个布线方案　(d) 第二个布线方案

图 5-2 一个总体布线问题的例子

从图 5-4 可看出，T-ILP 的优化目标是最小化最大拥挤度 C_{max} 并且需要满足一系列的约束。其中，第一个约束集要求每个线网从其候选解集合中只能选取一个作为其布线方案，y_j 代表一个候选解，k 表示线网的总数。第二个约束集确保 GRG 中每条边的走线数不超过最大拥挤度 C_{max}，其中，X_{ij} 表示一个二进制值，

如果候选解 y_j 的走线经过 GRG 中的边 e_i，则 X_{ij} 的值为 1，否则为 0。p 表示 GRG 中所有边的总数目。第三个约束集限制 y_j 是二进制值，其中，t 表示布线候选解的总个数。T-ILP 模型的优势有以下两点：

(1) T-ILP 模型的目标是最小化最大拥挤度，从而可得到更为均匀的布线方案。

(2) T-ILP 模型确保每个线网都有一个候选解，因此，在没有溢出边产生的情况下可完成布线而不需要额外的步骤。

然而 T-ILP 模型存在的不足是：当最大拥挤度的值大于布线边的最大通道容量值，溢出边的个数将陡增，导致布线结果的不可布线性，即未能完成布线工作。因此，需要额外步骤消除溢出边的情况。

$$
\begin{aligned}
&\text{Minimize } C_{\max} \\
&\text{S.T.} \quad y_1 + y_2 = 1 \\
&\qquad y_3 = 1,\ y_4 = 1 \\
&\qquad y_5 + y_6 = 1 \\
&\qquad y_1 + y_3 \le C_{\max} \\
&\qquad y_2 + y_5 \le C_{\max} \\
&\qquad y_1 \le C_{\max},\ y_2 \le C_{\max},\ y_4 \le C_{\max} \\
&\qquad y_5 \le C_{\max},\ y_6 \le C_{\max} \\
&\qquad y_1, y_2, y_3, y_4, y_5, y_6 \in \{0, 1\}
\end{aligned}
$$

图 5-3 在图 5-2 所示布线问题中 T-ILP 模型的具体形式

$$
\begin{aligned}
&\text{Minimize } C_{\max} \\
&\text{S.T.} \quad \sum_{y_j \in N_k} y_j = 1, \quad k = 1, 2, \cdots, n \\
&\qquad \sum_{j=1}^{1} X_{ij} y_j \le C_{\max}, \quad i = 1, 2, \cdots, p \\
&\qquad y_j \in \{0, 1\}, \quad j = 1, 2, \cdots, t
\end{aligned}
$$

图 5-4 T-ILP 模型的一般形式

5.4.2 M-ILP 模型

M-ILP 模型的目标是最大化布通线网的权重累加和，其中，权重是指每个线网的线长[138]。为如图 5-2(b)所示的总体布线实例所构建的具体 M-ILP 模型在图 5-5 给出。由于如图 5-2(c)和(d)所示的两个布线方案对应的 M-ILP 模型中布通线网的权重累加和都是等于 6，所以基于 M-ILP 模型的布线结果可能包括这两种布线方案。图 5-6 给出 M-ILP 模型的一般化形式，该模型也需要满足一系列的约束条件，但其中只有第三个约束集是跟 T-ILP 模型一样。第一个约束集是将方程松弛为线性方程，第二个约束集确保每条边的最大走线数不超过最大通道布线容量

$C(e_i)$，而并非 T-ILP 模型中的最大拥挤度 C_{max}。M-ILP 模型的优势可归纳如下。

(1) 每个候选布线解都有一个权重因子，因此，其他优化目标可方便融入目标函数。

(2) 基于 M-ILP 模型获得的布线解不会包括溢出边，因为第二个约束集严格限制 GRG 中每条边的走线数不超过该边对应的最大通道容量。

$$\begin{aligned}
&\text{Maximize} \quad 2y_1 + 2y_2 + y_3 + y_4 + 2y_5 + 2y_6 \\
&\text{S.T.} \quad y_1 + y_2 \le 1 \\
&\qquad y_3 \le 1,\ y_4 \le 1 \\
&\qquad y_5 + y_6 \le 1 \\
&\qquad y_1 + y_3 \le 2 \\
&\qquad y_2 + y_5 \le 2 \\
&\qquad y_1 \le 2,\ y_2 \le 2,\ y_4 \le 2 \\
&\qquad y_5 \le 2,\ y_6 \le 2 \\
&\qquad y_1, y_2, y_3, y_4, y_5, y_6 \in \{0, 1\}
\end{aligned}$$

图 5-5 在图 5-2 所示布线问题中 M-ILP 模型的具体形式

$$\begin{aligned}
&\text{Maximize} \quad \sum_{j=1}^{t} y_j \times Wl_j \\
&\text{S.T.} \quad \sum_{y_j \in N_k} y_j \le 1, \quad k = 1, 2, \cdots, n \\
&\qquad \sum_{j=1}^{1} X_{ij} y_j \le C(e_i), \quad i = 1, 2, \cdots, p \\
&\qquad y_j \in \{0, 1\}, \quad j = 1, 2, \cdots, t
\end{aligned}$$

图 5-6 M-ILP 模型的一般形式

但 M-ILP 模型也存在一些不足：(1) M-ILP 模型由于未考虑拥挤均衡，可能出现拥挤不均衡的布线方案，导致部分布线区域的走线较为密集而部分区域较为稀疏。比如，M-ILP 模型可能得到一个如图 5-2(d)所示较为不均匀的总体布线方案。(2) M-ILP 模型为了得到严格不违反最大通道容量约束的总体布线方案，可能会放弃可能产生溢出边且权重较小的一些线网的布线。因此， M-ILP 模型可能不会完全布通所有线网，从而需要一些额外的步骤(例如迷宫布线算法[9,123])以完成最终的布线。

5.4.3 O-ILP 模型

本章所提出的 O-ILP 模型是为了最小化所有线网的总线长，同时考虑到拥挤度分布情况，从而构建更为均匀的布线结果。如图 5-2(b)所示的布线问题，O-ILP 模型可获得如图 5-2(c)所示更为均匀的布线方案。针对图 5-2(b)所示的布线问题，

O-ILP 模型的具体形式如图 5-7 所示。

定义 5.4 (拥挤度) e_i 的拥挤度是指布线边的实际走线个数 $U(e_i)$与最大通道容量 $C(e_i)$的比值，计算公式如下。

$$congestion(e_i)=\frac{U(e_i)}{C(e_i)}$$

O-ILP 模型的一般化形式如图 5-8 所示。*Std*()表示所有边的拥挤度集合对应的标准差。第一个约束集限制每个线网只能从候选布线方案中选取一个候选解。第二个约束集确保候选布线方案 y_j 的值是二进制值。另外，O-ILP 模型中 Wl_j 表示方案 y_j 的线长，β 是惩罚项，γ 表示违反约束的次数，α 是用户自定义参数。参数 β 和 α 的选择情况将在实验部分具体给出。

$$\begin{aligned}
&\text{Minimize}\quad \beta\times((2y_1+2y_2+y_3+y_4+2y_5+2y_6)+\\
&\alpha\times Std(\,y_1/2,y_2/2,y_3/2,y_4/2,y_5/2,y_6/2))\\
&\text{S.T.}\quad y_1+y_2=1\\
&\qquad y_3=1,\ y_4=1\\
&\qquad y_5+y_6=1\\
&\qquad y_1+y_3\le 2\\
&\qquad y_2+y_5\le 2\\
&\qquad y_1\le 2,\ y_2\le 2,\ y_4\le 2\\
&\qquad y_5\le 2,\ y_6\le 2\\
&\qquad y_1,y_2,y_3,y_4,y_5,y_6\in\{0,1\}
\end{aligned}$$

图 5-7 在图 5-2 所示布线问题中 O-ILP 模型的具体形式

$$\begin{aligned}
&\text{Minimize}\ \ \beta\times(\sum_{j=1}^{t}y_j\times Wl_j+\alpha\times Std(congestion(e_1),\\
&\qquad congestion(e_2),\cdots,congestion(e_p)))\\
&\text{S.T.}\ \ \sum_{y_j\in N_k}y_j=1,\ k=1,2,\cdots,n\\
&\qquad \beta=\begin{cases}1, & \text{if}\quad \sum_{j=1}^{1}X_{ij}y_j\le C(e_i)\quad \forall e,i=1,2,\cdots,p\\ 1.026^{\gamma}, & \text{else}\end{cases}\\
&\qquad y_j\in\{0,1\},\ j=1,2,\cdots,t
\end{aligned}$$

图 5-8 O-ILP 模型的一般形式

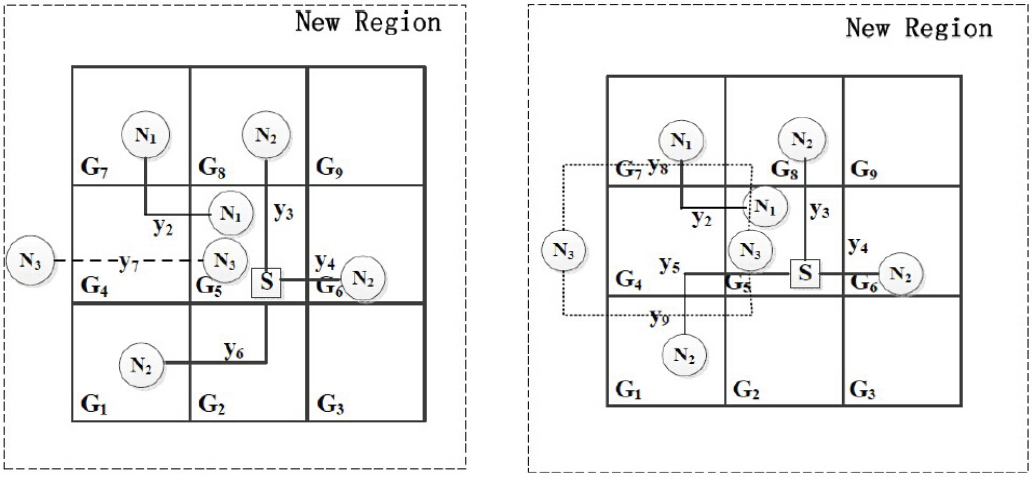
性质 5.1 本章算法基于 O-ILP 模型可获得相对未考虑拥挤均衡的方法更为均匀的布线结果，而且在一些情况可同时取得更小的线长值。

在样本大小一致的情况下，一般标准差越大，相应数据集的波动性也越大即

越不稳定。因此，本章采用所有边的拥挤度集合{*congestion*(e_i)}的标准差近似表示总体布线问题的拥挤均衡性。O-ILP 模型的优化目标包括线长最小化和拥挤均衡。

最小化集合{*congestion*(e_i)}的标准差可能使得在后续布线区域进行布线工作获得较小的线长指标。如图 5-9(a)和(b)所示的例子是分别来源于图 5-2(c)和(d)所示的布线方案完成后进一步扩展后新布线区域(New Region)。其中，虚线框标识新布线区域，线网 N_3 是在新布线区域中准备布线的线网。本章算法基于 O-ILP 模型进入新的布线区域前可获得如图 5-9(a)所示更为均匀的布线方案(拥挤度集合对应的标准差是 0.2611)，而算法基于 M-ILP 模型可能获得如图 5-9(b)所示的另外一个布线方案(拥挤度集合对应的标准差是 0.3279)。在进入新布线区域的布线过程，本章算法将布通线网 N_3。本章算法基于 O-ILP 模型可获得布线方案 y_7，对应所有线网的总线长为 8。而未考虑拥挤度均衡的算法只能获得布线方案 y_8 或 y_9，它们对应所有线网的总线长为 10。

综上所述，本章算法基于 O-ILP 模型可获得更为均匀的布线方案，甚至该方案拥有更小的线长，从而验证了性质 5.1 的有效性。为了进一步验证性质 5.1 的有效性，将在实验结果分析中设计相关实验，具体实验结果在表 5-3 中列出。



(a) 采用拥挤均衡策略的线长优化能力 (b) 未采用拥挤均衡策略的线长优化能力

图 5-9 拥挤均衡策略对线长优化的影响

O-ILP 模型的优势可归纳如下。

(1) O-ILP 模型考虑到拥挤度均衡，使得布线方案更为均匀，避免芯片中产生太多布线热点；

(2) 类似 M-ILP 模型，O-ILP 模型中每个候选解都对应一个权重因子，进而使得其他优化目标可容易融入至 O-ILP 模型的目标函数中；

(3) O-ILP 模型的构建是为了设计更为有效的 PSO 算法以求解相应的 ILP 模型。采用 PSO 算法求解 ILP 模型可解决由于随机取整方法可能带来的偏差问题。

对比前两个 ILP 模型，O-ILP 模型在一定程度上更为有效，但 O-ILP 模型也存在不能完全布通所有线网的情况，这是因为 XGRouter 尽可能在不产生溢出边的情况下搜索合适的布线方案。为此，在 XGRouter 的后期，本章设计一种基于新布线边代价函数的迷宫算法用以布通未布完的线网集合。

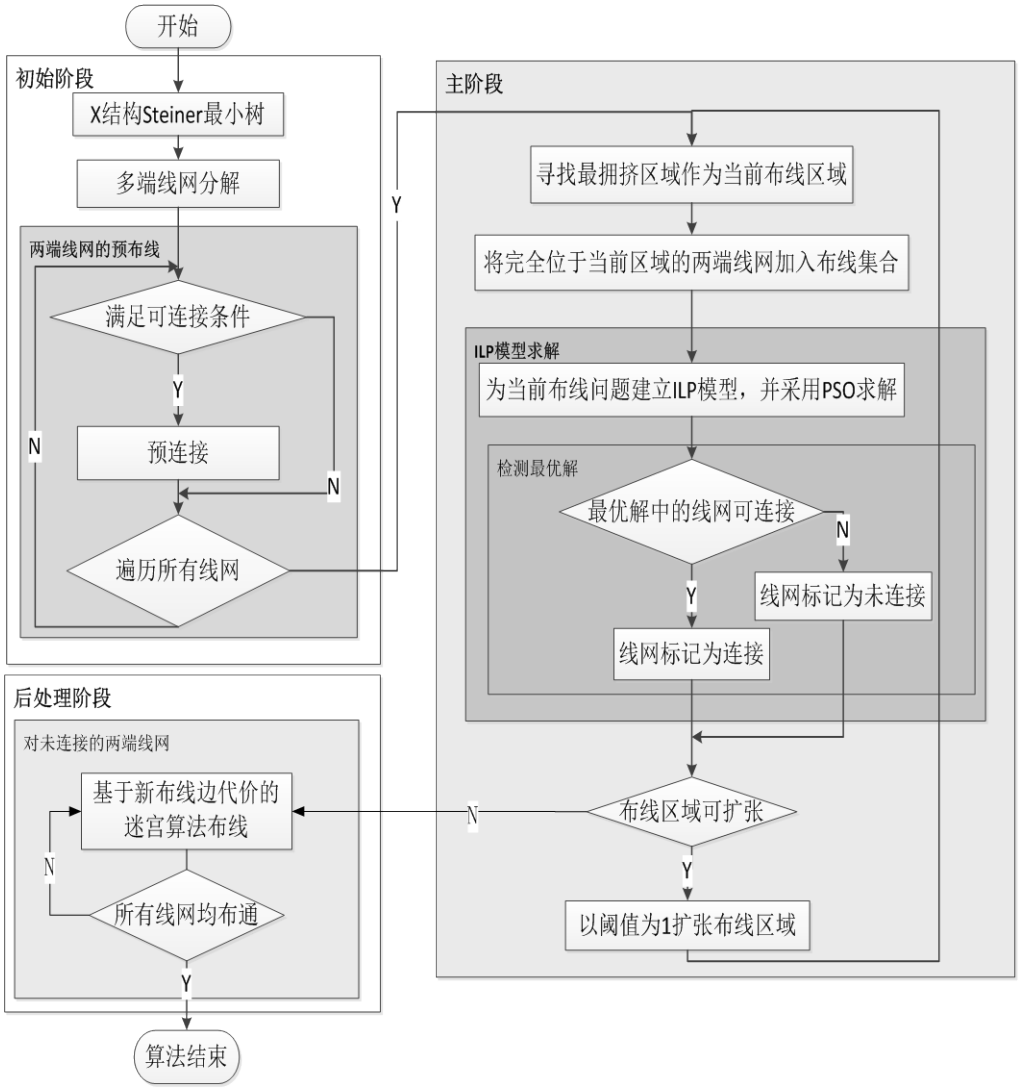


图 5-10 XGRouter 的算法流程

5.5 XGRouter 的详细设计过程

本章提出的 X 结构总体布线算法，XGRouter，主要包括三个阶段：初始布线阶段、主阶段及后处理阶段。XGRouter 的算法流程如图 5-10 所示，5.5.1 至 5.5.3 三个小节将分别详细介绍 XGRouter 的三个阶段。

5.5.1 初始布线阶段

在总体布线中，多端线网通常会被采用 Steiner 最小树或最小生成树分解为多个两端线网，而 Steiner 最小树方法相对最小生成树算法拥有更小的线长。因此，XGRouter 算法采用文献[100]提出的 X 结构 Steiner 最小树算法构造 XSMT 以对多端线网进行线网分解。之后计算每两端点所构成线段的斜率值，如果其斜率值为 0, -1, +1 和∞，则初始布线阶段将在保证不违反通道容量约束的前提下直接以 X 结构边的走线方式连接该两端线网。如果 GRG 边的实际走线数超过了最大通道容量，初始布线阶段放弃这类两端线网的连接。

初始布线阶段的流程如图 5-10 和算法 1 所示，其中，本章算法检查每个两端线网是否满足预布线的条件。如果满足，则直接连接两端线网并标识该线网为已连接，否则，继续检查下一个线网是否满足条件直至所有两端线网都被检查过。在初始布线阶段完成后，所有两端点线网都被逐一检查过。

算法 1 初始布线阶段算法

输入：待布线的线网集合 *N*
for 每个 *N* 中的线网 *i* **do**
if 线网 *i* 的引脚个数等于 2 **then**
if 线网 *i* 的两个引脚的相对位置满足预布线条件 **then**
布通线网 *i* 并标记线网 *i* 为已布线；
else
标记线网 *i* 为未布线并将其加入未布线线网集合；
end if
else
利用 XSMT 的构造算法将线网 *i* 分解为多个两端子线网；
for 线网 *i* 分解后的每个两端子线网 *j* **do**
if 子线网 *j* 的两引脚的相对位置满足预布线条件 **then**
布通子线网 *j* 并标记子线网 *j* 为已布线；
else
标记子线网 *j* 为未布线并将其加入未布线线网集合；
end if
end for
end if
end for

5.5.2 主阶段

在 XGRouter 的主阶段，本章采用划分策略以减少总体布线问题的规模，从而有助于 PSO 更为有效求解 O-ILP 模型。初始布线子区域的选择对最终总体布

线的质量有重大影响，而选择最拥挤区域作为初始布线子区域是一种不错的选择[138]。在 XGRouter 的初始布线阶段完成后，可获得芯片的近似拥挤情况，从初始布线结果中可以寻找芯片的最拥挤区域。本章算法选择规模为 2×2 的 GRG 网格大小作为主阶段的初始布线子区域，为该子区域内的布线问题建立相应 O-ILP 模型并采用改进的 PSO 算法进行求解。在布通每个子区域后，沿该区域的所有布线方向扩展一个 GRG 网格大小作为下一个布线子区域(例如区域从 2×2 扩展至 3×3)，并采用相同的方法进行布线，直至目前的布线子区域已包含整个芯片的布线区域。XGRouter 的主阶段算法如算法 2 所示，其中,本章用于求解 O-ILP 模型的改进 PSO 算法，其具体设计过程将在随后小节中介绍，改进 PSO 算法的算法流程如算法 3 所示。

算法 2 主阶段算法

输入：初始布线后尚未布线的线网集合 *N'*
寻找初始布线中最拥挤的布线子区域
while *true* **do**
  for *N'*中每个线网 *i* **do**
    if 线网 *i* 的两个引脚都在当前布线子区域内 **then**
      添加线网 *i* 至当前布线区域的线网集合 *CN*
    end if
  end for
  if 集合 *CN* 不为空 **then**
    为线网集合 *CN* 的布线问题建立 O-ILP 模型并调用算法 3 求解
  end if
  if 最优解方案的惩罚因子是 1 **then**
    布通 *CN* 中的所有线网并全部标记为已布通
  else
    标记其中造成惩罚的线网为未布线，而其余线网进行布线则标记为已布通
  end if
  以单位网格大小沿所有布线方向扩张当前布线区域作为下一次布线区域
  if 当前布线区域已包含整个芯片 **then**
    break；
  end if
end while

PSO 算法在被应用于离散问题中，已经有不同学者采用一些不同策略将连续型的 PSO 算法离散化。本章工作根据 O-ILP 问题的特点，重新定义了 PSO 的操作算子，从而提出一种求解 O-ILP 模型的有效 PSO 算法。5.5.2.1 小节设计一种适合于 O-ILP 模型的编码策略。5.5.2.2 小节介绍重新定义的 PSO 操作算子。5.5.2.3 小节描述了 PSO 算法适应度函数的设计过程。

算法 3 求解 O-ILP 的 PSO 算法

输入：当前布线区域的线网集合 *CN*

初始化 PSO 参数，包括种群大小、最大迭代次数等；
设置每个粒子为其初始的历史最优解并从中得到初始的种群全局最优解；
while *i*<最大迭代次数 **do**
  for 种群中的每个粒子 *p* **do**
    执行变异操作；
    执行粒子与其历史最优解的交叉操作；
    执行粒子与种群全局最优解的交叉操作；
    计算粒子的适应度函数值；
    if 粒子 *p* 的适应度值小于其历史最优值 **then**
      令当前粒子为其历史最优解；
    end if
    if 粒子 *p* 的适应度值小于种群全局最优**值** **then**
      令当前粒子为种群全局最优解；
    end if
  end for
end while

5.5.2.1 编码策略

由于 PSO 算法的鲁棒性，PSO 算法对编码策略的要求并不严格。尽管如此，一个好的编码策略不仅能改进算法的搜索效率，而且能够减少冗余的算法搜索空间，进而提高算法的性能。一般来说，编码策略的选择主要考虑以下三个准则：完整性、健全性及非冗余性[136]。

定义 5.5 (完整性) 问题搜索空间(即可行解空间)的每个点都能成为粒子编码空间中相应点的表现形式。

定义 5.6 (健全性) 编码空间中的每个粒子均能对应到问题搜索空间中潜在解方案。

定义 5.7 (非冗余性) 编码空间中的每个粒子能与问题搜索空间的潜在解方案一一对应。

对于一个编码策略要完全满足以上三个编码准则是非常困难的[136]。本章算法提出一种 0-1 整数编码策略表示一个布线方案，即 PSO 算法中的一个粒子。图 5-11 给出了如图 5-2(b)所示的布线问题所对应 PSO 算法的粒子编码，其中包括为四个两端线网所构造的六个布线候选解，因此该粒子的编码长度是 6。

布线候选解编号	1	2	3	4	5	6
粒子的编码	0	1	0	0	1	0

图 5-11 图 5-2(b)中布线实例的编码图

性质 5.2 针对总体布线问题所设计的 0-1 整数编码策略，能够满足完整性和非冗余性准则，但不能很好地满足健全性准则。

如图 5-2(d)所示的布线方案，其粒子的编码为‘011110’。假设最大通道容量为 1，那么如图 5-2(d)所示的 GRG 边 G_{45} 的(布线单元 G_4 和 G_5 之间的边)实际走线数为 2，大于最大通道容量，违反了 O-ILP 模型的约束，是一个不可行解，所以该编码不满足健全性准则。

为了更好地满足健全性准则，将在后续小节所讨论的交叉算子和变异算子融入检查策略，并且在适应度函数的设计中引入惩罚机制策略。这样可确保每个两端线网只有一个布线候选解会被选中，且对违反最大通道容量约束的粒子施加严厉的惩罚。

5.5.2.2 粒子的更新公式

针对总体布线问题所建立的 ILP 模型是一个离散问题，因此，为基本 PSO 构建的粒子更新公式不再适应于 O-ILP 模型的求解。为此，本章算法设计了交叉和变异算子用以构建离散形式的粒子更新公式[146,147]，并将检查策略融入这些算子以满足健全性原则，最终重新定义的粒子更新公式如下所示。

$$X_i^t = N_3(N_2(N_1(X_i^{t-1}, w), c_1), c_2) \tag{5-1}$$

其中，w 是惯性权重因子，c_1 和 c_2 是加速因子，N_1 表示变异操作，N_2 和 N_3 表示交叉操作。这里假定 r_1, r_2, r_3 都是在区间[0, 1)的随机数。

(1) 粒子的速度更新可表示如下。

$$W_i^t = N_1(X_i^{t-1}, w) = \begin{cases} M(X_i^{t-1}), & r_1 < w \\ X_i^{t-1}, & \text{others} \end{cases} \tag{5-2}$$

其中，w 表示粒子进行变异操作的概率。

如图 5-12 所示的变异操作算子的执行过程中，粒子编码中的一个随机位(mp1)被选中为变异操作位置，再根据公式(5-3)重新计算 mp1 的值(*val*)，其中符号‘%’表示取模运算符。在执行完变异(mutation)操作后，粒子的编码更新为‘101111’，其中，该粒子对应的布线方案违反 O-ILP 模型的第一个约束集，因为如图 5-2(b)所示同一个线网的两个候选解 y_5 和 y_6 同时被选中。因此，设计一个检查(check)策略用以检查布线方案是否满足 O-ILP 模型中第一个约束集，在执行完检查步骤后，粒子的编码变为如图 5-12 所示的‘101110’。

$$val = (val + 1)\%2 \tag{5-3}$$

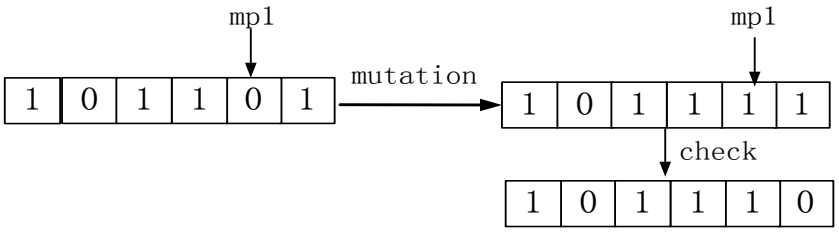


图 5-12 本章算法的变异算子

(2) 粒子的个体经验学习可表示如下。

$$S_i^t = N_2(W_i^t, c_1) = \begin{cases} C_p(W_i^t), & r_2 < c_1 \\ W_i^t, & \text{others} \end{cases} \tag{5-4}$$

其中，c_1 表示粒子与其历史最优方案进行交叉操作的概率。

(3) 粒子与种群其他粒子进行合作学习可表示如下。

$$X_i^t = N_3(S_i^t, c_2) = \begin{cases} C_p(S_i^t), & r_3 < c_2 \\ S_i^t, & \text{others} \end{cases} \tag{5-5}$$

其中，c_2 表示粒子与种群全局最优方案进行交叉操作的概率。

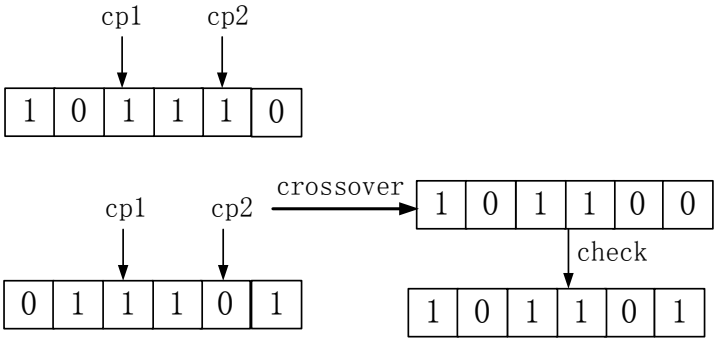


图 5-13 本章算法的交叉算子

如图 5-13 所示的交叉(crossover)算子，粒子编码中有两个随机位置(cp1 和 cp2，令 cp1< cp2)被选中，之后粒子与个体历史最优方案(全局最优方案)交换从 cp1 到 cp2 这一区间的编码值。类似变异操作算子，检查策略也被执行，从而获得交叉后粒子的最终编码为‘101101’。

检查策略的步骤如下所示。

步骤 1：选择发生变异操作或交叉操作的粒子编码位置。

步骤 2：对每个位置 y_j，其表示一个两端线网 N_k 的一个候选解，检查该线网 N_k 的其他候选解的编码值与 y_j 的值是否满足 O-ILP 模型的第一个约束集。若不满足则，改变其他候选解的编码值使其满足约束，否则保持编码值不变。

如图 5-12 所示，在完成变异操作后，粒子的编码变为‘101111’，当执行检查策略时，已完成变异的位置 y_5 是如图 5-2(b)所示的两端线网 SG_1 的一个候选解，

而 y_6 是 SG_1 的另外一个候选解，检查到 y_5+y_6 不等于 1，从而不满足 O-ILP 模型的第一个约束集，检查策略令 y_6 的值更新为 0 以满足约束。此时执行完如图 5-12 所示的检查策略后，粒子的编码更新为‘101110’。

5.5.2.3 适应度函数

总体布线问题的指标包括溢出数、线长、拥挤度等。其中，溢出数是总体布线第一重要的优化目标，严重影响芯片设计的可布性。因此，在 O-ILP 模型中的第二个约束集严格限制 GRG 中边的实际走线数不超过最大通道容量以避免溢出的产生。线长和拥挤度同样也是非常重要的 VLSI 布线指标[148]，因而在 XGRouter 的设计过程中，不但考虑线长最小化，还考虑拥挤度的均衡性。PSO 算法的最终适应度函数定义如下。

$$fitness = \beta \times (\sum_{j=1}^{t} y_j \times Wl_j + \alpha \times Std(congestion(e_1), congestion(e_2), \cdots, congestion(e_p))) \tag{5-6}$$

其中，β 表示惩罚项，α 表示拥挤度的权重，*Std*()表示当前布线区域中所有 GRG 边的拥挤度集合的标准差。如果最优布线方案不违反最大通道容量约束，β 的值为 1，否则 β 的值为 1.026^r，这里 r 是最优布线方案存在违反约束的次数。Wl_j 表示布线候选解的线长，它是根据 X 结构网格距离进行计算。每个两端线网所包含两个端点位置可分为如引理 5.1 至 5.5 所示的五种情况，并根据引理 5.1 至 5.5 的边距离式子计算相应的线长。

如图 5.2(c)和 5.2(d)所示的两个总体布线方案对应的拥挤度集矩阵 CM_1 和 CM_2 如下所示。

$$CM_1 = \begin{vmatrix} G_{12} & G_{23} & G_{45} & G_{56} & G_{78} & G_{89} \\ G_{14} & G_{25} & G_{36} & G_{47} & G_{58} & G_{69} \end{vmatrix} = \begin{vmatrix} 0.5 & 0.0 & 0.5 & 0.5 & 0.0 & 0.0 \\ 0.0 & 0.5 & 0.0 & 0.5 & 0.5 & 0.0 \end{vmatrix}$$

$$CM_2 = \begin{vmatrix} G_{12} & G_{23} & G_{45} & G_{56} & G_{78} & G_{89} \\ G_{14} & G_{25} & G_{36} & G_{47} & G_{58} & G_{69} \end{vmatrix} = \begin{vmatrix} 0.0 & 0.0 & 1.0 & 0.5 & 0.0 & 0.0 \\ 0.5 & 0.0 & 0.0 & 0.5 & 0.5 & 0.0 \end{vmatrix}$$

计算 CM_1 和 CM_2 相应的标准差 *Std*(CM_1)和 *Std*(CM_2)分别为 0.2611 和 0.3279。从图 5.2 可看出标准差 *Std*(CM_1)较小所对应的总体布线方案拥有更为均匀的拥挤分布。因此，在适应度函数的设计中，本章考虑拥挤度集矩阵的标准差优化以使得总体布线方案更为均匀。

5.5.3 后处理阶段

在 XGRouter 的主阶段完成后，总体布线方案中会经常存在一些未连接的两端线网，为此，后处理阶段中设计了基于新布线边代价的迷宫算法用以布通这些未连接线网，是最终布通所有线网的重要步骤。迷宫布线在新布线边代价的基础上同时考虑到线长和拥挤均衡，进行未连接线网的布线。计算 GRG 中新布线边代价如算法 4 所示。基于新布线边代价，不断为未连接线网执行迷宫布线直到不存在未连接线网，具体的迷宫算法如算法 5 所示。

算法 4 新布线边代价

if 两端子线网的边 e_i 已被原多端线网分解后的其他子线网经过 **then**
    cost(e_i)=0;
else
    cost(e_i)= *cost*(e_i)+*length*(e_i);//*length* 是指边 e_i 在 GRG 中的长度
    cost(e_i)= *cost*(e_i)+*cong*(e_i);// *cong* 是指边 e_i 在 GRG 中的拥挤度
end if

算法 5 基于新布线边代价的迷宫算法

输入：主阶段后仍未布线的剩余线网集合 *RN*
for 在 *RN* 中的每个两端线网 *i* **do**
  priority_queue *Q*；
  添加线网 *i* 的一个引脚至 *Q*，标记该引脚为待扩张点；
  while *Q* 不为空 **do**
    以一个单元网格大小沿所有布线方向扩张 *Q.top* 点；
    if 该点的标识是未扩张 **then**
      调用算法 4 以更新并计算线网 *i* 的布线代价；
      if 该点能继续扩张 **then**
        向 *Q* 中添加该点；
      end if
    end if
  end while
end for

5.6 实验结果

本章算法是采用 C/C++编程，并采用九个基准电路[149]验证本章算法的有效性。为了研究本章算法的有效性，六组实验在 5.6.2 小节至 5.6.7 小节中分别实施对比，而所有基准电路的参数说明及其初始布线阶段的布线结果则在 5.6.1 小节中描述。

5.6.1 基准电路的参数说明及其初始布线阶段的布线结果

表 5-2 的第 1 至 4 列分别给出基准电路中电路名称、网格大小、线网个数、单元个数，其中线网的规模大小从 11k 到 64k，单元的数目从 12k 至 66k。在 XGRouter 的初始布线阶段进行线网分解后，两端线网的个数是从 25k 至 169k。在表 5-2 至表 5-7 的符号‘AVG’表示平均值。表 5-2 中‘bf.nets’表示在 XGRouter 的初始布线阶段前需要进行总体布线的两端线网数目，而‘af.nets’表示在初始布线阶段后已经布通的两端线网数目。表 5-2 的第 7 列说明了在初始布线阶段后已经布通的两端线网数目占所有两端线网数目的百分比为 72.98%。因此，XGRouter 在初始布线阶段后可获得一个近似的布线拥挤分布情况。

表 5-2 测试电路的属性及初始布线的结果

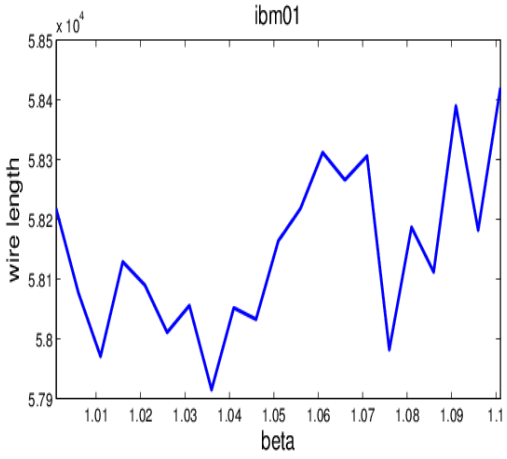
	基准电路			初始布线情况		
名称	网格大小	线网数	单元数	bf.nets	af.nets	百分比(%)
ibm01	64*64	11507	12036	25698	20127	78.32
ibm02	80*64	18429	19062	55739	39871	71.53
ibm03	80*64	21621	21924	43272	30181	69.75
ibm04	96*64	26163	26346	49821	34193	68.63
ibm06	128*64	33354	32185	75912	59812	78.79
ibm07	192*64	44394	44848	97391	69123	70.97
ibm08	192*64	47944	50691	121912	89134	73.11
ibm09	256*64	50393	51461	115871	91268	78.77
ibm10	256*64	64227	66948	169391	113459	66.98
AVG						72.98

5.6.2 O-ILP 模型的参数选择

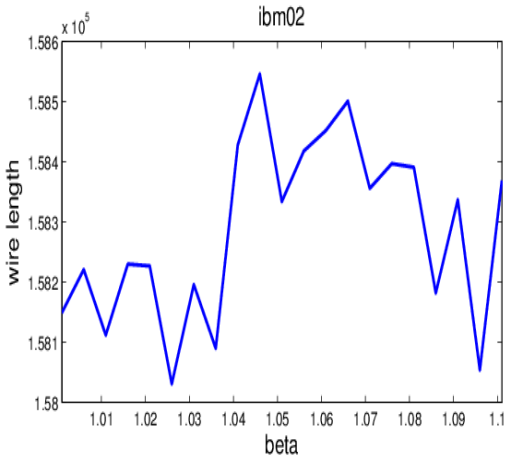
参数 β 是惩罚项，当总体布线方案违反最大通道容量约束，参数 β 对应的函数底数略大于 1。在实验中，底数的值分别设置为 1.001 至 1.101，以 0.005 递增，再分别作为惩罚项参数 β 的底数进行实验，并在如图 5-14(a)、(b)和(c)中分别给出基准电路 ibm01、ibm02 及 ibm03 上基于不同底数值所得到的优化结果。从图 5-14 所示的结果可看出这些电路对应的最佳线长一般在底数的取值从 1.026 至 1.036 的条件下获得。因此，在随后的实验中，本章不妨将底数的取值设置为 1.026。

另外，在所有基准电路中，其拥挤度集对应的标准差值一般为 0.01 至 1，而线长的取值一般为 10 至 1000。权重参数 α 的选择分别设置为从 20 至 200，以 2

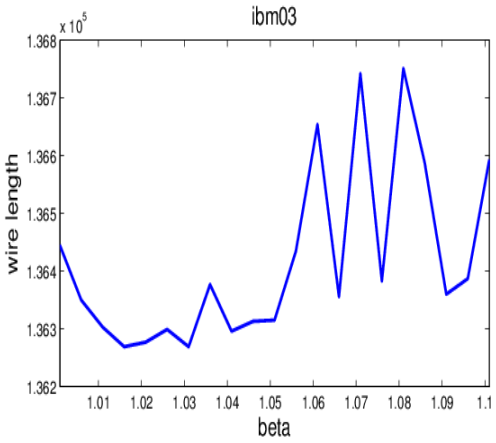
递增，从实验结果观察到权重参数α在这些取值范围内对最后的优化效果没有明显影响。因此，本章算法中参数α的值不妨设置为100。



(a) ibm01



(b) ibm02



(c) ibm03

图 5-14 在参数 β 的不同底数取值下总体布线优化结果

5.6.3 拥挤度均衡策略的有效性验证

为了研究O-ILP模型中拥挤均衡策略的有效性，本章开展了相关实验，其中表5-3列出了线长指标和能反映拥挤情况的标准差指标的实验结果。符号'WOS'和'WS'分别表示PSO算法中未考虑和有考虑拥挤均衡两种情况，'STD'表示拥挤度集合的标准差，'TWL'表示所有线网的总线长。

从表5-3可看出考虑到拥挤均衡策略，WS相对于WOS在标准差指标上减少率为4.93%，从中可看出由于所有GRG边的拥挤度集的标准差得到减少，本章O-ILP模型可获得更均匀的总体布线方案。表5-3的第2，4，6列分别给出了总线长的相关结果，从中可看出WS相对于WOS在总线长指标方面能够取得0.02%

的减少率。因此，相关实验结果表明采用拥挤均衡策略在大多数基准电路(从 ibm04 至 ibm10)中可获得更短的总线长，从而进一步验证 O-ILP 模型的有效性和性质 5.1 的合理性。

表 5-3 未加入和加入拥挤均衡策略的优化结果对比

基准电路	WOS		WS		减少率(%)	
	TWL	STD	TWL	STD	TWL	STD
ibm01	58137	0.0818	58149	0.0794	-0.02	2.93
ibm02	158100	0.0415	158259	0.0409	-0.10	1.45
ibm03	136261	0.0376	136283	0.0356	-0.02	5.32
ibm04	153589	0.0635	153560	0.0611	0.02	3.78
ibm06	262559	0.0169	262142	0.0159	0.16	5.92
ibm07	341835	0.0242	341782	0.0233	0.02	3.72
ibm08	380005	0.0200	379882	0.0183	0.03	8.50
ibm09	389344	0.0457	389288	0.0436	0.01	4.60
ibm10	542997	0.0244	542441	0.0224	0.10	8.20
AVG					0.02	4.93

5.6.4 本章算法的统计结果

表 5-4 本章算法的统计结果

基准电路	BT	WT	AE	(AE-BT)/AE	(AE-WT)/AE
ibm01	57855	58170	57989	0.0023	-0.0031
ibm02	157944	158139	158049	0.0007	-0.0006
ibm03	136236	136332	136284	0.0004	-0.0004
ibm04	153232	153570	153425	0.0013	-0.0009
ibm06	261618	262067	261729	0.0004	-0.0013
ibm07	341706	341866	341794	0.0003	-0.0002
ibm08	379774	380102	379902	0.0003	-0.0005
ibm09	389262	389701	389367	0.0003	-0.0009
ibm10	542413	542919	542568	0.0003	-0.0006
AVE				0.0007	-0.0009

因为用于求解 O-ILP 模型的 PSO 算法是一种随机算法，所以在每个基准电路的实验分别运行 20 次并在表 5-4 和图 5-14 中给出相应的统计实验结果。表 5-4 给出了多次运行结果的最佳值(BT)，最差值(WT)以及平均值(AE)，最后两列给出了最佳值相对平均值的改进率以及最差值相对平均值的增加率。从表 5-4 可看出，20 次实验结果中最差值和最佳值相对其平均值的平均改变率分别是-0.0009 和 0.0007，由此可见本章算法具有较好的鲁棒性。在后续的实验设计中，本章将

采用 20 次运行结果的平均值与其他算法的实验结果进行比较。

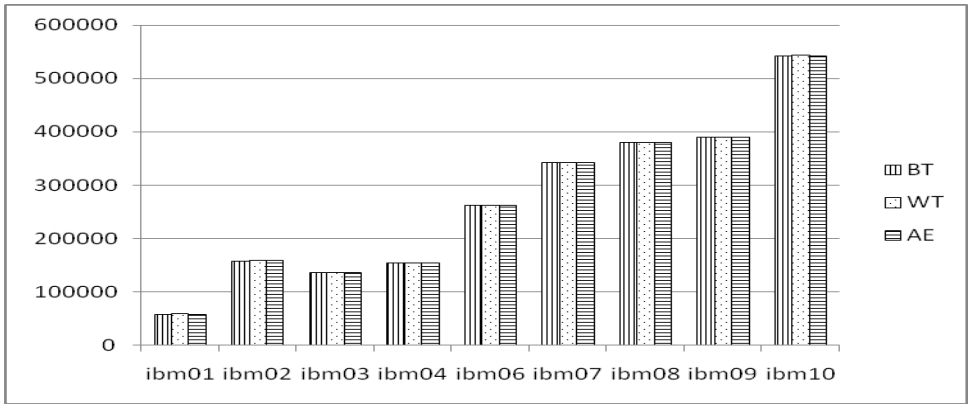


图 5-14 本章算法的统计结果柱状图

5.6.5 与五种矩形总体布线串行算法的对比

为了验证本章算法的有效性，将本章算法(OURS)与五种矩形总体布线串行算法[5,6,9,10,17]在上述基准电路上进行实验对比。这些矩形总体布线串行算法基于曼哈顿结构并且均在 ISPD98 基准电路上[149]进行测试，相应的实验结果在表 5-5 和图 5-15 中给出。从表 5-5 看出，在线长(TWL)指标上，本章算法相对这五种串行算法[6,10,5,9,17]分别取得 6.72%、7.53%、7.50%、7.01%及 6.68%的减少率，特别是在 ibm01 上与文献[8]的结果相比可达到 9.94%的减少率。因为所有算法均未产生溢出边，所以关于溢出数(OVF)的实验数据均为 0，未在表 5-5 中列出。由于本章算法引入 X 结构并从全局的角度进行总体布线(即本章算法是并行算法)，所以本章算法具有相对更强的线长优化能力，且能较好克服这些串行算法对线网布线顺序的依赖性问题。

表 5-5 与五种矩形总体布线串行算法在线长的对比

基准电路	[6]	[10]	[5]	[9]	[17]	OURS	减少率(%)				
							[6]	[10]	[5]	[9]	[17]
ibm01	63332	64389	62659	63720	62498	57989	8.44	9.94	8.99	7.45	7.21
ibm02	168918	171805	171110	170342	169881	158049	6.43	8.01	7.22	7.63	6.96
ibm03	146412	146770	146634	147078	146458	136284	6.92	7.14	7.34	7.06	6.95
ibm04	167101	169977	167275	170095	166452	153425	8.18	9.74	9.80	8.28	7.83
ibm06	277608	278841	277913	279566	277696	261729	5.72	6.14	6.38	5.82	5.75
ibm07	366180	370143	365790	369340	366133	341794	6.66	7.66	7.46	6.56	6.65
ibm08	404714	404530	405634	406349	404976	379902	6.13	6.09	6.51	6.34	6.19
ibm09	413053	414223	413862	415852	414738	389367	5.73	6.00	6.37	5.92	6.12
ibm10	578795	583805	590141	585921	579870	542568	6.26	7.06	7.40	8.06	6.43
AVE							6.72	7.53	7.50	7.01	6.68

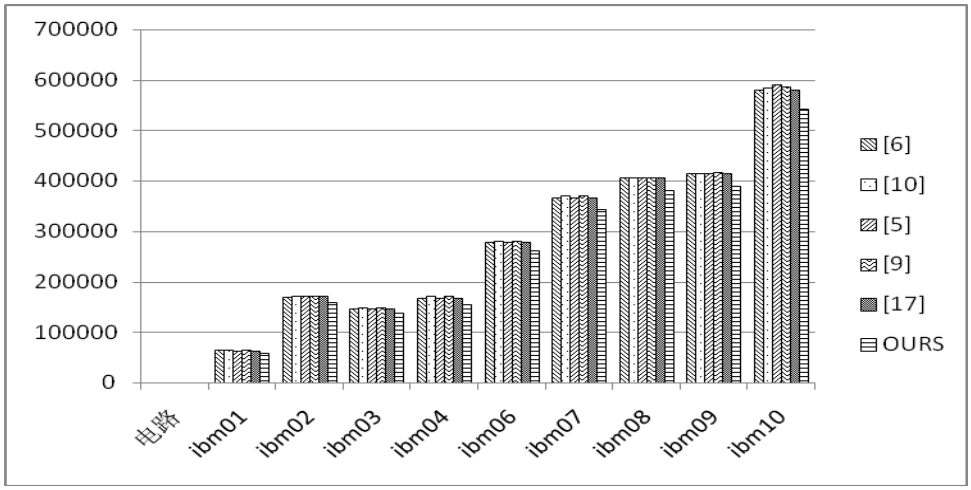


图 5-15 与五种矩形总体布线串行算法在线长的对比结果柱状图

5.6.6 与两种 X 结构总体布线算法的对比

将本章算法与两种 X 结构总体布线算法[57,150]进行对比，相应结果在表 5-6 中给出。文献[150]中 Labyrinth 总体布线算法在文献[57]中基于 X 结构重新设计相应的算法以进行对比。由于这两种 X 结构总体布线算法在有些基准电路中存在溢出数，所以在表 5-6 分别给出溢出数和总线长两个指标的实验结果。其中第 2，4，6 列分别给出了三种 X 结构总体布线算法的总线长结果，第 8 列和第 9 列分别给出了本章算法相对两种 X 结构总体布线算法在总线长上的减少率，分别为 22.46%和 16.22%。特别是在 ibm03 上相对文献[150]可达到 26.41%的线长减少率。表 5-6 中的相关实验结果显示本章算法可获得高质量的总体布线方案，其原因是本章算法可同时考虑到多数线网的走线状态，从全局角度进行总体布线，同时初始布线阶段给出一个有效的初始布线策略。

表 5-6 与两种 X 结构总体布线算法在 TWL 和 OVF 的对比

基准电路	[150]		[57]		OURS		减少率(%)	
	TWL	OVF	TWL	OVF	TWL	OVF	[150] TWL	[57] TWL
ibm01	76517	398	69575	60	57989	0	24.21	16.65
ibm02	204734	492	188691	0	158049	0	22.80	16.24
ibm03	185194	209	158837	0	136284	0	26.41	14.20
ibm04	196920	882	187443	385	153425	0	22.09	18.15
ibm06	346137	834	314082	0	261729	0	24.39	16.67
ibm07	449213	697	391289	0	341794	0	23.91	12.65
ibm08	469666	665	440612	1	379902	0	19.11	13.78
ibm09	481176	505	467727	1	389367	0	19.08	16.75
ibm10	679606	588	685521	661	542568	0	20.16	20.85
AVE	343240	586	322642	123	269012	0	22.46	16.22

5.6.7 与四种矩形总体布线并行算法的对比

表 5-7 与四种矩形总体布线并行算法在 TWL 的对比

基准电路	[139]	[138]	[140]	[11]	OURS	减少率(%)			
						[139]	[138]	[140]	[11]
ibm01	68981	67674	66058	62659	57989	15.93	14.31	12.22	7.45
ibm02	190418	182268	174062	171110	158049	17.00	13.29	9.20	7.63
ibm03	160755	151299	147524	146634	136284	15.22	9.92	7.62	7.06
ibm04	176610	173778	172652	167275	153425	13.13	11.71	11.14	8.28
ibm06	296981	282325	280007	277913	261729	11.87	7.30	6.53	5.82
ibm07	408510	394170	381694	365790	341794	16.33	13.29	10.45	6.56
ibm08	429913	415025	413300	405634	379902	11.63	8.46	8.08	6.34
ibm09	442514	418615	416554	413862	389367	12.01	6.99	6.53	5.92
ibm10	634247	593186	591036	590141	542568	14.45	8.53	8.20	8.06
AVE						14.18	10.42	8.88	7.01

将本章算法与四种矩形总体布线并行算法[11,138-140]进行对比，表 5-7 和图 5-16 给出相应的实验结果。从表 5-7 可看出，本章算法在总线长方面相对文献[139]，[138]，[140]，[11]中的四种算法分别取得 14.18%，10.42%，8.88%及 7.01%的减少率。特别是相对文献[139]在 ibm02 上可达到 17.00%的总线长减少率。本算法相对于其他并行算法能够有效减少总线长的原因包括：(1) XGRouter 引入了 X 结构，可相对曼哈顿结构取得更少的总线长；(2) 本章采用改进的 PSO 算法求解 O-ILP 模型，可解决采用随机取整方法求解 ILP 模型易产生偏差的情况，从而最终带来总线长的优化。

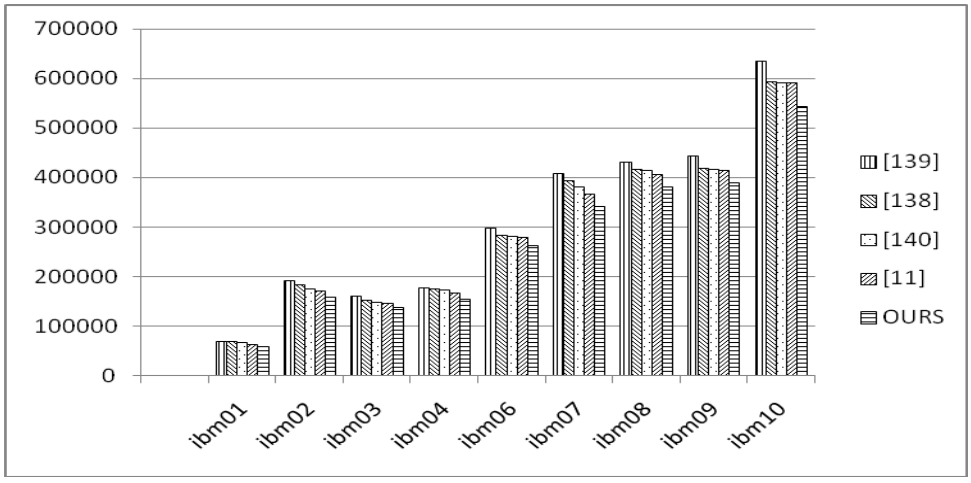


图 5-16 与四种矩形总体布线并行算法在 TWL 的对比结果柱状图

5.7 小结

本章提出一种基于 PSO 的高质量 X 结构总体布线算法，该算法不仅在大规

模电路中可获得较好解，还可在总线长方面相对现有工作而言取得最佳结果。而且本章提出一种新颖的 ILP 模型，可获得更为均为的总体布线方案，同时设计一种有效 PSO 算法用以求解 O-ILP 模型。本章工作是第一次使用并行算法求解 X 结构总体布线问题。

第六章 总结与展望

6.1 论文工作总结

当前集成电路产业向超深亚微米工艺不断推进，芯片的集成度进一步提高，一块芯片上所能集成的电路元件越来越多，加上存储空间的局限性和封装工艺的限制，对超大规模集成电路设计方法提出了新的挑战。VLSI 物理设计阶段中总体布线的结果对最后详细布线的成功与否和芯片的性能影响极大[1]。针对总体布线相关问题，研究人员提出了大量方法，但大部分总体布线算法都是以曼哈顿结构为模型基础。而基于曼哈顿结构进行线长与时延的优化，由于其布线走向有限，不能够充分地利用布线区域，导致互连线资源的过分冗余，故基于曼哈顿结构的优化策略在进行互连线线长优化时其优化能力受限。因此，有必要从根本入手改变传统的曼哈顿结构，故研究人员开始尝试以非曼哈顿结构为基础模型进行布线，实现芯片整体性能的优化。

非曼哈顿结构的引入使得物理设计的诸多性能得到提高，比曼哈顿结构具有明显的优势。曼哈顿结构下的总体布线算法虽然部分可以应用到非曼哈顿结构布线，但需要进行特别的处理，使算法变得更加复杂，从而难以直接有效地对该问题进行求解。然而，已有的解决非曼哈顿结构下总体布线问题的启发式算法，或解决的问题规模小，或只能解决局部阶段工作，或与曼哈顿结构算法相比带来优化程度不大。

基于此背景，本文选取获得工业联盟支持的 X 结构作为非曼哈顿结构的代表，将针对 X 结构下 VLSI 总体布线中存在的若干问题和困难展开研究，根据 X 结构的性质及该结构下总体布线问题的特点，作了一些有益的尝试，主要的研究内容及创新之处如下：

(1) 针对 X 结构下 VLSI 总体布线算法中多端线网的最佳连接模型——X 结构 Steiner 最小树构建问题，本文设计并实现一种基于离散粒子群优化的有效算法。首先为了克服 PSO 算法用于高维空间优化问题中收敛速度慢的不足，设计一种自适应的参数调节策略。其次，根据 Steiner 最小树问题的特点，结合遗传算法中的变异算子和交叉算子，提出了可求解离散问题的 PSO 更新操作策略。然后，设计了一种适合于 X 结构 Steiner 最小树问题的有效编码策略。最后，设计相应的仿真实验，通过实验结果说明所提算法的可行性和有效性。

(2) 构建时延驱动 Steiner 树是 VLSI 性能驱动总体布线阶段中一个非常重要的问题。针对此问题，本文基于 MOPSO 和 Elmore 时延模型，提出了一种构建

X 结构 MRMCST 的有效算法，进而有助于 X 结构下性能驱动总体布线问题的研究。首先，相对(1)工作中只考虑 Steiner 点的变换，该算法设计了边变换策略，用以增强 PSO 算法中粒子的进化能力。该算法同时考虑到 Steiner 点和布线边的变换策略，相对(1)工作能够进一步加强布线树的线长优化能力，从而取得更强的线长优化。其次，基于 pseudo-Steiner 点及其灵活度的定义，通过四种边选择方式重新调度 pseudo-Steiner 点的位置，提出了一种有效的编码策略。边变换策略可很好结合所提出的边点对编码策略以减少拐弯数。最后，为了能够同时优化 MRMCST 模型中线长和半径两个相互竞争的目标，本文基于多目标 PSO 模型设计相应的多目标算法以构造最终的时延驱动 X 结构 Steiner 最小树。实验结果表明最终的布线树在线长、半径、最大源漏点时延、总的源漏点时延以及拐弯数等五个指标上均能取得较好的优化效果，并且在不同 IC 工艺下算法是稳定且有效的。另外，拐弯数是影响芯片可制造性的一个关键指标且对 VLSI 布线有重要意义，本文是第一次在非曼哈顿结构性能布线工作中考虑到拐弯数的优化。

(3) 针对多层绕障 X 结构 Steiner 最小树问题，以最小化布线总代价为目标，同时考虑到通孔数的优化，提出了一种基于离散 PSO 算法和惩罚机制的多层绕障 X 结构 Steiner 最小树构建算法。该算法在粒子的更新公式中引入边变换操作以加强布线树的拓扑变化能力，进一步增加布线树的绕障能力，并将(2)工作中的单层边点对编码策略扩展至多层空间上的多层边点对编码策略。通过提出合适的惩罚机制用以融入粒子的适应度函数设计，更有效地考虑绕障情况和通孔产生情况，继而从基于多层结构的全局角度考虑多层布线问题的求解空间，从而达到更有力优化布线总代价的目的。同时，引入预处理策略大大减少了绕障情况的重复判断次数和通孔数的重复计算次数。实验结果表明该算法相关策略的有效性以及该算法相对同类算法能取得可观的布线总代价减少率，是在当前同类算法中具有最强的布线总代价优化能力。此外，本文针对定向边约束的绕障 X 结构 Steiner 最小树这一更为接近实际芯片设计工艺的问题，重新定义布线边的代价，并基于多层绕障 X 结构 Steiner 最小树构建算法的基础上，构建求解该问题的有效算法，并探讨不同定向边约束模型的优劣，以期为非曼哈顿结构总体布线提供一定的理论支撑。本文是第一次开展定向边约束的绕障 X 结构 Steiner 最小树问题的研究。

(4) 针对 X 结构下的总体布线问题，提出一种基于整数线性规划模型、划分策略及 PSO 等技术的高质量 X 结构 VLSI 总体布线算法。该算法基于一种新颖的 ILP 模型，可获得更为均为的总体布线方案，同时设计一种有效 PSO 算法用以求解本文的 O-ILP 模型。为了使 O-ILP 模型更为有效求解，采用划分策略，将大规模的总体布线问题划分为多个中小规模的问题，从总体布线问题中最拥挤的区域开始布线，并不断扩张布线区域的大小。在总体布线后续阶段，设计一种基

于新布线边代价的迷宫布线算法，以线长优化和拥挤度均衡为目标，布通一些仍未完成总体布线的线网。本文是第一次使用并行算法求解 X 结构总体布线问题。实验结果表明该算法不仅在大规模电路中可获得较好解，还可在总线长方面相对现有工作而言取得最佳结果。

6.2 未来工作展望

如前所述，非曼哈顿结构的提出，带来了一些机遇和挑战，引起物理设计中总体布线阶段一系列问题的产生，同时鉴于总体布线的重要性，本文开展的以 X 结构为非曼哈顿结构代表的总体布线算法研究是一项具有重要科学意义和重大研究价值的工作。在这一背景下，本文开展了一些相关研究工作，并取得一定的研究成果，但是这些研究只是一个起步，部分工作还不够成熟，需要进一步考虑很多问题并作深入研究，继而开展进一步的研究工作。就作者对 X 结构下总体布线相关工作的认识和理解，未来迫切需要进一步开展的研究工作包括以下几方面：

(1) 针对 X 结构总体布线中最重要的一个连接模型——X 结构 Steiner 最小树问题，应该进一步研究 X 结构 Steiner 最小树模型的几何性质，以期能够充分利用该模型潜在的更多几何性质，设计更为有效的优化策略，从而获得质量、效率更好的构建方案。同时参照求解矩形 Steiner 最小树问题的精确算法的设计，探索求解 X 结构 Steiner 最小树问题的精确算法，以期能够建立一个标杆，以更好地衡量其他求解 X 结构 Steiner 最小树问题的启发式算法所获得解方案的质量，为启发式算法的进一步设计提供较为明确的指导。

(2) 由于多层布线问题的研究是当前 VLSI 研究的一个热点，本文已探索了多层绕障 X 结构 Steiner 最小树问题的求解，并获得一定的研究成果。下一步工作可利用该研究中多层布线树的编码等相关策略，将 X 结构 Steiner 最小树问题和时延驱动 X 结构 Steiner 最小树问题扩展至多层布线的环境下，以更全面地研究 X 结构下多层总体布线相关问题。

参考文献

[1] 洪先龙, 严晓浪, 乔长阁. 超大规模集成电路布图理论与算法. 北京: 科学出版社, 1998.

[2] 吴雄. 国际集成电路产业发展现状与前景.电子与自动化，1996, 6: 3-8.

[3] 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.

[4] Gao J R, Wu P C, Wang T C. A new global router for modern designs. In: Proceedings of the 2008 Asia and South Pacific Design Automation Conference. Los Alamitos, CA, USA: IEEE Press, 2008. 232–237.

[5] Zhang Y, Xu Y, Chu C. Fastroute 3.0: A fast and high quality global router based on virtual capacity. In: Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design. Piscataway, NJ, USA: IEEE Press, 2008. 344–349.

[6] Roy A J, Markov I L. High-performance routing at the nanometer scale. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2008, 27(6): 1066–1077.

[7] Liu W H, Li Y L. Optimizing the antenna area and separators in layer assignment of multilayer global routing. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2014, 33(4): 613-626.

[8] Shojaei H, Davoodi A, Basten T. Collaborative multiobjective global routing. IEEE Transactions on Very Large Scale Integration Systems, 2013, 21(7): 1308-1321.

[9] Moffitt M D. Maizerouter: Engineering an effective global router. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2008, 27(11): 2017–2026.

[10] Ozdal M M, Wong M D F. Archer: A history-based global routing algorithm. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2009, 28(4): 528–540.

[11] Cho M, Lu K, Yuan K, et al. Boxrouter 2.0: A hybrid and robust global router with layer assignment for routability. ACM Transactions on Design Automation of Electronic Systems, 2009, 14(2): 1–21.

[12] Hsh P Y, Chen H T, Hwang T T. Stacking signal TSV for thermal dissipation in global routing for 3D IC. In: Proceeding of the 18th Asia and South Pacific Design Automation Conference. Yokohama, Japan: IEEE Press, 2013. 699-704.

[13] Liu W H, Wei Y G, Sze C, et al. Routing congestion estimation with real design constraints. In: Proceedings of the 50th Annual Design Automation Conference. New York, NY, USA: ACM Press, 2013. Article No. 92.

[14] Xu Y, Zhang Y, Chu C. Fastroute 4.0: Global router with efficient via minimization. In: Proceedings of the 2009 Asia and South Pacific Design Automation Conference. Piscataway, NJ, USA: IEEE Press, 2009. 576–581.

[15] Wu T H, Davoodi A, Linderoth J T. GRIP: Scalable 3D global routing using integer programming. In: Proceedings of the 46th Annual Design Automation Conference. New York, NY, USA : ACM Press, 2009. 320–325.

[16] Dai K R, Liu W H, Li Y L. Efficient simulated evolution based rerouting and congestion-relaxed layer assignment on 3-D global routing. In: Proceedings of the 2009 Asia and South Pacific Design Automation Conference. Piscataway, NJ, USA: IEEE Press, 2009. 570–575.

[17] Chang Y J，Lee Y T, Gao J R, et al. NTHU-Route 2.0: A robust global router for modern designs. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2010, 29(12): 1931-1944.

[18] Dai K R, Liu W H, Li Y L. NCTU-GR: Efficient simulated evolution-based rerouting and congestion-relaxed layer assignment on 3-D global routing. IEEE Transactions on Very Large Scale Integration Systems, 2012, 20(3): 459-472.

[19] Cong J, He L, Koh C K, et al. Performance optimization of VLSI interconnect layout. Integration, the VLSI Journal, 1996, 21(1-2): 1-94.

[20] 洪先龙, 朱祺, 经彤, 等. 非直角互连——布线技术发展的新趋势. 半导体学报, 2003, 24(3): 225-233.

[21] Chang C F, Chang Y W. X-Route: An X-architecture full-chip multilevel router. In: Proceedings of the 2007 IEEE International SOC Conference. Hsin Chu, Taiwan: IEEE Press, 2007. 229-232.

[22] 马军, 杨波, 马绍汉. 近乎最佳的 Manhattan 型 Steiner 树近似算法. 软件学报, 2000, 11(2): 260-264.

[23] Erzin A I, Zalyubovskiy V V, Shamardin Y V, et al. A timing driven congestion aware global router. In: Proceedings of the 2nd International Conference on Emerging Applications of Information Technology. Novosibirsk, Russia: IEEE Press, 2011. 375-378.

[24] Ma X, Liu Q. An artificial fish swarm algorithm for steiner tree problem. In: Proceedings of the 2009 IEEE International Conference on Fuzzy Systems. Jeju Island Korea: IEEE Press, 2009. 59-63.

[25] Dong C, Wang G F , Chen Z Y, et al. A VLSI routing algorithm based on improved DPSO. In: Proceedings of the 2009 IEEE International Conference on Intelligent Computing and Intelligent Systems. Shainghai, China: IEEE Press, 2009. 802-805.

[26] Sarrafzadeh M, Feng L K, Wong C K. Single-layer global routing. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1994, 13(1): 38-47.

[27] Bozorgzadeh E, Kastner R, Sarrafzadeh M. Creating and exploiting flexibility in rectilinear Steiner trees. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2003, 22(5): 605-615.

[28] Chen H Y, Hsu C H, Chang Y W. High-performance global routing with fast overflow reduction. In: Proceedings of the 2009 Asia and South Pacific Design Automation Conference. Piscataway, NJ, USA: IEEE Press, 2009. 582-587.

[29] 张轶谦, 洪先龙, 蔡懿慈 .基于精确时延模型考虑缓冲器插入的互连线优化算法. 电子学报, 2005, 33(5): 783-787.

[30] Liu B, Cai Y C, Zhou Q, et al. Algorithm for post global routing RLC crosstalk avoidance. In: Proceedings of the 7th International Conference on Solid-State and Integrated Circuits Technology. Beijing, China: IEEE Press, 2004. 1948-1951.

[31] Wu D, Hu J, Mahapatra R, et al. Layer assignment for crosstalk risk minimization. In: Proceedings of the 2004 Asia and South Pacific Design Automation Conference. Piscataway, NJ, USA: IEEE Press, 2004. 159-162.

[32] ISPD 2007 Global Routing Contest [Online]. Available: http://www.sigda.org/ispd2007/ contest.html.

[33] ISPD 2008 Global Routing Contest [Online]. Available: http://www.sigda.org/ispd2008 /contests/ ispd08rc.html.

[34] Liu W H, Kao W C, Li Y L, et al. NCTU-GR 2.0: Multithreaded collision-aware global routing with bounded-length maze routing. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2013, 32(5): 709-722.

[35] Han Y, Ancajas D M, Chakraborty K, et al. Exploring high-throughput computing paradigm for global routing. IEEE Transactions on Very Large Scale Integration Systems, 2014, 22(1): 155-167.

[36] Kar B, Sur-Kola S, Mandal C. STAIRoute: Global routing using monotone staircase channels. In: Proceeding of the 2013 IEEE Computer Society Annual Symposium on VLSI. Natal, Brazil: IEEE Press, 2013. 90-95.

[37] Zhang Y L, Chowdhury S, Pan D Z. BOB-router: A new buffering-aware global router with over-the-block routing resources optimization. In: Proceeding of the 19th Asia and South Pacific Design Automation Conference. Singapore, IEEE Press: 2014. 513-518.

[38] Qi Z, Cai Y, Zhou Q, et al. VFGR: A very fast parallel global router with accurate congestion modeling. In: Proceeding of the 19th Asia and South Pacific Design Automation Conference. Singapore, IEEE Press: 2014. 525-530.

[39] Teig S L. The X architecture: not your father's diagonal wiring. In: Proceeding of the 2002 International Workshop on System Level Interconnect Prediction. New York, NY, USA: ACM Press, 2002. 33-37.

[40] X Initiative Home Page, 2001. [Online]. Available: http://www.xinitiative.com.

[41] Thurber A, Xue G. Computing hexagonal steiner trees using PCx for VLSI. In: Proceeding of the 6th IEEE International Conference on Electronic, Circuits & System. Pafos, Spain: IEEE Press, 1999. 381-384.

[42] Coulston C S. Constructing exact octagonal Steiner minimal tree. In: Proceeding of the 13th ACM Great Lakes Symposium on VLSI. New York, NY, USA: ACM Press, 2003. 28-29.

[43] Chiang C, Chiang C S. Octilinear Steiner tree construction. In: Proceeding of the 45th Midwest Symposium on Circuits and Systems. Tulsa, USA: IEEE Press, 2002. 603-606.

[44] Samanta T,Ghosal P, Rahaman H,et al. A heuristic method for constructing hexagonal Steiner minimal trees for routing in VLSI. In: Proceeding of the 2006 International Symposium on Circuits and Systems. Island of Kos, Greece: IEEE Press, 2006. 1788-1791.

[45] Zhu Q, Zhou H, Jing T, et al. Spanning graph-based nonrectilinear Steiner tree algorithms. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2005, 24 (7): 1066-1075.

[46] Sardar R, Ratna M, and Tuhina S. Geometry independent wirelength estimation method in VLSI routing. In: Proceeding of the 26th International Conference on VLSI Design and 12th International Conference on Embedded Systems. Pune, India: IEEE Press, 2013. 257-261.

[47] Yan J T, Wang T T, Lin K M, et al. Timing-driven octilinear Steiner tree construction based on Steiner-point reassignment and path reconstruction. ACM Transactions on Design Automation of Electronic Systems, 2008, 13(2): 1-18.

[48] Samanta T, Rahaman H, Dasgupta P. Near-optimal Y-routed delay trees in nanometric interconnect design. IET Computers & Digital Techniques, 2011, 5(1): 36-48.

[49] Samanta R, Erzin A, Raha S. Timing-driven Steiner tree construction on uniform λ-geometry. In: Proceeding of the 18th International Symposium on VLSI Design and Test. Coimbatore, India: IEEE Press, 2014. 1-4

[50] Koh C K, Madden P H. Manhattan or non-manhattan? A study of alternative VLSI routing architectures. In: Proceeding of the 10th Great Lakes symposium on VLSI. New York, USA: ACM Press, 2000. 47-52.

[51] Hursey E, Jayakumar N, Khatri S P. Non-manhattan routing using a manhattan router. In: Proceeding of the 18th International Conference on VLSI Design. Kolkata, India: IEEE Press, 2005. 445-450.

[52] Ho T Y. PIXAR: A performance-driven X-architecture router based on a novel multilevel framework. Integration, the VLSI Journal, 2009, 42(3): 400-408.

[53] Ho T Y. A performance-driven multilevel framework for the X-based full-chip router. Springer-Verlag Berlin Heidelberg: Integrated Circuit and System Design, Power and Timing Modeling, Optimization and Simulation. 2009. 209-218.

[54] Qu H C, Chien H C C, Chang Y W. Non-uniform multilevel analog routing with matching constraints. In: Proceedings of the 49th Annual Design Automation Conference. New York, USA: ACM Press, 2012. 549-554.

[55] Ito N, Katagiri H, Yamashita R, et al. Diagonal routing in high performance microprocessor design. In: Proceedings of the 2006 Asia and South Pacific Design Automation Conference. Piscataway, NJ, USA: IEEE Press, 2006. 624-629.

[56] Agnihotri A R, Madden P H. Congestion reduction in traditional and new routing architectures. In: Proceedings of the 13th ACM Great Lakes symposium on VLSI. New York, NY, USA: ACM Press, 2003. 211-214.

[57] Hu Y, Jing T, Hong X, et al. A routing paradigm with novel resources estimation and routability models for X-architecture based physical design. Springer-Verlag Berlin Heidelberg: Embedded Computer Systems: Architectures, Modeling, and Simulation. 2005. 344-353.

[58] Cao Z, Jing T, Hu Y, et al. DraXRouter: Global routing in X-architecture with dynamic resource assignment. In: Proceedings of the 2006 Asia and South Pacific Design Automation Conference. Piscataway, NJ, USA: IEEE Press, 2006. 618-623.

[59] Yang P, Yang H Y, Qiu W, et al. Optimal approach on net routing for VLSI physical design based on Tabu-ant colonies modeling. Applied Soft Computing, 2014, 21: 376-381.

[60] Bhattacharya P, Khan A, Sarkar, S K. A global routing optimization scheme based on ABC algorithm. In: Proceeding of the 2nd International Conference on Advanced Computing, Networking and Informatics. Switzerland: Springer International Publishing, 2014. 189-197.

[61] Arora T, Mose M E. Ant colony optimization for power efficient routing in manhattan and non-manhattan VLSI architectures. In: Proceeding of the 2009 Swarm intelligence symposium. Nashville, USA: IEEE Press, 2009. 137-144.

[62] San P P, Ling S H, Nuryani S L, et al. Evolvable rough-block-based neural network and its biomedical application to hypoglycemia detection system. IEEE Transactions on Cybernetics, 2014, 44(8): 1338-1349.

[63] Liu L L, Yang S X, Wang Q. Particle swarm optimization with composite particles in dynamic environments. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 2010, 40(6): 1373–1392.

[64] Chou H H, Hsu L Y, Hu H T. Turbulent-PSO-based fuzzy image filter with no-reference measures for high-density impulse noise. IEEE Transactions on Cybernetics, 2013, 43(1): 296–307.

[65] Qu B Y, Liang J J, Suganthan P N. Niching particle swarm optimization with local search for multi-modal optimization. Information Sciences, 2012, 197: 131–143.

[66] Nasir M, Das S, Maity D, et al. A dynamic neighborhood learning based particle swarm optimizer for global numerical optimization. Information Sciences, 2012, 209: 16–36.

[67] Xue B, Zhang M J, Browne M N. Particle swarm optimization for feature selection in classification: A multi-objective approach. IEEE Transactions on Cybernetics, 2013, 43(6): 1656–1671.

[68] Ruiz-Cruz R, Sanchez E N, Ornelas-Tellez F, et al. Particle swarm optimization for discrete-time inverse optimal control of a doubly fed induction generator. IEEE Transactions on Cybernetics, 2013, 43(6): 1698–1709.

[69] Peng S J, Chen G L, Guo W Z. A discrete PSO for partitioning in VLSI circuit. In: Proceedings of the 2009 International Conference on Computational Intelligence and Software Engineering. Wuhan, China: IEEE Press, 2009. 1-4.

[70] Peng S J, Chen G L, Guo W Z. A multi-objective algorithm based on discrete PSO for VLSI partitioning problem. In: Proceedings of the 2nd International Conference on Quantitative Logic and Soft Computing. Jimei, China: Springer-Verlag Berlin Heidelberg, 2010. 651-660.

[71] 郭文忠, 陈国龙. 一种求解多目标最小生成树问题的有效粒子群优化算法. 模式识别与人工智能, 2009, 22(4): 597-604.

[72] Chen G L, Guo W Z, Chen Y Z. A PSO-based intelligent decision algorithm for VLSI floorplanning. Soft Computing, 2010, 14(12): 1329-1337.

[73] Chen J Z, Chen G L, Guo W Z. A discrete PSO for multi-objective optimization in VLSI floorplanning. In: Proceedings of the 4th International Symposium on Intelligence Computation and Applications. Huangshi, China: Springer-Verlag Berlin Heidelberg, 2009. 400-410.

[74] 刘耿耿, 王小溪, 陈国龙, 等. 求解 VLSI 布线问题的离散粒子群优化算法. 计算机科学, 2010, 37(10): 197-201.

[75] Shen Y L, Liu Q Z, Guo W Z. Obstacle-avoiding rectilinear Steiner minimum tree construction based on discrete particle swarm optimization. In: Proceedings of the 7th International Conference on Natural Computation. Shanghai, China: IEEE Press, 2011. 2179–2183.

[76] Liu G G，Chen G L,Guo W Z，et al. DPSO-based rectilinear Steiner minimal tree construction considering bend reduction. In: Proceedings of the 7th International Conference on Natural Computation. Shanghai, China: IEEE Press, 2011. 1161-1165.

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

[78] Eberhart R C, Kennedy J. A new optimizer using particles swarm theory. In: Proceedings of the 6th International Symposium on Micro Machine and Human Science. Nagoya, Japan: IEEE press, 1995. 39-43.

[79] Tyagi G, Pandit M. Combined heat and power dispatch using particle swarm optimization. In: Proceedings of the 2012 IEEE Students' Conference on Electrical, Electronics and Computer Science. Bhopal, India: IEEE Press, 2012. 1-4.

[80] Padma M P, Komorasamy G. A modified algorithm for clustering based on particle swarm optimization and K-means. In: Proceedings of the 2012 International Conference on Computer Communication and Informatics. Coimbatore, India: IEEE Press, 2012. 1-5.

[81] Kennedy J, Eberhart R C. A discrete binary version of the particle swarm algorithm. In: Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics. Orlando, FL, USA: IEEE Press, 1997. 4104-4109.

[82] Clerc M. Discrete particle swarm optimization–illustrated by the traveling salesman problem. In: Onwubolu G C, Babu B V (eds) New Optimization Techniques in Engineering. Berlin, Germany: Springer-Verlag Berlin Heidelberg, 2004. 219-239.

[83] Pan Q K, Tasgetiren M F, Liang Y C. A discrete particle swarm optimization algorithm for the permutation flowshop sequecing problem with makespan criteria. In: Proceedings of the 26th SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence. London, British: Springer, 2006. 19-31.

[84] Gottlieb J, Julstrom B A, Rothlauf F, et al. Prüfer Numbers: A poor representation of spanning trees for evolutionary search. In: Proceedings of the 2001 Genetic and Evolutionary Computation Conference. San Francisco, USA: Morgan Kaufmann, 2001. 343-350.

[85] Shi Y H, Eberhart R C. A modified particle swarm optimizer. In: Proceedings of the 1998 IEEE International Conference on Evolutionary Computation. Anchorage, AK, USA: IEEE Press, 1998. 69-73.

[86] 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.

[87] Beasley J E. OR-Library: distributing test problems by electronic mail. The Journal of the Operational Research Society, 1990, 41(11): 1069-1072.

[88] Zachariasen M. GeoSteiner Homepage, 2003. [Online]. Available: http://www.diku.dk/geosteiner.

[89] Ho T Y, Chang Y W, Chen S J. Full-chip nanometer routing techniques. Berlin, Germany: Springer-Verlag Berlin Heidelberg, 2007.

[90] Hu J, Sapatnekar S. A survey on multi-net global routing for integrated circuits. Integration, the VLSI Journal, 2001, 31(1): 1-49.

[91] Warme D M, Winter P, Zachariasen M. Exact algorithms for plane steiner tree problems: A computational study. Netherlands, Dordrecht: Kluwer Academic Publishers, 2000. 81-116.

[92] Borah M, Owens R M, Irwin M J. An edge-based heuristic for Steiner routing. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1994, 13(12): 1563-1568.

[93] 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, 2008, 27(1): 70-83.

[94] Cong J, Kahng A B, Robins G, et al. Provably good performance-driven global routing. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1992, 11(6): 739–752.

[95] Seo D Y, Lee D T. On the complexity of bicriteria spanning tree problems for a set of points in the plane. [PhD Dissertation]. USA: Northwestern University, 1999.

[96] Hou H, Hu J, Sapatnekar S S. Non-hanan routing. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1999, 18(4): 436-444.

[97] Yan J T. Dynamic tree reconstruction with application to timing-constrained congestion-driven global routing. IET Computers & Digital Techniques, 2006, 153(2): 117-129.

[98] Liang J, Hong X, Jing T. G-Tree: Gravitation-direction-based rectilinear Steiner minimal tree construction considering bend reduction. In: Proceedings of the 7th international conference on ASIC. Guilin, China: IEEE press, 2007. 1114-1117.

[99] 梁敬弘, 洪先龙, 经彤. G-Tree:基于引力指向技术减少拐弯的 Steiner 树算法. 计算机辅助设计与图形学学报, 2008, 2: 144-148.

[100] Liu G G, Chen G L, Guo W Z. DPSO based octagonal steiner tree algorithm for VLSI routing. In: Proceedings of the 5th International Conference on Advanced Computational Intelligence. Najing, China: IEEE press, 2012. 383-387.

[101] Boese K D, Kahng A B, McCoy B A, et al. Near optimal critical sink routing tree constructions. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1995, 14(12): 1417-1436.

[102] Ismail Y I, Friedman E G, Neves J L. Equivalent elmore delay for RLC trees. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2000, 19(1): 83-97.

[103] Guo W Z, Park J H, Yang L T, et al. Design and analysis of a MST-based topology control scheme with PSO for wireless sensor networks. In: Proceedings of the 2011 IEEE Asia-Pacific Services Computing Conference. Jeju Island, Korea: IEEE press, 2011. 360–367.

[104] Guo W Z, Xiong N X, Vasilakos A V, et al. Distributed k-connected fault-tolerant topology control algorithms with PSO in future autonomic sensor systems. International Journal of Sensor Networks, 2012, 12(1): 53–62.

[105] Balling R. The maximin fitness function: multiobjective city and regional planning. In: Proceedings of the 2nd International Conference on Evolutionary Multi-criterion Optimization. Berlin, Germany: Springer-Verlag Berlin Heidelberg, 2003. 1-15.

[106] Laumanns M, Thiele L, Deb K, et al. Combining convergence and diversity in evolutionary multi-objective optimization. Evolutionary Computation, 2002, 10(3): 263–282.

[107] Alfred V A, John E H, Jeffrey U. Data structures and algorithms. Boston, MA, USA: Addison-Wesley Longman Publishing, 1983.

[108] Julstrom B A. Encoding rectilinear Steiner trees as lists of edges. In: Proceeding of the 2001 ACM Symposium on Applied Computing. New York, NY, USA: ACM Press, 2001. 356–360.

[109] Rudolph G. Convergence analysis of canonical genetic algorithms. IEEE Transactions on Neural Networks, 1994, 5(1): 96-101.

[110] Lv H, Zheng J, Zhou C, et al. The convergence analysis of genetic algorithm based on space mating. In: Proceeding of the 5th International Conference on Natural Computation. Tianjin, China: IEEE press, 2009. 557-562.

[111] Deb K, Pratap A, Agarwal S, et al. A fast and elitistmultiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 2002, 6(2):182–197.

[112] Zitzler E, Laumanns M, Thiele L. SPEA2: Improving the strength pareto evolutionary algorithm. In: Giannakoglou K C, Tsahalis D T, Periaux J, Papailiou K D, Fogarty T (eds) Evolutionary methods for design optimization and control with applications to industrial problems. International Center for Numerical Methods in Engineering, 2001. 95–100.

[113] Zitzler E. Evolutionary algorithms for multiobjective optimization:methods and applications. [PhD Dissertation]. Switzerland: Swiss Federal Institute of Technology, 1999.

[114] Zitzler E, Thiele L, Laumanns M, et al. Performance assessment of multiobjective optimizers: An analysis and review. IEEE Transactions on Evolutionary Computation, 2003, 7(2): 117–132.

[115] Conover W J. Practical nonparametric statistics. NewYork: Wiley, 1999.

[116] Feng Z, Hu Y, Jing T, et al. An O(nlogn) algorithm for obstacle-avoiding routing tree construction in the λ-geometry plane. In: Proceeding of the 2006 International Symposium on Physical Design. New York, USA: ACM Press, 2006. 48-55.

[117] Jing T T, Feng Z, Hu Y, et al. λ-OAT: λ-geometry obstacle-avoiding tree construction with O(nlogn) complexity. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2007, 26(11): 2073-2079.

[118] Huang X, Liu G G, Guo W Z, et al. Obstacle-avoiding octagonal Steiner tree construction based on particle swarm optimization. In: Proceeding of the 9th International Conference on Natural Computation. Shenyang, China: IEEE Press, 2013. 539- 543.

[119] Liu G G, Guo W Z, Niu Y Z, et al. A PSO-based-timing-driven octilinear Steiner tree algorithm for VLSI routing considering bend reduction. Soft computing, 2014. DOI: 10.1007/s00500-014-1329-2.

[120] Yang Y, Zhu Q, Jing T, et al. Rectilinear Steiner minimal tree among obstacles. In: Proceeding of the 5th International Conference on ASIC. Beijing, China: IEEE Press, 2003. 348–351.

[121] Hu Y, Jing T, Hong X L, et al. An-OARSMan: Obstacle-avoiding routing tree construction with good length performance In: Proceeding of the 2005 Asia and South Pacific Design Automation Conference. New York, USA: ACM Press, 2005. 7–12.

[122] Zhou Z, Jiang C D, Huang L S, et al. On optimal rectilinear shortest paths and 3-Steiner tree routing in presence of obstacles. Journal of Software, 2003, 14(9): 1503-1514.

[123] Ajwani G, Chu C, Mak W K. FOARS: FLUTE based obstacle-avoiding rectilinear Steiner tree construction. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2011, 30 (2): 194-204.

[124] Liu C H, Kuo S Y, Lee D T, et al. Obstacle-avoiding rectilinear Steiner tree construction: a Steiner-point-based algorithm. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2012, 31 (7): 1050-1060.

[125] Chow W K, Li L, Young E F, et al. Obstacle-avoiding rectilinear Steiner tree construction in sequential and parallel approach. Integration, the VLSI Journal, 2014, 47(1): 105-114.

[126] Huang T, Li L, Young E F. On the construction of optimal obstacle-avoiding rectilinear Steiner minimum trees. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2011, 30(5): 718-731.

[127] Huang T, Young E F. ObSteiner: An exact algorithm for the construction of rectilinear Steiner minimum trees in the presence of complex rectilinear obstacles. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2013, 32(6): 882-893.

[128] Totsukawa H, Senou H, Ohmura M. A parallel genetic algorithm for 3-D rectilinear Steiner tree with bounded number of bends. In: Proceeding of the 51st Midwest Symposium on Circuits and Systems. Knoxville, USA: IEEE Press, 2008. 89-92.

[129] Lin C W, Huang S L, Hsu K C, et al. Multilayer obstacle-avoiding rectilinear Steiner tree construction based on spanning graphs. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2008, 27(11): 2007-2016.

[130] Yildiz M C, Madden P H. Preferred direction Steiner trees. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2002, 21(11): 1368-1372.

[131] Liu C H, Chou Y H, Yuan S T, et al. Efficient multilayer routing based on obstacle-avoiding preferred direction Steiner tree. In: Proceeding of the 2008 International Symposium on Physical Design. New York, USA: ACM Press, 2008. 118-125.

[132] Liu C H, Chen I C, Lee D T. An efficient algorithm for multi-layer obstacle-avoiding rectilinear Steiner tree construction. In: Proceeding of the 49th ACM/IEEE Design Automation Conference. San Francisco, USA: IEEE Press, 2012. 613-622.

[133] Chuang J R, Lin J M. Efficient multi-layer obstacle-avoiding preferred direction rectilinear Steiner tree construction. In: Proceeding of the 16th Asia and South Pacific Design Automation Conference. Piscataway, USA: IEEE Press, 2011. 527-532.

[134] Lin Y C, Chien H A, Shih C C, et al. A multi-layer obstacles-avoiding router using X-architecture. WSEAS Transactions on Circuits and Systems, 2008, 7(8): 879-888.

[135] 郭文忠, 陈国龙, 夏添. 异构机群下数据流自适应分配策略. 计算机辅助设计与图形学学报, 2009, 21(8): 1175-1181.

[136] 郭文忠, 陈国龙, XIONG Naixue, 等. 求解 VLSI 电路划分问题的混合粒子群优化算法. 软件学报, 2011, 22(5): 833-842.

[137] Ao J, Dong S, Chen S, et al. Delay-driven layer assignment in global routing under multi-tier interconnect structure. In: Proceedings of the 2013 ACM International Symposium on International Symposium on Physical Design. New York, NY, USA: ACM Press, 2013. 101-107.

[138] Cho M, Pan D Z. BoxRouter: A new global router based on box expansion and progressive ILP. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2007, 26(12): 2130-2143.

[139] Albrecht C. Global routing by new approximation algorithms for multicommodity flow. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2001, 20(5): 622-632.

[140] Hu J, Roy J A, Markov I L. Sidewinder: A scalable ILP-based router. In: Proceedings of the 2008 International Workshop on System Level Interconnect Prediction. New York, NY, USA: ACM Press, 2008. 73-80.

[141] Vannelli A. An interior point method for solving the global routing problem. In: Proceedings of the 1989 IEEE Custom Integrated Circuits Conference. San Diego, CA, USA: IEEE Press, 1989. 1-4.

[142] Behjat L, Chiang A, Rakai L, et al. An effective congestion-based integer programming model for VLSI global routing. In: Proceedings of the 2008 Canadian Conference on Electrical and Computer Engineering. Niagara Falls, ON, USA: IEEE Press, 2008. 931-936.

[143] Behjat L, Vannelli A, Rosehart W. Integer linear programming models for global routing. INFORMS Journal on Computing, 2006, 18(2): 137-150.

[144] Dong J, Zhu H L, Xie M, et al. Graph Steiner tree construction and its routing applications. In: Proceedings of the IEEE 10th International Conference on ASIC. Shenzhen, China: IEEE Press, 2013. 1-4.

[145] Hung J H, Yeh Y K, Lin Y C, et al. ECO-aware obstacle-avoiding routing tree algorithm. WSEAS Transactions on Circuits and Systems, 2010, 9(9): 567-576.

[146] Guo W Z, Liu G G, Chen G L, et al. A hybrid multi-objective PSO algorithm with local search strategy for VLSI partitioning. Frontiers of Computer Science, 2014, 8(2): 203-216.

[147] Qian C, Yu Y, Zhou Z. An analysis on recombination in multi-objective evolutionary optimization. Artificial Intelligence, 2013, 204: 99-119.

[148] Alpert C, Tellez G. The importance of routing congestion analysis. DAC Knowledge Center Online Article, 2010. [Online]. Available: http://www.dac.com/back_end+topics.aspx?article=47&topic=2.

[149] ISPD 1998 Global Routing Benchmark Suite, 1998. [Online]. Available: http://cseweb.ucsd.edu/kastner/research/labyrinth/.

[150] Labyrinth, 2004. [Online]. Available: http://www.ece.ucsb.edu/ kastner/labyrinth/.

致谢

值此本文完成之际，谨向指导、关心和帮助过我的所有老师、朋友和亲人表示衷心的感谢！

衷心感谢我的导师陈国龙教授和郭文忠教授。本课题的研究始终是在陈老师的指导和关心下进行，得到了他的许多重要的建议和决策。陈老师以他渊博的知识、敏锐的洞察力、严密的逻辑思维将我带进浩瀚的学术海洋，为我创造了良好的学术环境和宽松的学习气氛，使我博士得以顺利毕业。陈老师在学风上的严谨求实态度以及高尚的学术道德使我深受教育。

郭文忠教授孜孜不倦的治学态度和高尚的学术道德给了我莫大的启迪和帮助。在科学研究的过程中，他不仅向我传授了做学问的秘诀，还传授了做人的准则，这些都将使我终生受益匪浅。在攻读博士期间，在我的课题方案选择和具体实现过程中，郭老师都提出了许多指导性的建议，并给予了许多帮助和支持。在论文完成之际，谨向陈老师和郭老师表达我最诚挚的谢意!

感谢福建省网络计算与智能信息处理重点实验室的牛玉贞、陈羽中、程红举、刘延华、钟尚平、廖祥文等老师，谢谢你们对我研究工作的指导和帮助。感谢福建省网络计算与智能信息处理重点实验室的同门们，谢谢你们在学习、生活等方面无私的帮助。

感谢我的父亲和母亲，感谢他们对我的哺育、教诲和无私无尽的爱。

感谢我的妻子，正是有了她在精神上的鼓励和在生活上的关心，使我能够克服在生活、学习和研究中的挫折和困难，顺利完成了课题任务。

最后，请允许我向参与本论文的预答辩、评审以及答辩的各位老师表示深深的敬意，感谢各位老师在百忙之中抽出宝贵的时间对我的论文进行评审，您们的关心和指教使我获益颇多，期望在今后的工作中能够继续得到您们的教诲和帮助。

个人简历

姓名： 刘耿耿　　出生年月： 1988 年 1 月

性别： 男　　籍　　贯： 福建南安

学习经历：

2005.09-2009.07　福州大学网络工程专业　　本科　　获工学学士学位

2009.09--至今　　福州大学应用数学专业　　研究生　　攻读博士学位

工作经历：

2009.09-2011.01　福州大学数学与计算机科学学院学生会副主席

在学期间的研究成果及发表的学术论文

第一作者

期刊论文(5 篇)

1. Genggeng Liu, Xing Huang, Wenzhong Guo, Yuzhen Niu, and Guolong Chen. Multilayer Obstacle-Avoiding X-Architecture Steiner Minimal Tree Construction Based on Particle Swarm Optimization. ***IEEE Transactions on Cybernetics.*** (**SCI** Sources, DOI: 10.1109/TCYB.2014. 2342713)
2. Genggeng Liu, Wenzhong Guo, Yuzhen Niu, Guolong Chen, Xing Huang. A PSO-based timing-driven Octilinear Steiner tree algorithm for VLSI routing considering bend reduction. ***Soft Computing, Springer.*** (**SCI** Sources, DOI:10.1007/s00500-014-1329-2)
3. Genggeng Liu, Wenzhong Guo, Rongrong Li, Yuzhen Niu, Guolong Chen. XGRouter: High-Quality Global Router in X-Architecture with Particle Swarm Optimization. ***Frontiers of Computer Science***. (**SCI** Sources, DOI: 10.1007/s11704-015-4107-1)
4. 刘耿耿, 郭文忠, 陈国龙. X 结构下 VLSI 多层绕障 Steiner 最小树算法. *计算机辅助设计与图形学学报*. (Accepted, **EI**)
5. 刘耿耿, 王小溪, 陈国龙, 郭文忠, 王少铃. 求解 VLSI 布线问题的离散粒子群优化算法.*计算机科学*, 2010, 37(10):197-201.

国际会议论文(2 篇)

6. Liu Genggeng, Chen Guolong, Guo Wenzhong. DPSO based octagonal steiner tree algorithm for VLSI routing. ***In: 2012 IEEE fifth international conference on advanced computational intellligence***, Nanjing, China. 2012, 383-387. **(EI/ISTP** Sources**)**
7. Liu Genggeng, Chen Guolong, Guo Wenzhong, Chen Zhen. DPSO-based rectilinear Steiner minimal tree construction considering bend reduction. ***In: 2011 applicableSeventh International Conference on Natural Computation***. 2011, 1161-1165. **(EI/ISTP** Sources**)**

第二作者

期刊论文(2 篇)

8. Guo Wenzhong, Liu Genggeng, Chen Guolong, Peng Shaojun. A hybrid multi-objective PSO algorithm with local search strategy for VLSI partitioning. ***Frontiers of Computer Science***, 2014, 8(2): 203-216.(**SCI** Sources)
9. Huang Xing, Liu Genggeng, Guo Wenzhong, Chen Guolong. Obstacle-Avoiding Algorithm in X-Architecture Based on Discrete Particle Swarm Optimization for VLSI Routing. ***ACM Transactions on Design Automation of Electronic Systems***. Accepted, **SCI** Sources)

国际会议论文(1 篇)

10. Huang Xing, Liu Genggeng, Guo Wenzhong, Chen Guolong. Obstacle-avoiding Octagonal Steiner tree construction based on particle swarm ptimization. ***In: Proceeding of the 9th international conference on natural computation***. Shenyang, China. 2013: 539-543. **(EI/ISTP** Sources**)**

发 明 专 利

1. X 结构下超大规模集成电路总体布线方法(专利号：201410123885.4)
2. 超大规模集成电路多层绕障 Steiner 最小树构造方法(专利号：201410124000.2)
3. 八角结构 Steiner 最小树下的 VLSI 绕障布线器(专利号：201410589725.9)

攻读博士学位期间从事的科学研究和获奖情况

博士期间参与的科研项目：

- 非曼哈顿结构下 VLSI 多层总体布线算法研究(编号：11271002)，*国家自然科学基金项目，*2013.1-2016.12
- 非曼哈顿结构下带粒子群优化的 VLSI 总体布线算法研究(编号：11141005), *国家自然科学基金项目,* 2012.1-2012.12
- 超大规模集成电路物理设计中多目标优化问题的粒子群优化算法研究(编号：10871221)，*国家自然科学基金项目*，2009.1-2011.12

获奖情况：

- 2013 年度福州大学博士生专项助研奖学金
- 2014 年度福州大学博士生专项助研奖学金
- 2014 年度研究生国家奖学金