计 算 机 研 究 与 发 展
Journal of Computer Research and Development
https://crad.ict.ac.cn
62(8): 1948−1961, 2025

一种基于带权有向图的印刷电路板群组布线算法

邓新国[1] 张鑫泓[1] 陈家瑞[1] 刘清海[2,3] 陈传东[4]
[1](福州大学计算机与大数据学院 福州 350108)
[2](福州大学离散数学中心 福州 350108)
[3](中国福建光电信息科学与技术创新实验室 福州 350024)
[4](福州大学物理与信息工程学院 福州 350108)
(xgdeng@fzu.edu.cn)

A Weighted Directed Graph-Based Algorithm for Group Routing in Printed Circuit Boards

Deng Xinguo[1], Zhang Xinhong[1], Chen Jiarui[1], Liu Qinghai[2,3], and Chen Chuandong[4]
[1](*College of Computer and Data Science*, *Fuzhou University*, *Fuzhou* 350108)
[2](*Center for Discrete Mathematics*, *Fuzhou University*, *Fuzhou* 350108)
[3](*Fujian Science & Technology Innovation Laboratory for Optoelectronic Information of China*, *Fuzhou* 350024)
[4](*College of Physics and Information Engineering*, *Fuzhou University*, *Fuzhou* 350108)

Abstract Routing is considered as an essential component in the design of printed circuit board (PCB). Current existing PCB designs mostly rely on the process results from electronic design automation tools, and traditional automatic routing research often focuses on the only general bus routing without considering bus groups to be determined during the routing process. Due to the absence of general bus grouping, there may be situations where there are more nets in one group than in the other groups, resulting in larger line width and line clearance occupied by this group when compared with other bus groups in the original bus routing, thereby posing new challenges to effective and efficient routing. To overcome this drawback, we focus on studying PCB group routing. In this study, a group routing algorithm based on a weighted directed graph is proposed. A Hanan grid graph is constructed, containing the only merged edges and their adjacent relationships. Following this, a weighted directed graph is developed using the merged edge information to represent the routing resources on the circuit board. For routing planning, a heuristic search algorithm equipped with multi-wire avoidance features is utilized. The routing situations are then classified into several potential scenarios, with each considered separately, to accomplish detailed routing and obtain a final result of group routing. Results from experiments demonstrate that 100% routability is consistently achieved by using the algorithm on complex industrial examples that have been previously tested, and that the design rule constraints of all benchmark industrial PCB cases are not violated.

Key words electronic design automation (EDA); printed circuit board; group routing; weighted directed graph; path planning

摘 要 布线是印刷电路板设计中的重要一环.现有的印刷电路板设计多依赖于电子设计自动化工具的处

收稿日期: 2024-02-04; **修回日期**: 2025-02-07
基金项目: 国家自然科学基金项目(61977017); 国家重点研发计划(2021ZR142)
This work was supported by the National Natural Science Foundation of China (61977017) and the National Key Research and Development Program (2021ZR142).
通信作者: 陈家瑞(jrchen@fzu.edu.cn)

理，而传统的自动布线研究多聚焦于总线布线，没有将布线时确定的群组作为研究对象. 由于未经总线分组，可能存在群组中线网较多的情况，这将导致群组所占据的线宽与线间距比原先总线布线中各总线组分别占据的线宽与线间距更大，从而给实际布线带来了新的挑战. 为此，提出一种基于带权有向图的群组布线算法. 首先构建仅含有合并边以及它们之间邻接关系的 Hanan 网格图. 接着，利用合并边信息构建带权有向图，完成对电路板上布线资源的表示. 然后，使用一种具有多线避让功能的启发式搜索算法来进行布线规划. 最后，通过将布线归类为数种可能的情况分别考虑，完成详细布线并得到群组布线的最终结果. 实验结果表明，所提算法在已经测试过的工业界复杂例子上均能达到 100% 的布通率，并且不会违反所有工业印刷电路板基准用例的设计规则约束.

关键词 电子设计自动化；印刷电路板；群组布线；带权有向图；路径规划

中图法分类号 TP312；TP391；TP399

DOI: 10.7544/issn1000-1239.202440069 **CSTR:** 32373.14.issn1000-1239.202440069

电子设备广泛应用于与国家基础设施和民众生活紧密相关的各项领域. 在这些设备中，印刷电路板（printed circuit board, PCB）被誉为“电子产品之母”. PCB 不仅为电子元器件提供了电子元器件的支持和电气连接，还承担着电路的载体角色，是电子设备不可或缺的核心组成部分[1].

当前，随着集成电路技术的迅猛发展，PCB 的复杂度也在显著增加. 这种复杂度体现在多个方面：不断增加的线网（net）和引脚（pin）数量、多达十数层的 PCB 层数、极高的引脚密度、独特的物理限制以及高速 PCB 设计所带来的信号完整性问题[1] 等，这就使得无法像过去那样仅仅使用人工对设计规则予以逐一检查，因此，以往那种借由个人经验进行的 PCB 手工布线成为了一项耗时且容易出错的工作[2]，传统的设计流程变得不再适用[3-5]. 有鉴于此，复杂的 PCB 设计现多依赖于电子设计自动化（electronic design automation, EDA）工具处理以实现布线效率的优化，以最大限度地满足电子设计周期的需求[6-8]. 利用 EDA 进行的 PCB 设计，通常包括原理图绘制、结构设计、布局、布线等流程[9]，其中布线问题是最为关键的问题之一[10-11].

在具有复杂线网的 PCB 电路布线问题中，群组布线问题尤为重要. 此问题涉及将人工指定的多个线网视作群组，继而将它们以彼此相邻、长度相近的布线方式加以连接. 如何在不违反设计规则约束（design rule constraints, DRC）的条件下有效利用有限的布线空间并提高布线成功率是群组布线需要研究的关键问题.

本文主要贡献有 3 点：

1）在使用 Hanan 网格方法构图后，在构图过程中引入一种面合并算法以及合并边方法以缩减原始 Hanan 网格的规模，在有效减少后续构建带权有向图的复杂度的同时，解决了传统方法得到的网格构图通常有一定量冗余信息的问题.

2）提出一种基于带权有向图的群组布线算法，利用得到的合并边及其位置关系等信息构建带权有向图，实现对群组布线问题的数学建模，从而克服传统网格化方法易受网格精度以及网格数量限制的缺陷.

3）提出一种与路径规划算法相适配的详细布线算法，将详细布线问题分解为多种基本情形考虑，设计数种基本线形以应对各种可能的布线情形，使算法具备实际应用价值，避免详细布线算法与路径规划算法不适配的问题.

1 相关工作

PCB 布线问题通常可被划为逃逸布线（escape routing）与区域布线（area routing）两个阶段[12]，逃逸布线阶段主要负责将线网从引脚布线至组件边界，区域布线阶段则负责对逃逸至组件边界的线网进行连接，而群组布线一般在区域布线阶段进行. 群组布线问题就具体场景而言，又可简化为若干子问题，典型的有总线布线. 总线布线与群组布线的区别在于，它会在布线前预先进行分组，将预计需要一起布线的线网划分为一组总线[13]，而群组布线可以在布线时确定群组，也可以进行交互式布线，具有较大的灵活性. 印刷电路板群组布线是商业布线工具中重要的组成部分，然而，关于印刷电路板群组布线问题的相关研究却鲜少出现在近年来的公开文献当中，因而对该问题的研究只能参考其子问题——总线布线. 对于总线布线而言，其中一种分类依据是根据布线时

有无轨道约束其被划分为有轨道总线布线问题和无轨道总线布线问题. 由于未经总线分组, 可能存在群组中线网较多的情况, 这将导致群组中线网所占用的总线宽比原先总线布线中各总线组分别占据的线宽更大, 从而给实际布线带来新的挑战.

有些研究不针对特定阶段的布线问题, 而是专注于更一般的印刷电路板自动布线问题. 如 Lin 等人[3]提出了一种具有并发分层布线功能的全阶段印刷电路板布线方法, 该方法基于布尔可满足性问题(Boolean satisfiability problem, SAT), 但是该方法只适用于解决球栅阵列(ball grid array, BGA)封装的印刷电路板布线问题[14]. Lin 等人[4]则提出了一种全板布线算法, 在构图阶段使用了均分网格, 可以对非 BGA 封装进行布线, 并且可以进行多层差分对布线. 但是该方法在复杂情况下的布通率难以达到 100%[14]. He 等人[15]提出了一种两阶段印刷电路板布线算法, 该算法的 2 个阶段分别为基于蒙特卡洛树搜索[16](Monte Carlo tree search, MCTS)的全局布线和基于 A*的详细布线, 能够进行 BGA 和非 BGA 封装布线. 但是该方法是一种布线仿真方法, 所使用的例子为简化后的现实电路, 是一种理想化的场景, 因而无法直接应用于复杂的工业界样例[17].

印刷电路板有轨道总线布线问题只允许在给定的布线轨道上进行布线. 对于这一问题, Hung 等人[18]提出了一种基于有向无环图(directed acyclic graph, DAG)的总线布线算法. 该算法利用最长公共子序列(longest common subsequence, LCS)、Dijkstra 算法以及拆线重布的方法优化布线. 然而, 问题在于, 可能会产生大量的间距违规, 并且在初始拥塞区域过多时, 无法有效消除拥塞[19]. Zhang 等人[20]提出了一种基于迷宫布线和 SAT 的总线布线算法. 基于多级框架的总线布线器 MARCH[21]是一种基于并发和多层次方案的总线迷宫布线方法, 它由粗粒度的拓扑感知路径规划和细粒度的轨道分配组成, 应用了有效的拆线布线方案. Kim 等人[22]提出了一种拓扑感知总线布线算法, 使用专门针对总线级别的 3D 平铺网格进行布线拓扑生成. 但是这些算法大都聚焦于有轨道总线布线问题, 具有一定的特殊性, 通常难以扩展到更加一般的无轨道约束总线布线问题.

近年来, 也有几种算法被提出以解决无轨道约束的某些总线布线问题, 它们通常需要依赖网格来进行路径规划和实际布线. 其中一些算法事先预置均分网格. Cheng 等人[23]提出了一种考虑障碍物的长度匹配总线布线算法, 该算法利用先深度优先搜索后广度优先搜索的思想进行布线, 并在流程中加入了拆线重布环节. Yan 等人[24-25]提出了基于长度匹配的总线布线算法, 利用带权无向图表示区域之间的布线资源, 并使用最大流算法寻找线网匹配. 上述算法将特殊设计约束即长度匹配约束作为考虑重点.

另一些算法则构建 Hanan 网格. Kong 等人[26]通过构建冲突图和更新 Hanan 网格来规划路径, 使用关键割减少拥塞. Wu 等人[27]采用整数线性规划(integer linear programming, ILP)方法, 在动态的 Hanan 网格图上完成全局总线布线, 该算法同样利用关键割方法解决拥塞问题. ILP 方法可以提供总线布线的全局视图, 但可能需要相当长的时间[28].

为解决群组布线问题, 本文提出一种使用面合并算法缩减 Hanan 网格规模, 利用基于带权有向图的群组布线算法, 在有效表示电路板上的布线资源的同时避免传统网格方法易受网格精度和数量限制的缺陷. 同时采用具有多线避让功能的启发式搜索算法, 该搜索算法在自动布线的同时也支持人工引导布线方向, 这在有效降低时间开销的同时保证了布线质量. 最后该算法使用了一种适用于所提出的路径规划算法的详细布线算法, 具有较高的运行效率.

2 问题描述

首先输入一组需要进行群组布线的双端线网(net)$\mathcal{N}$, 这些线网所包含的一组引脚(pin)$\mathcal{P}$, 还有印刷电路板上的一系列障碍物如过孔(via)、禁行区等以及印刷电路板的分层信息, 此外还有一系列设计规则约束等. 给定双端线网的每对引脚必须在不违反设计规则约束的前提下相互连接. 印刷电路板群组布线问题需要遵循的设计规则约束如下:

1)由 2 点确定的最短轨迹即最小线长必须大于等于允许的最小线宽;

2)走线必须满足线与线之间、线与各种障碍物之间的最小线间距;

3)同层的走线之间不允许存在交叉;

4)走线拐角角度必须是 135°且走线方向必须与印刷电路板板的 x 轴呈 45°的整数倍;

5)所有引脚的走线尽量汇聚;

6)所有引脚的走线拓扑尽量相似;

7)所有引脚的走线长度尽量相近.

该问题的目标是在不违反设计规则约束的前提下, 最大化连接的线网对数, 即布通率应尽可能的高; 在此基础上, 布线时间应尽可能的短.

3 群组布线算法

群组布线算法流程如图 1 所示. 首先, 构建 Hanan 网格图, 并在构建完成后对它进行边合并. 接着, 利用前述得到的合并边信息构建带权有向图, 完成对电路板上布线资源的表示. 然后, 利用一种具有多线避让功能的启发式搜索算法来进行布线的边序列搜索, 继而进行路径规划. 最后, 通过将布线归类为数种可能的情况分别考虑, 完成详细布线并得到群组布线的最终结果. 在详细布线步骤前, 若线网未规划完毕, 就会重新进行启发式搜索, 直到完成所有失败线网的路径规划为止.

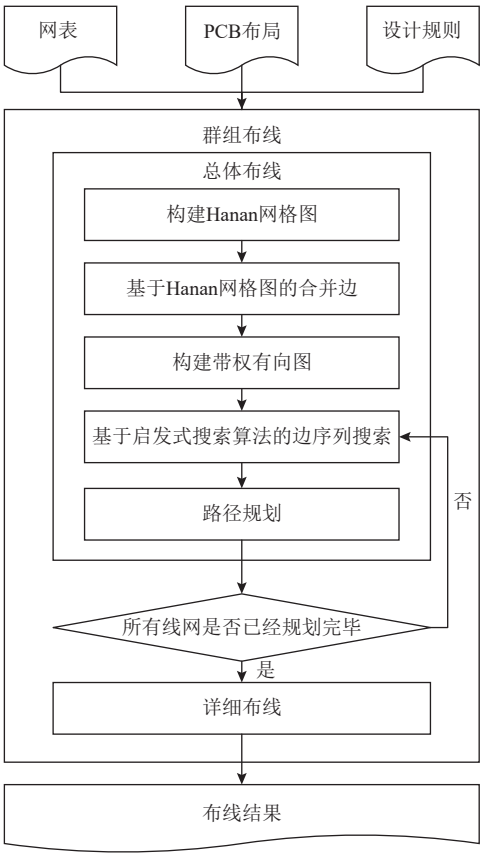


Fig. 1 Flow chart of grouping routing algorithm

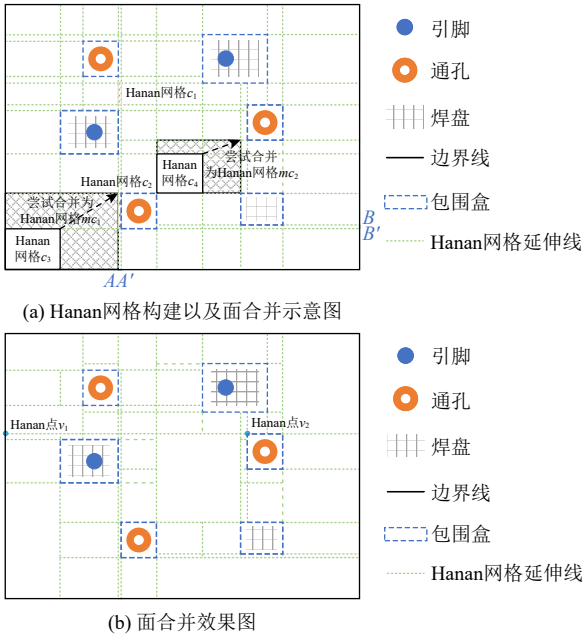
图 1 群组布线算法流程图

3.1 构建 Hanan 网格图

为了能够有效记录 PCB 中的各种布线资源, 例如各器件、各障碍物的大小和坐标, 以及它们之间的相互位置关系等, 传统的做法是将它们以网格图的方式加以表示. 典型的网格图有 Hanan 网格图. 在几何学中, 对平面中有限点的集合构造 Hanan 网格将会对集合中的每个点构造横线以及竖线[29]. 而在实际应用中, Hanan 网格在构图时会对集中的每个点进行横向或是纵向延伸, 直到延伸线碰触到其他器件的边界抑或是 PCB 板边界为止. 器件如焊盘、通孔的边界会在考虑器件与走线之间的最小间距之后以矩形包围盒的方式予以体现. 随后, 在构图结束时, 该方法便能够得到顶点、边以及网格之间的邻接关系.

3.2 基于 Hanan 网格图的面合并以及合并边

Hanan 网格图, 构建示意图如图 2(a)所示. 从示意图中不难看出, Hanan 网格构图存在有易受网格精度影响的缺陷. 以 Hanan 网格 c_1 也即面 c_1 为例, 显然当走线线宽超过该网格的宽度时, 走线便无法从竖直方向顺利通过, 因而对布线资源造成了浪费. 更糟糕的是, 当网格的划分越稠密时, 这一问题便会越明显, 在极端情况下甚至可能存在无法布线的情况. 有鉴于此, 本文提出了构造基于 Hanan 网格图的面合并以及合并边算法以克服这一缺陷.



(a) Hanan网格构建以及面合并示意图

(b) 面合并效果图

Fig. 2 Schematic diagram of grid construction

图 2 网格构建示意图

为了有效缩减问题规模, 需要对 Hanan 网格图进行面合并. 算法首先需要确定面合并的顺序, 不失一般性, 算法按从左到右, 自下而上的顺序进行面合并. 假设整体布线方向为水平方向. 以下给出面合并算法, 算法从对角线、纵向以及横向 3 个方向尝试进行面合并, 在面合并遇到障碍物或是相邻面存在宽度小于线宽与线间距之和时停止.

面合并算法步骤如算法 1 所示.

算法 1. 面合并算法.

输入: Hanan 网格图 H;

输出：面合并后的 Hanan 网格图H'.

① 设图H中所有面的集合为F，图H中所有障碍物包围盒的边集合为Ob，由面的左下角坐标组成的最小优先队列$Q\neq\varnothing$，ℓ=1；
② while $\ell\leqslant3$
③ for $f\in F$
④ if 当前面f的任意一条边e与Ob中的所有边$obst$都不存在相交 then
⑤ $Q\leftarrow Q.push(\langle f_{\mathrm{lbx}},f_{\mathrm{lby}}\rangle)$；
⑥ end if
⑦ end for
⑧ while $Q\neq\varnothing$
⑨ 选择Q中队首元素对应的面f，面f的左下角坐标为$f_{\mathrm{lb}}\langle f_{\mathrm{lbx}},f_{\mathrm{lby}}\rangle$，右上角坐标为$f_{\mathrm{rt}}\langle f_{\mathrm{rtx}},f_{\mathrm{rty}}\rangle$，它们之间构成区域面积；
⑩ end while
⑪ while 当前区域面积内的面均存在于Q中
⑫ if 当前区域的相邻面的长或宽不大于线宽与线间距之和 then
⑬ break；
⑭ else
⑮ if ℓ=1 then
⑯ 根据图H的邻接关系寻找位区域面积斜右上方的面f'_{rt}，该面左下角顶点与区域面积右上方顶点重合. 令f'_{rt}的右上角坐标为$f'_{\mathrm{rt}}\langle f'_{\mathrm{rtx}},f'_{\mathrm{rty}}\rangle$，若$f_{\mathrm{lb}}$与$f'_{\mathrm{rt}}$围成的区域中的面均存在于$Q$中，则$f_{\mathrm{lb}}$与$f'_{\mathrm{rt}}$之间构成区域面积；
⑰ end if
⑱ if ℓ=2 then
⑲ 根据图H的邻接关系寻找与区域面积正上方的面f'_{top}，该面右下角顶点与区域面积右上方顶点重合. 令f'_{top}的右上角坐标为$f'_{\mathrm{top}}\langle f'_{\mathrm{rtx}},f'_{\mathrm{rty}}\rangle$，若$f_{\mathrm{lb}}$与$f'_{\mathrm{rt}}$围成的区域中的面均存在于$Q$中，则$f_{\mathrm{lb}}$与$f'_{\mathrm{rt}}$之间构成区域面积；
⑳ end if
㉑ if ℓ=3 then
㉒ 根据图H的邻接关系寻找与面积正右方的面f'_{right}，该面左上角顶点与区域面积右上方顶点重合，令f'_{right}的右上角坐标为$f'_{\mathrm{right}}\langle f'_{\mathrm{rtx}},f'_{\mathrm{rty}}\rangle$，若$f_{\mathrm{lb}}$与$f'_{\mathrm{rt}}$围成的区域中的面均存在于$Q$中，则$f_{\mathrm{lb}}$与$f'_{\mathrm{rt}}$之间构成区域面积；
㉓ end if
㉔ end while
㉕ 让包含在区域面积中的面f从Q中出队；
㉖ if（区域包含的面f的个数）>1 then
㉗ 根据区域面积更新集合F，将区域作为合并面，更新图H中区域包含的所有面的邻接关系；
㉘ end if
㉙ ℓ++；
㉚ end while
㉛ return 合并面后的 Hanan 网格图H'.

算法 1 行②的循环在常数项时间内可以完成. 在行③~⑦使用了 1 次循环，时间复杂度为$O(n)$. 而算法的主要时间花费则集中在行⑧~㉘中所包含的步骤，它包含了 2 次循环，最坏时间复杂度为$O(n^2)$. 综上，算法的整体时间复杂度为$O(n^2)$.

当整体布线方向为竖直方向时，只需将算法 1 中的行⑱和行㉑交换即可. 假设图 2(a)中线段AA'以及线段BB'的长度小于线宽与线间距之和，其他部分除非长度与这些线段相等，否则长度均大于线宽与线间距之和. 根据算法 1，首先在图 2(a)中选择位于左下角的 Hanan 网格c_3，继而沿对角线向右上方进行扩展. 由于扩展后得到的合并面mc_1，其相邻面c_2的宽度小于线宽与线间距之和，因此此次合并无效. 而在扫描至 Hanan 网格c_4时，根据算法将其扩展至合并面mc_2，在检查后发现此次合并有效，因此合并面mc_2便能够得到保留.

而传统的 Hanan 网格构图，只能在构图结束后得到合并边中的数段短边，以图 2(b)为例，在 Hanan 点v_1与v_2之间，它仅仅能得到其中的 8 段短边，这将对后续的布线规划造成困难. 有鉴于此，本文提出了一种边合并方法. 该方法将障碍物包围盒或者边界线都视为障碍物线，以图 2(b)为例，它能够得到以 Hanan 点v_1，v_2为端点的一整条合并边. 该方法的具体步骤为：首先，扫描所有 Hanan 边，之后，对位于障碍物线之间且具有相同横坐标或纵坐标的线段进行合并，所得到的边即为合并边. 在得到合并边之后，将位于障碍物线之间且具有相同横坐标或纵坐标的线段予以删除. 重复这一过程，便能够得到全部合并边.

3.3 构造带权有向图

这一阶段主要是利用合并边构造带权有向图的算法，将合并边的邻接关系以及通道容量大小保存至有向图中.

算法将所有合并边按照一定顺序进行排序，不失一般性，可将竖直边按照竖直坐标从小到大排序，水平边同理. 随后，算法将排序后的每一条合并边都抽象为有向图G中的 2 个顶点$v_{i_1}^{dir_1}$，$v_{i_2}^{dir_2}$，以水平边为例，它表示了布线的 2 个方向，即走线从上往下穿过该边(顶点的方向 dir 记作 up)以及走线从下往上穿过该边(顶点的方向 dir 记作 down)，竖直边同理(顶点的方向 dir 分别记作 left, right). 所有顶点构成顶点的集合$V(G)$.

随后，算法从顶点集合$V(G)$中按照排序后的顺序取出顶点对$v_i^{dir_i}$，$v_j^{dir_j}$，并尝试在它们之间构造边e_{ij}. 它们所对应的合并边记作HE_{v_i}及HE_{v_j}在图例中HE_{v_1}简化为l_1，以此类推. 构造时遵循以下原则：

1)若HE_{v_i}和HE_{v_j}相互平行，则有：

dir_i和dir_j方向相反的顶点对之间不能构造边. 图 3(a)所示，走线显然无法在从上往下穿过HE_{v_1}的同时再从下往上穿过HE_{v_2}，因此顶点v_1^{up}和v_2^{down}之间无法构造边.

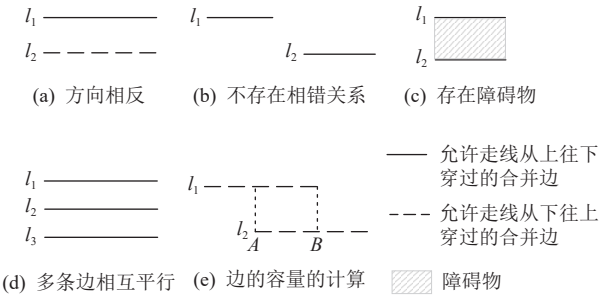


Fig. 3 Several cases of constructing edges when vertices correspond to merged edges are parallel to each other

图 3 顶点对应合并边相互平行时构造边的几种情形

HE_{v_i}及HE_{v_j}之间若不存在相错关系，则它们之间不能构造边. 如图 3(b)所示，走线在穿过HE_{v_1}和HE_{v_2}的过程中，必然会经过其他顶点对应的合并边(图中未直接画出)，因此顶点v_1^{up}和v_2^{up}之间无法构造边.

HE_{v_i}及HE_{v_j}之间若有障碍物，则它们之间不能构造边. 示意图如图 3(c)所示.

若$v_i^{dir_i}$或$v_j^{dir_j}$已经存在相互平行的连边，则不能构造边. 如图 3(d)所示，不妨设此时坐标已经按照y轴坐标从大到小排序(从小到大排序同理)，则v_1^{up}和v_2^{up}将先行构造边，该边记作e_{12}. 显然此时在v_1^{up}和v_3^{up}之间便无法再构造边，因为走线在通过v_1^{up}之后，必须先经过v_2^{up}才能到达v_3^{up}.

边的容量$cap(e_{ij})$按HE_{v_i}与HE_{v_j}之间投影的交叠长度计算. 如图 3(e)所示，HE_{v_1}在合并边HE_{v_2}上的投影为线段AB，线段AB的长度即为边的容量$cap(e_{ij})$.

2)若HE_{v_i}和HE_{v_j}相互垂直，则有：

HE_{v_i}及HE_{v_j}之间若不存在交叉关系，则它们之间不能构造边. 图 4(a)所示，走线在穿过HE_{v_1}和HE_{v_2}的过程中，必然会经过其他顶点对应的合并边(图 4(a)未直接画出)，因此顶点v_1^{up}和v_2^{left}之间无法构造边.

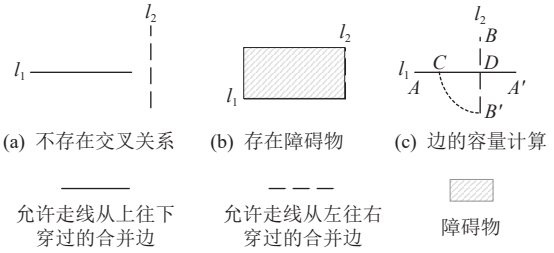


Fig. 4 Several cases of constructing edges when vertices correspond to merging edges are perpendicular to each other

图 4 顶点对应合并边相互垂直时构造边的几种情形

HE_{v_i}及HE_{v_j}之间若存在障碍物则不能构造边，示意图如图 4(b)所示.

HE_{v_i}及HE_{v_j}之间若相交，则交叉点将线段HE_{v_i}及HE_{v_j}分割为 4 个部分，分别记做$HE_{v_{i1}}$，$HE_{v_{i2}}$以及$HE_{v_{j1}}$，$HE_{v_{j2}}$. 每条走线每次有且只能通过其中 2 条线段所围成的区域. 不失一般性，不妨设走线可能通过的其中 2 条线段为$HE_{v_{i1}}$以及$HE_{v_{j1}}$. 边的容量$cap(e_{ij})$按$HE_{v_{i1}}$和$HE_{v_{j1}}$长度的最小值计算. 如图 4(c)所示，交叉点D将线段AA'以及线段BB'分割为AD, DA', BD, DB'，根据合并边所允许走线的通过方向不难得出，走线将会自上往下穿过线段AD继而从左向右穿过线段DB'，此时便能够得到边的容量$cap(e_{ij})$即是线段AD以及线段DB'两者长度中的较小者.

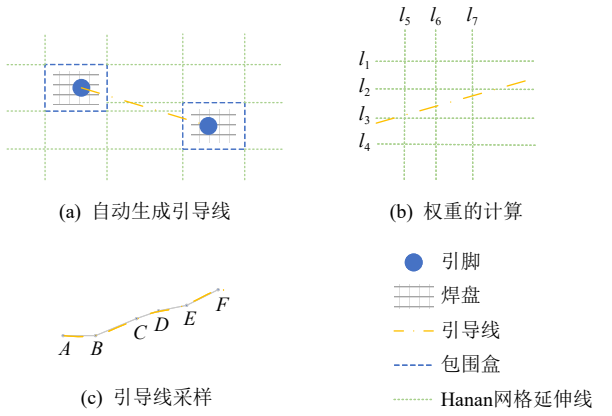
为了有效缩减问题规模，算法在实际构造边时进行了一定简化. 首先根据起始点及终点器件的坐标分布，确定整体布线方向. 以整体布线方向从左到右为例，在实际构造带权有向图时可以省略构造表示，允许走线从右往左穿过的合并边的顶点. 其他方向同理.

为了实现群组布线的聚集效果，在自动布线流程中引入了一种自动生成引导线的方法. 所谓的引导线是一条曲线，在实际布线过程中，算法希望所有走线的路径都能够尽可能地贴近引导线. 在通常的 PCB 设计中，起始点及终点都是相互分离的. 因此，可以计算所有器件起始和终点引脚中心点的均值坐标，如下所示：

$$X=\frac{1}{n}\sum_{i=1}^{n}x_i, \tag{1}$$

$$Y=\frac{1}{n}\sum_{i=1}^{n}y_i. \tag{2}$$

然后, 便能够利用这 2 个坐标确定一条引导线. 如图 5(a)所示, 算法在双端线网的 2 个引脚之间创建了一条引导线, 它是连接 2 个引脚中心的一条直线.



(a) 自动生成引导线 (b) 权重的计算

(c) 引导线采样

Fig. 5 Schematic diagram of guidelines

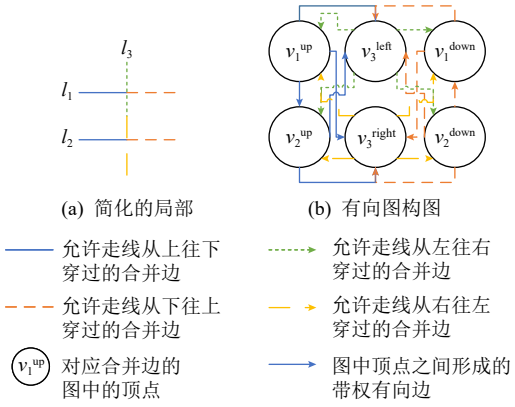
图 5 引导线示意图

对于合并边的权重, 做出如下规定: 1)与引导线相交的合并边的权重规定为 0, 以图 5(b)为例, 合并边 HE_{v2}, HE_{v3}, HE_{v5}, HE_{v6}, HE_{v7}的权重 I_2, I_3, I_5, I_6, I_7 均被设置为 0; 2)若合并边与引导线不存在交点, 如图 5(b)中的 I_1, I_4, 则计算它们之间的最小距离, 且如果这一最小距离大于设置的阈值 T, 则将该合并边视作障碍物, 这将有效加速之后进行的搜索过程. 由此, 如果算法能够在顶点 $v_i^{dir_i}$ 与 $v_j^{dir_j}$ 之间构造边 e_{ij}, 则令边 e_{ij} 的权重 $weight(e_{ij})$ 为合并边 HE_{vi} 和 HE_{vj} 的权重之和.

为了能够让群组布线的结果灵活适应各种复杂的工业化需求, 达到人工预期的良好效果, 算法同样支持人工指定引导线. 由于人工指定的引导线通常是一条曲线, 因此, 实际处理时算法需要根据一定的精度, 首先对引导线进行采样, 将其转换为多段首尾相接的线段. 如图 5(c)所示, 引导线通过采样, 被重新转换为线段 AB, BC, CD, DE, EF. 线段化后的引导线的处理方式与自动生成的引导线的处理方式相同.

最后, 算法得到了所有顶点构成的集合 $V(G)$、所有边构成的集合 $E(G)$、所有边的容量以及所有边的权值. 这样, 算法便得到了基于合并边的带权有向图 G.

图 6 展示了在某个简化的局部, 完整的带权有向图 G 的构造结果. 图 6(a)中, 合并边 l_1, l_2, l_3 按照规则可以被抽象为图 6(b)所示的带权有向图 G, 其中带权有向边的容量和权重没有具体给出.



(a) 简化的局部 (b) 有向图构图

Fig. 6 Construction result of weighted directed graph G

图 6 带权有向图 G 构造结果

3.4 基于启发式搜索算法的边序列搜索

这一阶段算法将会利用启发式的搜索算法, 从图 G 中逐线逐网地寻找连接起点与终点的迹.

首先算法将环绕在起始器件周围, 且走线必然穿过的一条合并边标记为起始顶点, 同理, 算法将环绕在结束器件周围, 且走线必然穿过的一条合并边标记为结束顶点. 接着, 算法将起始顶点的权重设置为 0, 在图 G 中以 A*搜索的方式不断试探寻找下一顶点, 直到寻找到结束顶点为止. 然后, 算法对顶点序列进行逆序, 从而得到有效的顶点序列. 最后, 算法根据有效的顶点序列以及 2 个顶点确定图中一条有向边的原则, 得到有效的边序列. 在 A*搜索的过程中, 算法如下设置权重:

$$F(n)=aG(n)+bH(n)+c\times weight(e), \tag{3}$$

其中 $F(n)$ 是 A*搜索算法的总体代价, $G(n)$ 是从起始顶点到当前位置的实际路径开销[30]. $weight(e)$ 是边权重, 表示从上一个顶点到当前搜索顶点的边的权重. a, b, c 是常数, 算法可以通过改变 a, b, c 的值从而强调不同的引导效果. $G(n)$ 可以通过下列递推式计算得到:

$$G(n)_i=G(n)_{i-1}+cost_i, \tag{4}$$

其中 $G(n)_i$ 表示第 1 到第 i 段规划路径的实际开销, $cost_i$ 表示第 i 段规划路径的实际开销. $cost_i$ 在实际计算中被设置为第 $i-1$ 段规划路径的终点与当前搜索到的顶点所对应合并边中点的连线长度.

$H(n)$ 是估价函数, 它是一种启发式函数, 表示了从当前位置到终点的预估距离. 对于预估距离的计算, 通常会使用曼哈顿距离或是欧几里得距离. 设起点为 $s(x_1,y_1)$, 终点为 $t(x_2,y_2)$, 则曼哈顿距离的定义为:

$$dist_{\text{Manhattan}}(s,t)=|x_1-x_2|+|y_1-y_2|. \tag{5}$$

欧几里得距离的定义为:

$$dist_{\text{Manhattan}}(s,t)=(x_1-x_2)^2+(y_1-y_2)^2. \quad (6)$$

为加速计算过程，算法选用曼哈顿距离进行计算. 在实际计算中，$H(n)$由当前顶点所对应合并边的中点到结束顶点的曼哈顿距离得到.

算法 2 给出了与数据结构相适配的 A*搜索算法，算法从起点开始，不断地搜索出 *open* 列表中 F 值最大的顶点，直到达到终点，然后通过回溯得到有效的顶点序列，继而通过 2 点确定一条边的方式确定有效的边序列.

算法 2. A*搜索算法.

输入：图 G，起始顶点 V_{start}，结束顶点 V_{end}；

输出：由一系列不重复的边组成的集合所表示的一条迹 W.

① 设最小优先队列 $Q_{\text{OPEN}}=\varnothing$，$Q_{\text{CLOSE}}=\varnothing$，搜索过程中拜访过的顶点及其前继顶点的集合 $V_{\text{visited}}=\varnothing$，有效的顶点集合 $V=\varnothing$；

② $Q_{\text{OPEN}}.push(\langle 0,V_{\text{start}}\rangle)$；

③ while $Q_{\text{OPEN}}\neq\varnothing$

④ 选择 F 值 F_{next} 最小的 F 同时需要满足与加入的上一组序列间没有障碍物，$cap(e_{\text{next}})$ 大于线宽与线间距之和的顶点 $V_{\text{next}}\in Q_{\text{OPEN}}$，$Q_{\text{CLOSE}}.push(\langle F_{\text{next}},V_{\text{next}}\rangle)$；

⑤ $V_{\text{visited}}\leftarrow V_{\text{visited}}\cup\{V_{\text{next}}\rightarrow V_{\text{cur}}\}$；

⑥ if $V_{\text{next}}=V_{\text{end}}$ then

⑦ while $V_{\text{next}}\neq V_{\text{start}}$

⑧ $V\leftarrow V\cup V_{\text{next}}$；

⑨ 根据 V_{next} 搜索 V_{visited} 得到 $V_{\text{cur}}=V_{\text{next}}\rightarrow V_{\text{cur}}$；

⑩ $V_{\text{next}}=V_{\text{cur}}$；

⑪ end while

⑫ end if

⑬ 将 V 逆序存放 $reverse(V)$；

⑭ 设 $v_i=V_{\text{start}}$；

⑮ for $v\in V$

⑯ if $v\neq V_{\text{start}}$

⑰ 设 $v_i=v$；

⑱ 根据 v_i，v_j 搜索 G 得到边 E_{ij}；

⑲ $W\leftarrow W\cup E_{ij}$，$v_i=v_j$；

⑳ return W，找到有效路径序列；

㉑ else

㉒ 计算顶点 V_{next} 的所有邻接节点 $negis(V_{\text{next}})$ 的 F；

㉓ end if

㉔ for $neig\in neigs$

㉕ $Q_{\text{OPEN}}\leftarrow Q_{\text{OPEN}}.push(\langle neig,F_{\text{neig}}\rangle)$；

㉖ end for

㉗ end for

㉘ end while

㉙ return 未找到的有效路径序列.

在结束一次路径搜索后，需要对得到的边序列中所包含的每条边的容量进行更新. 在对所有起始顶点重复上述搜索过程后，便能得到所有线网的有效路径规划顶点序列以及有效路径规划边序列，它们分别代表了走线将会穿过的合并边的序列以及这些合并边之间的位置关系.

3.5 路径规划

这一阶段算法对路径规划得到的顶点序列以及边序列，通过限制关系在各个顶点所对应的合并边上进行取点，得到的点集就是路径规划.

除开始顶点以及结束顶点以外，算法每次取出与当前顶点邻接的前后 2 条边. 由于带权有向图 G 中的每条边由 2 个顶点确定，而每个顶点则对应了构图中的 1 条实际合并边，因此，实际上算法每次取出的是构图中的 3 条合并边，而它们之间的位置关系对取点位置有不同限制.

现对取点时存在的几种限制关系进行讨论. 如图 7 所示，假设布线整体方向为从左向右，显然 3 条合并边之间存在相互平行、前 2 条边平行而当前边与后一条边垂直、前 2 条边垂直而当前边与后一条边平行、当前边与前后 2 条边都垂直 4 种情况. 图 7(a) 展示的是 3 条合并边相互平行的情况，算法需要在边序列中向后寻找一条与它们相互垂直的合并边. 设 $curredge_{\text{start}}.x$ 为当前合并边起点的横坐标，$curredge_{\text{end}}.x$ 为当前合并边终点的横坐标，$crossedge.x$ 为位于序列之后，且与当前合并边相互垂直的第 1 条合并边的横坐标，min定义为取括号内两者的最小值. 则取点范围为

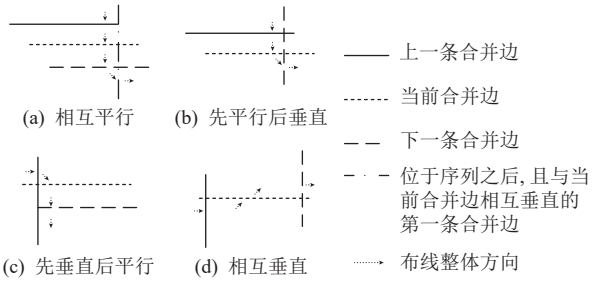


Fig. 7 Schematic diagram of four limiting relationships of acquisition points

图 7 取点的 4 种限制关系示意图

$$P_{\text{range}}=[curredge_{\text{start}}.x,\min(curredge_{\text{end}}.x,crossedge.x)]. \tag{7}$$

图 7(b)展示的是前 2 条边平行且当前边与后一条边垂直的情况. 设 $nextedge.x$ 为下一条合并边的横坐标, 在这种情况下, 此时的取点范围为

$$P_{\text{range}}=[curredge_{\text{start}}.x,\min(curredge_{\text{end}}.x,nextedge.x)]. \tag{8}$$

图 7(c)展示的是前 2 条边垂直而当前边与后一条边平行的情况. 设 $preedge.x$ 为上一条合并边的横坐标, max定义为取括号内两者的最大值, 在这种情况下, 此时的取点范围为.

$$P_{\text{range}}=[\max(preedge.x,curredge_{\text{start}}.x),curredge_{\text{end}}.x]. \tag{9}$$

图 7(d)展示的是当前边与前后 2 条边都垂直的情况. 在这种情况下, 此时的取点范围为

$$P_{\text{range}}=[\max(preedge.x,curredge_{\text{start}}.x),\ \min(curredge_{\text{end}}.x,nextedge.x)]. \tag{10}$$

以布线整体方向从左向右为例, 若上一个取点在当前合并边上的投影点落在所允许的取点范围之内, 那么优先将其作为选点. 如图 8 所示, 根据这一原则, 在合并边 HE_2 上, 算法选择点 v_1 的投影点 v_2 作为选点. 如果此时的取点范围包括了边界, 则考虑在边界的另一端进行取点. 若以上条件均不满足, 则算法需要在相对于走线穿过的方向及合并边上所允许的取点范围之内尽量靠右的位置进行取点, 间隔一个线宽与线间距之和的距离, 以最大限度地减少布线空间的浪费. 实际判断取点位置是否为布线方向右侧时, 算法首先将布线方向分为东、南、西、北、东南、西南、东北、西北 8 个方向, 以图 9 为例, 当前布线穿过方向显然为东北, 此时以斜率为 1 且朝向 x, y 轴正半轴的方向向量 ***dir*** 表示. 然后, 在合并边 AB 上取中点 M, 然后分别取中点 M 到 2 端点 A, B 的方向向量, 分别记做 ***MA*** 以及 ***MB***. 现在分别计算叉积

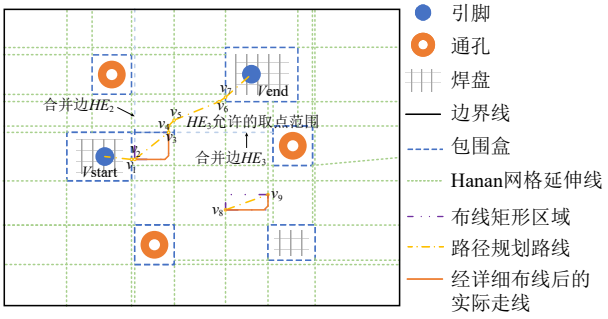


Fig. 8 Schematic diagram of path planning and detailed routing

图 8 路径规划以及详细布线示意图

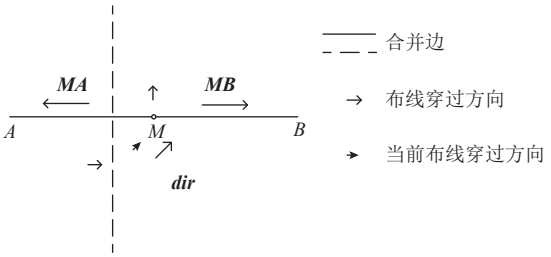


Fig. 9 Schematic diagram of vector and relative position judgment

图 9 向量与相对位置判断示意图

dot$_1$ 以及 ***dot***$_2$:

$$\boldsymbol{dot}_1=\boldsymbol{dir}\times\boldsymbol{MA}, \tag{11}$$

$$\boldsymbol{dot}_2=\boldsymbol{dir}\times\boldsymbol{MB}. \tag{12}$$

接着判断叉积 ***dot***$_1$, ***dot***$_2$ 的正负, 若叉积为正, 则向量之间的夹角为顺时针, 若叉积为负, 则为逆时针. 算法规定与 ***dir*** 之间夹角呈顺时针的向量所指向的方向为布线方向的右侧. 显然图 9 中 ***MB*** 方向为布线方向的右侧.

以图 8 为例, 根据这一原则, 在合并边 HE_3 上, 算法将取点范围内尽量靠近右侧的点 v_3 作为选点. 考虑到同一条合并边可能会被多条走线穿过, 所以算法在每次取点时都必须与上一条穿过该合并边的走线的取点保持一个线宽与线间距之和的距离. 重复上述过程, 算法便能够得到所有线网的路径规划序列, 这些序列以点集的形式予以表示.

3.6 详细布线

路径规划阶段得到了所有线网完整的路径规划序列, 但是这样的路径规划的直接连线结果违反了 DRC. 以图 8 为例, 此时路径规划后的走线角度并非 135°. 因此, 算法需要在此基础上进行详细布线, 以得到最终满足 DRC 的布线结果.

算法每次所要处理的是路径规划序列中的一条线段, 显然这一条线段是由 2 个点确定的. 算法将需要详细布线的一条线段的起点定义为入点, 而相应的终点则定义为出点.

为了缩减问题规模, 算法将详细布线所面临的场景分为了 2 种最为基本的情形: 2 条合并边之间相互平行以及 2 条合并边之间相互垂直, 不妨设此时的路径规划线段走向为自左向右. 对于 2 条合并边之间相互平行的情形又可细分为 2 种情况: 一是入点与出点的纵坐标相同(如图 8 中的点 v_1, v_2). 此时不做任何调整, 直接进入下一条线段的详细布线环节. 二是入点与出点的纵坐标不同(如图 8 中的点 v_8, v_9). 对于这类情况, 算法首先利用入点与出点的坐标信

息构造布线矩形区域，该矩形区域需要恰好包含这2个点. 由于先前构造带权有向图时对连边之间存在障碍物的情形进行了限制，因此，该矩形区域内不可能存在任何障碍物. 然后，算法在矩形区域内进行详细布线. 详细布线时遵循面积最小原则，即出点顺时针沿着矩形边界到达入点所走过的路径，以及经由详细布线后所得到的线段，它们之间所围成的多边形面积必须尽可能小. 显然，算法需要让转折的长度尽可能小，在考虑到DRC中的最小线长限制后，让转折的长度取最小线宽即可.

对于2条合并边之间相互垂直的情形，不失一般性的，设此时的路径规划线段走向为自左向上（如图8中的点v_2, v_3）. 对于这类情况，处理方法与2条合并边之间相互平行且入点与出点纵坐标不同的情况类似.

布线完成后检查起点线段与上一条线段的终点线段之间是否垂直，若是，则在起点线段开始处增加一转折，转折的长度同样取最小线宽即可.

对于其他走向的路径规划线段，算法在参照上述处理方法的同时进行相应的坐标变换即可. 在对所有路径规划线段应用详细布线算法后，便能够得到完整的符合DRC的详细布线结果.

4 实验结果及分析

算法使用C++作为编程语言进行实现. 所有实验都在配备了Intel® Xeon® Platinum 8260 CPU @ 2.40 GHz和128 GB内存的单台服务器上进行. 表1展示了来自于工业界的基准用例. 其中PCB5在PCB4的基础上将线网划分为2个群组. PCB9仅用于测试分层效果.

群组布线的布线质量可以通过采用一些评价指标进行评价. 群组布线的布线代价的计算方法由式(13)给出：

$$C_{\text{total}} = \alpha C_{WL} + \beta C_S + \gamma C_C. \tag{13}$$

总体布线代价C_{total}由3个部分加权求和得到，在不考虑DRC的情况下，C_{total}的值越小则说明群组布线的布线效果越好. α, β, γ表示权重，均可取1，所有群组的线长相似性代价C_{WL}、线段数代价C_S、紧凑性代价C_C的计算式的计算方法分别由式(14)~(16)所示.

$$C_{WL} = \frac{1}{n}\sum_{i=1}^{n}\frac{\max(WL_W^i)}{\min(WL_W^i)}, \tag{14}$$

$$C_S = \frac{1}{n}\sum_{i=1}^{n}\frac{\max(N_S^i)}{\min(N_S^i)}, \tag{15}$$

$$C_C = \frac{1}{n}\sum_{i=1}^{n}\frac{\min(W_S^i)}{\hat{W}_S}, \tag{16}$$

其中C_{WL}由每个群组i的单独相似性代价累加得到，n则代表了当前布线输入的群组数. $\max(WL_W^i)$表示群组i中线长最长的线网的线长，$\min(WL_W^i)$表示群组i中线长最短的线网的线长. C_{WL}的值越小则所有群组线网的整体布线线长越接近. $\max(N_S^i)$表示群组i中所包含的线段的数量最多的线网的段数，$\min(N_S^i)$表示群组i中的段数下界，取群组i中所有线网段数最小值即可. C_S的值越小则布线的拓扑相似性越高. $\min(W_S^i)$表示群组i中线段S所占据的宽度的最小值，计算W_S时需要分别计算群组i中最外层的2个线网与线段S的最小距离，两者中的较小者即为$\min(W_S^i)$. $\hat{W}_S$则取线网的线宽与线间距之和. C_C的值越小则布线的汇聚程度越高，而更高的汇聚程度能够有效地节约布线空间，从而尽可能减少给后续布线带来的困难.

将本文算法与2个最先进的布线器FreeRouting以及Cadance Allegro进行比较.2个布线器都使用了默认设置，并在实验中应用了相同的设计输入以及DRC. 而本文算法采用的是默认生成引导线的模式. 本文选择了组件之间相互远离的基准用例，以便清楚地展示布线结果. 通过比较3款布线器的布通率（正确地完成布线的线网占总线网数的比率[31]）、运行时间和违反设计规则（design rule violation, DRV）的次数总结了实验结果. 如表2所示，本文算法在布通率、运行时间和DRV次数方面优于FreeRouting和Cadance Allegro. 值得注意的是，本文算法以及Cadance Allegro可以完成总线或群组布线，而FreeRouting只能完成单线布线. 此外，Cadance Allegro以及FreeRouting无法在实际布线前进行相应的路径规划，这往往会导致一系列拥塞问题，继而在布线过程中需要频繁地进行拆线重布，这造成了大量的时间浪费. 更糟糕的

Table 1 Statistics for Benchmark Printed Circuit Board Design

表1 基准印刷电路板设计统计

设计	引脚数	焊盘数	过孔数	未布线线网数	总线群组组数
PCB1	10	10	0	5	1
PCB2	10	10	5	5	1
PCB3	16	32	22	8	1
PCB4	20	40	31	10	1
PCB5	20	40	31	10	2
PCB6	30	163	12	15	5
PCB7	84	172	86	42	3
PCB8	140	286	173	140	4
PCB9	6	12	12	3	1

Table 2 Comparison of Routing Results of Each Algorithm

表 2 各算法的布线结果比较

设计	布通率/%			运行时间/s			DRV 次数		
	FreeRouting	Cadance Allegro	本文算法	FreeRouting	Cadance Allegro	本文算法	FreeRouting	Cadance Allegro	本文算法
PCB1	100	100	100	0.03	0.008	0.001	0	0	0
PCB2	100	100	100	0.11	0.025	0.003	0	0	0
PCB3	100	100	100	0.7	0.19	0.023	1	0	0
PCB4	100	100	100	1.5	0.35	0.041	3	0	0
PCB5	100	100	100	1.5	0.37	0.048	3	0	0
PCB6	93.3	100	100	8.2	2.4	0.21	7	0	0
PCB7	57.1	97.6	100	94.7	13.2	2.1	15	2	0
PCB8	-	64.8	100	-	109.7	16.4	-	106	0

注: "-"表示设计无法在软件上运行, 或者运行时间超过 1 h.

是, FreeRouting 在一些布线发生拥塞的情况下会尝试设置过孔, 从而进行跨层布线, 这种做法将会使 PCB 板层数增加, 从而导致制造成本与制造难度大大增加.

表 3 通过使用式(14)~(16)对 3 款布线器的布线性能指标进行比较. 其中式(13)中 α, β, γ 的取值均为 1, 它们分别代表了群组布线同一群组中所有线网的布线线长尽量相近、所有线网的布线尽量汇聚以及所有线网的布线拓扑尽量相似的需求. 表 3 中 C_{total} 表示总体布线代价, C_{WL}, C_S, C_C 和 C_{total} 这些指标的值越小则布线效果越好. 如表 3 所示, 在线长相似性代价方面, 由于 FreeRouting 是以单线布线的布线结果进行对比的, 不具备聚集布线的能力, 因此通常来说它的线长相似性最好, 而 Cadance Allegro 在自动布线阶段采取的是聚集布线策略, 这种策略在进行切角处理后往往会占据大量的布线空间, 且在面对复杂例子时难以达到 100% 的布通率. 本文算法在某些例子上没有达到最佳的线长相似性, 但也在可接受的范围内. 在线段数代价方面本文算法较优, 这也意味着本文算法具有较好的布线拓扑相似性. 在紧凑性代价方面, 本文算法以及 Cadance Allegro 在面对有障碍物布线时都能保持最优, 这也就意味着相邻线网之间都能保持最小线间距, 而不具备聚集布线能力的 FreeRouting 在该指标上与本文算法以及 Cadance Allegro 有着较大差距. 本文算法在总体布线代价方面优于 FreeRouting 和 Cadance Allegro, 这就意味着, 总体而言, 本文算法对于群组布线问题有着更佳的效果.

图 10、图 11 显示了使用本文算法在 PCB4 上的布线结果, 能够清晰地展示本文算法的优势. 本文算法将 PCB 板上的过孔、焊盘等视作障碍物. 图 10 中, 本文算法将式(3)中的权重 $weight(e)$ 设置为 0, 从而屏蔽了引导线的功能, 此时群组走线退化为单线布线. 图 11 中, 本文算法将式(3)中的权重 $weight(e)$ 设置为 100, 从而强调了引导线的引导效果. 此时算法

Table 3 Comparison of Group Routing Performance of Each Algorithm

表 3 各方法的群组布线性能比较

设计	FreeRouting				Cadance Allegro				本文算法			
	C_{WL}	C_S	C_C	C_{total}	C_{WL}	C_S	C_C	C_{total}	C_{WL}	C_S	C_C	C_{total}
PCB1	1.04	1.67	4.04	6.75	1.01	1.00	3.50	5.51	1.16	1.67	1.00	3.83
PCB2	1.06	1.67	4.21	6.94	1.12	2.00	1.00	4.12	1.16	1.50	1.00	3.66
PCB3	1.07	2.00	4.36	7.43	1.17	1.50	1.00	3.67	1.29	1.32	1.00	3.61
PCB4	1.05	2.33	4.56	7.94	1.21	1.86	1.00	4.07	1.39	1.35	1.00	3.74
PCB5	1.05	2.33	4.56	7.94	1.14	2.00	1.00	4.14	1.17	1.22	1.00	3.39
PCB6	1.34	2.75	3.98	8.07	1.40	2.50	1.00	4.90	1.37	2.40	1.00	4.77
PCB7	1.20	2.13	4.29	7.62	1.20	1.82	1.00	4.02	1.18	1.60	1.00	3.78
PCB8	-	-	-	-	1.35	1.90	1.00	4.25	1.37	1.72	1.00	4.09

注: "-"表示设计无法在软件上运行, 或者运行时间超过 1 h.

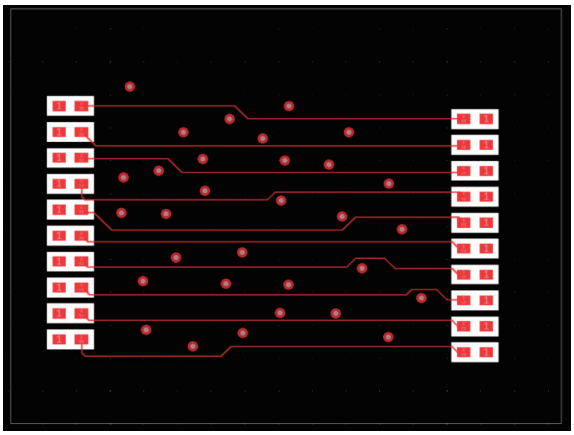


Fig. 10 Routing results of PCB4 when *weight*(*e*) is 0

图 10 权重 *weight*(*e*) 为 0 时的 PCB4 布线结果

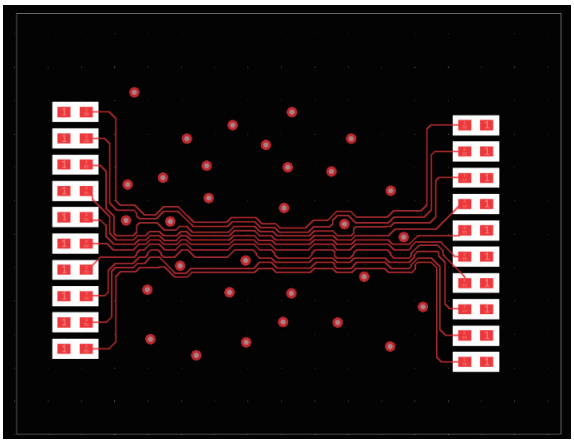


Fig. 11 Routing results of PCB4when *weight*(*e*) is 100

图 11 权重 *weight*(*e*) 为 100 时的 PCB4 布线结果

在左侧从上到下第 5 个焊盘与第 6 个焊盘中间出发，到右侧从上到下第 5 个焊盘与第 6 个焊盘中间中止，自动生成了 1 条引导线，并在参数设置时强调了引导线的引导作用，这将对这 10 根线网的路径规划产生汇聚作用，从而满足群组布线的要求. 图 12 则显示了本文算法在复杂例子上的布线结果. 图 13 显示的

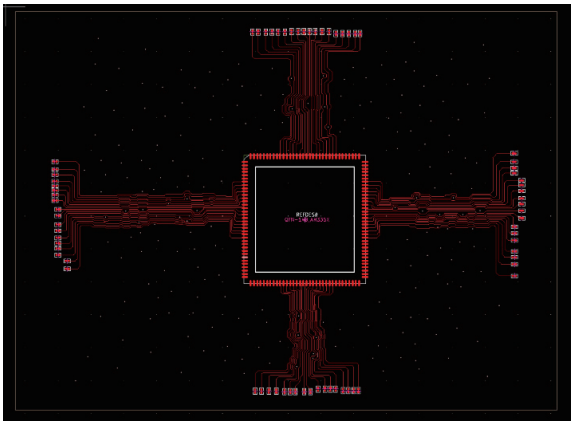


Fig. 12 Routing results of PCB8

图 12 PCB8 布线结果

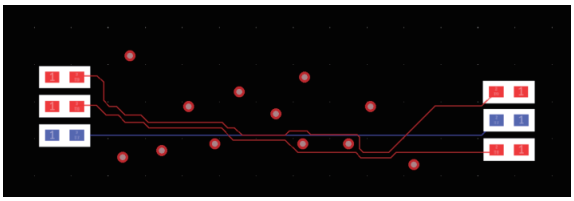


Fig. 13 Routing results of PCB9 after pre-layering when *weight*(*e*) is 100

图 13 经预先分层后权重 *weight*(*e*) 为 100 时 PCB9 的布线结果

是预先分配各引脚的层后，实施群组布线的布线效果，其中左边第 1, 2 行矩形焊盘（红色）与第 3 行矩形焊盘（蓝色）分别位于不同层，走线未发生交叉. 实验结果表明，本文算法对于 PCB 设计的群组布线任务是适用且高效的.

5 总 结

本文提出了一种用于 PCB 设计的群组布线算法，该算法考虑了 PCB 设计中的 DRC.算法利用构造带权有向图的方式实现了群组布线问题的数学建模，在有效表示电路板上的布线资源的同时避免了传统网格方法易受网格精度以及网格数量限制的缺陷. 此外，本文算法采用具有多线避让功能的启发式搜索算法，且支持人工指定引导线以引导布线方向，在有效降低时间开销的同时保证了布线质量. 最后，提出了一种简洁有效且与本文提出的路径规划算法相适配的群组布线方案. 与 FreeRouting 和 Cadance Allegro 相比，实验结果显示了本文算法的优越性.

作者贡献声明：邓新国负责方案设计和论文的撰写以及修改；张鑫泓负责完成实验、数据分析和论文的撰写以及修改；陈家瑞负责方案设计并修改论文；刘清海提出方案的概念和证明思路；陈传东提出指导意见并修改论文.

参 考 文 献

[1] Hsu J, Hung G, Tseng J, et al. High-speed data center PCB design challenges and findings by Intel® automatic in-board characterization[C/OL]//Proc of the 17th Int Microsystems, Packaging, Assembly and Circuits Technology Conf. Piscataway, NJ: IEEE, 2022[2024-04-08]. https://ieeexplore.ieee.org/document/9966724

[2] Tan Yan. Algorithmic studies on PCB routing[D]. Urbana-Champaign: University of Illinois at Urbana-Champaign, 2010

[3] Lin S T, Wang H H, Kuo C Y, et al. A complete PCB routing

methodology with concurrent hierarchical routing[C]//Proc of the 58th ACM/IEEE Design Automation Conf. Piscataway, NJ: IEEE, 2021: 1141−1146

[4] Lin T C, Merrill D, Wu Y Y, et al. A unified printed circuit board routing algorithm with complicated constraints and differential pairs[C]//Proc of the 26th Asia and South Pacific Design Automation Conf. New York: ACM, 2021: 170−175

[5] Deng Xinguo, Ye Sijin, Chen Jiarui, et al. Ordered escape routing combining improved A* algorithm with rip-up and reroute[J]. Journal of Electronics & Information Technology, 2021, 43(6): 1609−1616 (in Chinese)
(邓新国, 叶似锦, 陈家瑞, 等. 结合改进 A*算法与拆线重布的有序逃逸布线[J]. 电子与信息学报, 2021, 43(6): 1609−1616)

[6] Save Y D, Rakhi R, Shambhulingayya N D, et al. Oscad: An open source EDA tool for circuit design, simulation, analysis and PCB design[C]//Proc of the 20th Int Conf on Electronics, Circuits, and Systems. Piscataway, NJ: IEEE, 2013: 851−854

[7] Goyal P, Agrawal H, Paradiso J A, et al. BoardLab: PCB as an interface to EDA software[C]//Proc of the 26th Annual ACM Symp on the Djunct Publication of the User Interface Software and Technology. New York: ACM, 2013: 19−20

[8] Wane S, Patton R, Gross N. Unification of instrumentation and EDA tooling platforms for enabling smart chip-package-PCB-probe arrays co-design solutions using advanced RFIC technologies[C/OL]//Proc of the 2018 IEEE Conf on Antenna Measurements & Applications (CAMA). Piscataway, NJ: IEEE, 2018[2024-04-08]. https://ieeexplore.ieee.org/document/8530652

[9] Sherwani N A. Algorithms for VLSI Physical Design Automation[M]. 3rd ed. Berlin: Springer, 2012

[10] Fang S C, Chang K E, Feng W S, et al. Constrained via minimization with practical considerations for multi-layer VLSI/PCB routing problems[C]//Proc of the 28th ACM/IEEE Design Automation Conf. Piscataway, NJ: IEEE, 1991: 60−65

[11] Zhang Cong, Jin Huilin, Chen Jienan, et al. A hierarchy MCTS algorithm for the automated PCB routing[C]//Proc of the 16th Int Conf on Control & Automation. Piscataway, NJ: IEEE, 2020: 1366−1371

[12] Kahng A B, Lienig J. Software Engineering: A Practitioner's Approach[M]. Translated by Yu Yongbin, Feng Ce, Xu Ning, et al. 2nd ed. Beijing: China Machine Press, 2024 (in Chinese)
(Kahng A B, Lienig J. 超大规模集成电路物理设计: 从图分割到时序收敛[M]. 于永斌, 冯策, 徐宁, 等, 译. 2 版. 北京: 机械工业出版社, 2024)

[13] Kong Hui, Ma Qiang, Yan Tan, et al. An optimal algorithm for finding disjoint rectangles and its application to PCB routing[C]//Proc of the 47th Design Automation Conf. Piscataway, NJ: IEEE, 2010: 212−217

[14] Zhang Jixin, Xu Ning, Liu Mingyu. FanoutNet: A neuralized PCB fanout automation method using deep reinforcement learning[C]//Proc of the 37th AAAI Conf on Artificial Intelligence. Palo Alto, CA: AAAI, 2023: 8554−8561

[15] He Youbiao, Li Hebi, Jin Tian, et al. Circuit routing using Monte Carlo tree search and deep reinforcement learning[C/OL]//Proc of the 2022 Int Symp on VLSI Design, Automation and Test. Piscataway, NJ: IEEE, 2022[2024-04-08]. https://ieeexplore.ieee.org/document/9768074

[16] Liu Guoqing, Qian Yuhua, Zhang Yayu, et al. Best action identification algorithm in Monte Carlo tree search based on relative entropy confidence interval with given budget[J]. Journal of Computer Research and Development, 2023, 60(8): 1780−1794 (in Chinese)
(刘郭庆, 钱宇华, 张亚宇, 等. 给定预算下基于相对熵置信区间的蒙特卡洛树搜索最优动作识别算法[J]. 计算机研究与发展, 2023, 60(8): 1780−1794)

[17] Goh Y, Jung D, Hwang G, et al. Consumer electronics product manufacturing time reduction and optimization using AI-based PCB and VLSI circuit designing[J]. IEEE Transactions on Consumer Electronics, 2023, 69(3): 240−249

[18] Hung S C, Chen Hao, Sun Fankeng, et al. A DAG-based algorithm for obstacle-aware topology-matching on-track bus routing[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 40(3): 533−546

[19] Chen Jingsong, Kuang Jian, Zhao Guowei, et al. PROS 2.0: A plug-in for routability optimization and routed wirelength estimation using deep learning[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2022, 42(1): 164−177

[20] Zhang H T, Fujita M, Cheng C K, et al. SAT-based on-track bus routing[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 40(4): 735−747

[21] Chen Jingsong, Liu Jinwei, Chen Gengjie, et al. MARCH: Maze routing under a concurrent and hierarchical scheme for buses[C/OL]//Proc of the 56th Annual Design Automation Conf 2019. Piscataway, NJ: IEEE, 2019[2024-04-08]. https://ieeexplore.ieee.org/document/8807035

[22] Kim D, Do S, Lee S Y, et al. Compact topology-aware bus routing for design regularity[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2019, 39(8): 1744−1749

[23] Cheng Yihao, Yu Taochun, Fang Shaoyun. Obstacle-avoiding length-matching bus routing considering nonuniform track resources[J]. IEEE Transactions on Very Large Scale Integration Systems, 2020, 28(8): 1881−1892

[24] Yan Jintai, Chen Zhiwei. Obstacle-aware length-matching bus routing[C]//Proc of the 2011 Int Symp on Physical Design. New York: ACM, 2011: 61−68

[25] Yan Jintai. Single-layer obstacle-aware multiple-bus routing considering simultaneous escape length[J]. IEEE Transactions on Components, Packaging and Manufacturing Technology, 2022, 12(6): 902−915

[26] Kong Hui, Yan Tan, Wong M D F. Automatic bus planner for dense PCBs[C]//Proc of the 46th ACM/IEEE Design Automation Conf. Piscataway, NJ: IEEE, 2009: 326−331

[27] Wu Peici, Ma Qiang, Wong M D F. An ILP-based automatic bus planner for dense PCBs[C]//Proc of the 18th Asia and South Pacific Design Automation Conf. Piscataway, NJ: IEEE, 2013: 181−186

[28] Tang Hao, Liu Genggeng, Chen Xiaohua, et al. A Survey on Steiner tree construction and global routing for VLSI design[J]. IEEE Access, 2020, 8: 68593-68622

[29] Zachariasen M. A catalog of Hanan grid problems[J]. Networks, 2001, 38(2): 76−83

[30] Zhao Qi, Zhao Aqun. A multi-path routing algorithm base on A* algorithm[J]. Journal of Electronics & Information Technology, 2013, 35(4): 952−957 (in Chinese)
(赵奇, 赵阿群. 一种基于 A*算法的多径寻由算法[J]. 电子与信息学报, 2013, 35(4): 952−957)

[31] Ma Boyu, Xu Ning, Wu Haoying, et al. Routing Algorithm for flexible printed circuit channel area[J]. Journal of Computer-Aided Design & Computer Graphics, 2022, 34(8): 1179−1185 (in Chinese)
(马博宇, 徐宁, 吴皓莹, 等. 面向柔性电路通道区的自动布线算法[J]. 计算机辅助设计与图形学学报, 2022, 34(8): 1179−1185)



Deng Xinguo, born in 1975. PhD, associate professor. His main research interest includes electronic design automation.
邓新国, 1975 年生. 博士, 副教授. 主要研究方向为电子设计自动化.



Zhang Xinhong, born in 1998. Master. His main research interest includes electronic design automation.
张鑫泓, 1998 年生. 硕士. 主要研究方向为电子设计自动化.



Chen Jiarui, born in 1981. PhD, associate professor. His main research interest includes electronic design automation.
陈家瑞, 1981 年生. 博士, 副教授. 主要研究方向为电子设计自动化.



Liu Qinghai, born in 1982. PhD, professor. Member of CCF. His main research interests include graph theory and its application, combinatorial optimization, and electronic design automation.
刘清海, 1982 年生. 博士, 教授. CCF 会员, 主要研究方向为图论及应用、组合优化、电子设计自动化.



Chen Chuandong, born in 1982. PhD, associate professor. His main research interest includes electronic design automation.
陈传东, 1982 年生. 博士, 副教授. 主要研究方向为电子设计自动化.