结合改进A*算法与拆线重布的有序逃逸布线

邓新国[①] 叶似锦[①] 陈家瑞*[①] 陈传东[②]

[①](福州大学数学与计算机科学学院 福州 350108)

[②](福州大学物理与信息工程学院 福州 350108)

摘 要：逃逸布线是印刷电路板设计的一个重要组成部分。针对并行逃逸布线的方法用于较大规模电路板布线时速度慢且结果不够好的问题，该文提出一种结合改进A*算法与拆线重布的有序逃逸布线方法。首先，通过代价预估函数确定引脚的布线顺序，使用改进A*算法初始化有序逃逸布线。接着，优化同长度布线路径，调整拥挤区域布线路径。最后，使用A*算法和广度优先搜索进行拆线重布。实验结果表明，该方法对给出的所有测试用例都实现了100%的逃逸，得到有序逃逸路径的可行解非常接近最优解，CPU时间比布尔可满足性问题(SAT)算法与最小费用多商品流(MMCF)算法平均减少分别约为95.6%, 97.8%，总体线长也接近最优。提出的方法能够明显减少寻找可行解的时间，提高布线质量。

关键词：A*算法；拆线重布；有序逃逸布线；最短路径

中图分类号：TN43; TP302.1 **文献标识码**：A **文章编号**：1009-5896(2021)06-1609-08

DOI: 10.11999/JEIT201033

Ordered Escape Routing Combining Improved A* Algorithm with Rip-up and Reroute

DENG Xinguo[①] YE Sijin[①] CHEN Jiarui[①] CHEN Chuandong[②]

[①](*College of Mathematics and Computer Science*, *Fuzhou University*, *Fuzhou* 350108, *China*)

[②](*College of Physics and Information Engineering*, *Fuzhou University*, *Fuzhou* 350108, *China*)

Abstract: Escape routing is an important part of the integrated circuit physical design. In order to solve the problem of slow parallel escape routing with unsatisfactory outcomes, an algorithm of ordered escape routing, combining the improved A* algorithm with the rip-up and reroute method is proposed. Firstly, the routing sequence of pins is determined by the cost estimation function and the improved A* algorithm is used to initialize the ordered escape routing. Secondly, the routing paths of the same length are optimized and the routing paths of the crowded areas are adjusted. Finally, A* algorithm and breadth-first search are employed to rip-up and reroute. The experimental results shows that this method achieved 100% escape routing for all given test cases, and that the feasible solution of the ordered escape paths is, to a great extent, close to the optimal solution. Compared to the Boolean Satisfiability Problem (SAT) algorithm and MMCF algorithm, this algorithm reduces CPU time by 95.6% and 97.8%, respectively, and makes the overall line length shorter. It is evident that the proposed method reduces the time required to find the feasible solution and optimize wire routing.

Key words: A* algorithm; Rip-up and reroute; Ordered escape routing; Shortest path

收稿日期：2020-12-09；改回日期：2021-04-16；网络出版：2021-04-27

*通信作者： 陈家瑞 chenjiarui@fzu.edu.cn

基金项目：国家自然科学基金 (61977017)，国家科技部重点研发计划课题(2018YFB2202704)，中国福建光电信息科学与技术创新实验室(闽都创新实验室)基金(2021ZR142)

Foundation Items: The National Natural Science Foundation of China (61977017), The Key Research and Development Project of the Ministry of Science and Technology (2018YFB2202704), The Fujian Science & Technology Innovation Laboratory for Optoelectronic Information of China (Mindu Innovation Laboratory) (2021ZR142)

1 引言

有序逃逸布线问题，作为印刷电路板(Printed Circuit Board, PCB)设计中的关键问题。目前，电路板的布线在很大程度上还是由专业人员手工完成。随着电路板器件集成度不断提高，器件引脚数量大规模增加，现有自动逃逸布线算法已经无法为专业人员提供有效的帮助。因此，如何能够快速得到一个可行的布线方案已成为急需解决的问题。

逃逸布线是将引脚网格内的某些引脚连接到网格的边界上，最主要的考虑因素是可布线性，在可布线性得到满足的情况下优化线长，逃逸布线问题主要分为无序逃逸布线和有序逃逸布线两类[1,2]。目前，对于无序逃逸布线的研究已经取得了很好的成果，由于无序逃逸布线没有终点顺序要求，所以对于该问题的研究[3–5]都使用网络流方法进行求解并可以在短时间内得到最优解，且文献[6,7]研究并解决多层逃逸问题。

而有序逃逸布线由于逃逸目标线路顺序的限制，使用网络流方法无法解决，现有的研究提出的方法又可以分为并行布线方法和串行布线方法[8]。并行布线就是一次性对所有引脚进行线路规划，文献[9,10]用整数线性规划(Integer Linear Programming, ILP)解决了一个相关但不同的逃逸问题，该问题中引脚具有固定的逃逸终点，文献[11]使用分组等策略改进ILP，降低约束项数量。Tomioka等人[12]只采用无绕行单调模式，这意味着即使存在解决方案，也不一定能保证找到解决方案。尽管他们使用的方法在图有环时考虑非单调布线，但是在资源冲突需要绕行时，非单调布线就会失败。文献[13,14]将有序的逃逸布线转换为一个布尔可满足性问题(Boolean Satisfiability Problem, SAT)，并通过SAT求解器解决。该方法无需对路线始末进行任何假设，即可准确完成布线。但是，该方法很难解决逃逸布线容量不为1的情况。在最新的研究中，Jiao等人[15]使用最小费用多商品流(Min-cost Multi-Commodity Flow, MMCF)方法来解决这个问题，主要是通过使用虚拟节点，构建具有始末节点且符合求解约束的图，进而使用MMCF求解器来求解，在大规模引脚阵列中构建图，虚拟点的数量过多会使得运算量大幅度增加。

串行布线则是按照每条线的顺序依次进行布线，每条线布线完成后更新占用的资源，下一条线在空余资源上进行规划。串行布线对布线的先后顺序有很强的依赖性，因为先行的线占用了资源后，之后的引脚可能无法逃逸，因此，使用拆线重布[16]的方法来对已有的线进行调整。

并行布线的方法随着电路板引脚数量变多，时间复杂度会迅速增加，耗费的CPU时间会成倍增长，甚至会无法求解。因此，当遇到的引脚阵列规模较大时，并行布线的方法会允许程序寻找可行解而不是最优解。

在逃逸布线中兼顾可布线性与线长，提出了一种3阶段串行布线的逃逸布线方法。主要步骤为：首先构造预估函数估算引脚逃逸所需的资源,确定可行的布线顺序，以此构造初始解，接着对拥挤区域与同长度布线路径进行优化，最后使用拆线重布的方法对无法逃逸的引脚进行重新布线。

2 问题描述

逃逸布线是将引脚阵列内的某些引脚连接到组件边界上。

目前市面上器件的封装形式主要分为交错引脚阵列(Staggered Pin Arrays, SPA)[17]和网格引脚阵列(Grid Pin Array, GPA)，本文主要研究GPA下的逃逸布线，如图1。定义一个多行多列的规则排布引脚阵列，假设在每4个引脚中间有一个中心节点，使得需要逃逸的引脚通过中心节点连接到边界上。

算法输入为给定引脚阵列构建的有向图$G(V,E)$，其中$V=\{v_0,v_1,v_2,\cdots,v_m\}$是引脚与中心点标号集合，$E$是边集合，$S,T$分别表示起点集合(引脚集合)和终点集合，$x_{uw}=1$表示由节点$u$前往节点$w$的有向边被选中，$x_{uw}=0$则表示没被选中，

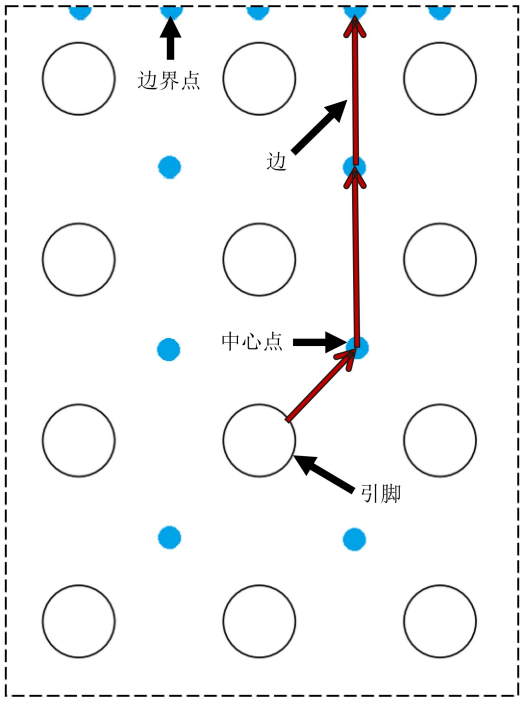

图 1 GPA单边逃逸布线示意图

c_{uw}表示由节点u前往节点w的有向边的代价，将问题描述为一个整数线性规划问题，即

$$\begin{aligned}
\min\quad & \sum_{(u,w)\in E} c_{uw}x_{uw}\\
\text{s.t.}\quad & \sum_{w\in V} x_{uw} \le 1, u\in V\\
& \sum_{w\in V} x_{uw} - \sum_{w\in V} x_{wu} = 0,\ u\notin S\cup T\\
& \sum_{w\in V} x_{uw} - \sum_{w\in V} x_{wu} = 1,\ u\in S\\
& \sum_{w\in V} x_{wu} - \sum_{w\in V} x_{uw} = 1,\ u\in T\\
& x_{uw}\in(0,1),(u,w)\in E
\end{aligned}\tag{1}$$

其中，第1个约束确保每个节点只会被一条引脚布线选中，避免冲突。第2个约束确保每个除了源和目标节点外的中间节点入度与出度是平衡的，即构造的线路是连续的。其余约束确保每条线由逃逸引脚出发，到达边界终点，最终目标是最小化总体线长。每条边的基础代价都为1。

有序逃逸布线问题的输入与逃逸布线相同，但要给出逃逸引脚的顺序。本文中出线的顺序都以顺时针排列，如图2。

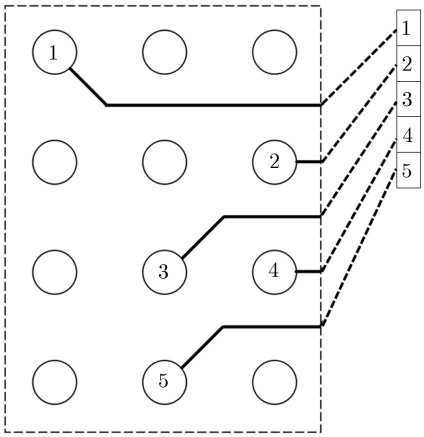

图 2 单边有序逃逸布线示意图

当引脚间的布线容量增加时，问题规模会成倍地增加，从而导致算法复杂的大幅提高。但是在实际情况下，引脚间的布线容量不一定为1，算法亦考虑了逃逸布线问题中可变容量的情况，不同容量的引脚阵列如图3所示。

3　有序逃逸布线

图4为算法流程图，算法框架分为构建逃逸布线初始解、对拥挤区域的部分线路进行优化调整以及拆线重布共 3 个部分。

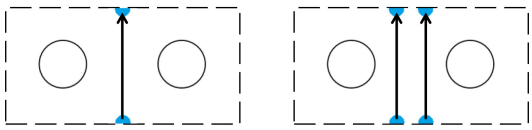

图 3 布线容量可变示意图

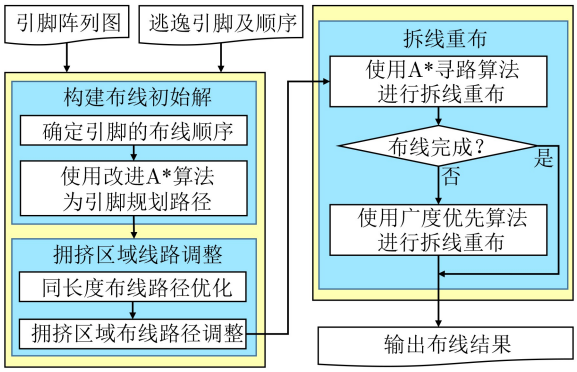

图 4 算法流程图

3.1　构建布线初始解

在串行布线中，布线的顺序对于最终解的质量会产生重大影响，在构建初始解时，采用构造预估函数对每一个引脚在当前可用资源下逃逸可能耗费的资源进行计算，根据计算结果确定引脚的逃逸顺序，能够使初始解的结果更优。设边界点集合为K，边界点标记值为k，初始值为–1，当前要逃逸的引脚编号为n，对应目标边界点编号为m，则$k_m = n$，计算引脚与边界点的曼哈顿距离作为该引脚的逃逸预估代价，引脚代价预估函数定义同式(2)。

$$\text{cost}_n = \min(|x_n - x_m| + |y_n - y_m|), m\in M \tag{2}$$

其中，M为可用边界点集合，任取$m\in M$，可使以下布尔表达式成立

$$\prod_{i=1}^{|K|}(k_i \le k_{i+1} \vee k_{i+1} < 0) = 1 \tag{3}$$

按照预估函数计算出各个引脚逃逸所需的预估代价后，选择所需代价最小的引脚优先逃逸，因为引脚逃逸所需资源越少，则代表引脚需要在之后的引脚先布线后再进行绕路逃逸的可能性越低。

获得需要逃逸的引脚编号与边界点编号后，需要选用一种最短路径的寻路算法，阅读多路径[18]文献及多次实验对比后，最终选用A*算法来规划逃逸路径，总体代价函数如式(4)所示

$$F(i) = G(i) + H(i),\quad H(i) \le H*(i) \tag{4}$$

其中，$F(i)$是当前点i的综合代价，$G(i)$是从起点沿着当前生成的路径移动到当前点的真实移动代价，$H(i)$是A*算法的启发式函数，计算从当前点移动到终点的估算成本，$H*(i)$是从当前点移动到终点

的真实成本。本算法选用当前点与终点的曼哈顿距离作为估算成本$H(i)$，由于$H(i)=H*(i)$，因此A*算法能够在最短时间取得最优解。A*算法主要通过维护OPEN表和CLOSE表来规划路线。其中，OPEN表用于存放已经生成，且已用启发式函数做过估算，但尚未产生后继节点的那些节点，也称未考查节点；CLOSE表则用于存放已经生成，且已经考查过的节点。

A*算法步骤如下：

输入：图$\boldsymbol{G}$，起始点s，终点t。

输出：路径点集合$\boldsymbol{P}$。

步骤 1　读入图$\boldsymbol{G}$，起始点s，终点t。

步骤 2　初始化OPEN=$\{s\}$，CLOSE=$\{\}$。

步骤 3　如果OPEN=$\{\}$，规划路径失败。

步骤 4　从OPEN表中取出$\boldsymbol{F}$值最小的节点n，n放入CLOSE表中。

步骤 5　如果n=t，规划路径成功，返回路径。

步骤 6　计算所有n的后继节点$\boldsymbol{F}$值。

步骤 7　将不在OPEN表中的点加入OPEN表，否则比较F值，如果当前计算的F值更小，则更新该节点的前驱节点为当前节点。

步骤 8　返回步骤3。

为了进一步提高算法的寻路效率，使用改进的A*算法[19]，将代价函数修改为

$$F(i)=(1-w)\times G(i)+w\times H(i) \tag{5}$$

其中，w定义为

$$w=\frac{G(i)}{G(i)+H(i)} \tag{6}$$

当搜索区域靠近边界时，权值w会减小以弱化线长的影响，能够使算法规划路径速度更快，为w设置上下界$w\in(0.6,0.9)$确保算法可接纳性不受影响。此外，还使用小根堆来使A*算法最坏时间复杂度由$O(n^2)$降为$O(n\log_2 n)$。

为证明所采用的构造预估函数逃逸所需耗费的资源来引导初始解布线顺序是有效的，使用另外两种方法来代替预估函数计算的布线顺序，选取给出的测试用例中引脚阵列规模较大的4个例子来对比3种方法构建的初始解的总线长与布通率。对比结果如表1所示，其中，第1种是依照逃逸引脚的顺序进行布线，第2种是距离可用边界近的引脚优先逃逸，第3种则是使用预估函数进行成本估计后确定逃逸顺序。

表1、表2和表3中总线长以单元网格为基准。从结果中可以看出，第3种方法在大规模阵列的布通率上有较大提高，在小规模阵列上也可以避免部分引脚进行不必要的绕行。

3.2 拥挤区域线路调整

调整阶段主要是对拥挤区域与可同长度布线的路径进行优化，策略是在不减少当前已经布好的线的情况下，对现有线的位置进行调整，释放因布线顺序所产生的无意义的资源占用。

3.2.1 同长度布线路径优化

同长度布线路径优化是指引脚选择另一条相同程度的路线可以防止去其他引脚绕路前往终点，从

表 1　3种顺序选择方法结果对比

测试用例	方法1		方法2		方法3	
	总线长	布通率(%)	总线长	布通率(%)	总线长	布通率(%)
Case5	52	92.0	52	92.0	50	92.0
Case6	46	86.6	60	93.3	57	93.3
Case7	255	80.0	259	82.2	276	88.8
Case8	419	91.4	429	92.8	471	97.1
平均	–	87.5	–	90.1	–	92.8

表 2　3种算法布线结果对比

测试用例	#Row	#Col	#Pin	#Side	#Cap	文献[13]SAT-based		文献[15]MMCF-based		本文方法	
						总线长	用时(s)	总线长	用时(s)	总线长	用时(s)
Case1	8	6	6	1	1	21	0.07	21	0.05	21	0.002
Case2	8	8	8	2	1	24	0.07	24	0.09	24	0.001
Case3	8	8	10	2	1	39	0.26	39	0.15	39	0.003
Case4	8	8	10	3	1	36	0.66	36	1.34	36	0.003
Case5	10	6	25	3	1	*67	*0.14	*67	*0.58	67	0.027
Case6	10	10	15	4	1	62	1.57	62	3.36	62	0.011
Case7	20	15	45	2	1	326	2.02	338	10.24	326	0.251
Case8	30	30	70	4	1	–	–	*530	*30.15	489	0.032
Case9	8	10	31	2	2	–	–	162	6.12	153	0.085
Case10	30	60	130	3	2	–	–	–	–	2210	4.311

表 3　3个阶段布线结果对比

测试用例	阶段1			阶段2			阶段3		
	布通率(%)	总线长	用时(s)	布通率(%)	总线长	用时(s)	布通率(%)	总线长	用时(s)
Case5	92.00	50	0.006	92.00	50	0.002	100	67	0.019
Case6	93.33	57	0.002	93.33	56	0.001	100	62	0.008
Case7	88.88	276	0.010	93.33	288	0.005	100	326	0.236
Case8	97.14	471	0.012	97.14	467	0.005	100	489	0.015
Case9	74.19	82	0.021	74.19	82	0.002	100	153	0.085
Case10	95.38	2048	0.192	95.38	2044	0.082	100	2210	4.037

而避免资源的浪费。如图5所示，调整引脚2的布线方式，能够使得引脚1在布线时使用更少的资源。

图5为右侧单边逃逸，实线为已布好的线，虚线表示该引脚调整后的走线。

3.2.2　拥挤区域布线路径调整

拥挤区域是指在一定区域内线的排布使得某些引脚无法找到可行的边界点作为自己的终点。如果边界点集中有点i使式(7)成立

$$k_{i+1}-k_i>1, k_i>=0 \tag{7}$$

则代表该区域对于引脚k_i+1是拥挤的，如图6所示。算法将对部分引脚进行重新布线，在不减少成功逃逸的引脚数量的前提下，为目标引脚预留出可行的目标终点。

本阶段算法步骤如下：

输入：已布线图$\boldsymbol{G}$，未逃逸引脚集合$\boldsymbol{R}$。

输出：布线路径集合$\boldsymbol{P}$。

步骤 1　读入布线图$\boldsymbol{G}$与引脚$\boldsymbol{R}$。

步骤 2　从引脚集合$\boldsymbol{R}$中选取引脚$\boldsymbol{r}$。

步骤 3　使用式(7)寻找引脚$\boldsymbol{r}$的拥挤区域。

步骤 4　检索拥挤区域两侧的空余边界点。

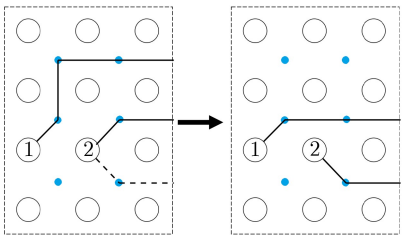

图 5 同长度优化调整示意图

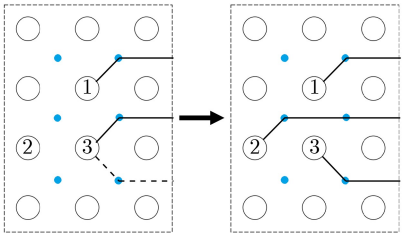

图 6 造成阻挡的拥挤区域布线调整示意图

步骤 5　将已布好的线向两侧移动。

步骤 6　对引脚$\boldsymbol{r}$进行重新布线。

步骤 7　返回步骤2，直至集合为空。

3.3　拆线重布

如果经过第2阶段优化后依然存在无法逃逸的引脚，则代表仅靠对现有线路的简单调整无法给予该引脚足够的资源，因此，将对图中线路进行拆线重布来寻找所有引脚的逃逸方法。

3.3.1　基于A*算法的拆线重布

使用前两个阶段确定的边界终点，A*算法能够快速规划路径。

代价函数对于拆线重布阶段至关重要，通过修改边的代价将指引程序在不同的方向检索可行路径。若一条边同时被多条线路选中，则该边为拥挤边，当线路需要通过拥挤边时，增加拥挤边的代价会鼓励程序考虑从其他位置布线。

因此，本阶段边的代价由代价函数确定

$$c_n^i=(b_n+h_n^i)\times p_n \tag{8}$$

其中，c_n^i表示边n在被第i次重复选中后的代价，b_n表示边n的基础代价。h_n^i是历史代价，用来表示边的拥挤程度，定义为

$$h_n^{i+1}=\begin{cases}h_n^i+b_n, & \text{边}n\text{为拥挤状态}\\ h_n^i, & \text{其他}\end{cases} \tag{9}$$

p为人为设定的惩罚系数，目的是防止边的代价的增长速度过快，定义为

$$p=\frac{i}{i+1} \tag{10}$$

如果某条边的代价达到$c_{\max}$，则代表布通率无法达到100%，程序将最后一次对未逃逸引脚队列中的引脚进行布线后进入下一个步骤，$c_{\max}$设为定值100。

对拆线重布效果有主要影响的另一个因素是布线顺序，对每个引脚前一次布线的真实线长升序排列来确定布线顺序可以有效地减少重布线的次数，并且提高布线的质量。

3.3.2 基于广度优先算法的拆线重布

由于预先确定的始末节点可能会导致算法无法找到合适的逃逸路径，因此，在不预设边界终点的情况下，使用基于广度优先算法进行第二轮拆线重布，当算法检索到边界节点时，使用式(3)判断该边界节点是否符合要求。同时，代价函数重新定义如式(11)所示：

$$c_n^i = b_n + h_n^i \times p_n \tag{11}$$

其中，p定义为常数1.1，广度优先搜索会消耗大量的时间，因此通过放大历史代价来加快算法规划路径的速度，但是过多地增加边的代价，会导致算法规划的路径绕行过多，为了平衡布线速度与总线长两个方面，该常数设置为1.1时能在短时间内得到较优的布线结果。需要说明的是，如果经过上一个步骤后全部引脚都能够成功逃逸，则不进行这个步骤。

4 实验结果及分析

由于知识产权限制，本文无法得到已发表论文中实现方法的源代码。因此，本文对互联网中开源的论文方法复现代码进行修改调试，重新实现的程序可能由于计算机硬件以及其他各方面的差异，导致实验结果与已发表的论文中结果有所不同。但在相同的测试案例中，我们复现的代码的实验结果与已有论文中的结果相差较小，因此，将复现代码的实验结果作为对比的实验结果。

由于没有行业基准的有序逃逸布线测试用例，文献[13–15]构造了一些可布线的测试用例，并使用了一些文献[15]中的测试用例用作对比，这些案例的引脚阵列规模由小到大，包含了单侧、两侧、三侧以及全方向的逃逸用例，同时也构建了引脚间容量为2的测试用例来测试算法性能。实验结果如表2所示，其中Row和Col代表引脚的行数和列数，Pin表示需要逃逸的引脚数量，Cap表示引脚间的布线容量，Side则表示该引脚阵列有几个方向可用于逃逸，其中带*号的表示该结果为文献[13–15]中的实验结果。图7则给出了Case8的布线结果图。

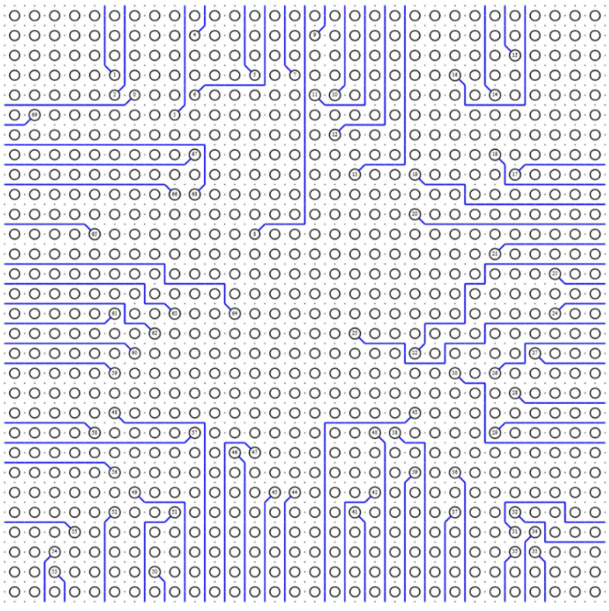

图 7 Case8最终布线结果图

实验使用个人电脑CPU为Intel Core i5-6500，主频为3.20 GHz。从表2可以看出，随着引脚阵列规模的不断增大，无论是使用SAT求解器，或是使用ILP求解器，都会因为约束式的大量增加，导致求解速度不断下降，甚至是无法求解；而我们的算法在数据规模变大时，求解时间仍处在可接受的范围内。

算法的CPU时间相较于SAT算法结果平均提升约95.6%，相较于MMCF算法结果平均提升约97.8%。在大规模引脚阵列案例Case8中，与MMCF方法相比，总体线长减少约7.7%，布线时间减少约99.8%。在容量为2的测试用例中，SAT方法由于本身的限制，无法解决此类问题，而使用MMCF方法在规模较大的例子中会因为构建的图规模过大而需要消耗大量的CPU时间。而本文算法在可变容量的情况下，依旧可以在短时间内给出可行解，证明了算法的可行性。

为进一步分析算法效果，收集对比了部分测试用例经过每一个阶段后的布线率、总线长以及每一个阶段的用时情况，结果如表3所示。可以看出，经过第1阶段的布线，剩余无法逃逸的引脚数量比例已经很低。在经过第2阶段优化调整后，总线长都有一定程度的减少，可以看出拥挤资源得到释放。在第3阶段结束后，所有的引脚都能够成功逃逸，证明了该方法每个阶段的有效性。从表中可以看出串行布线最耗费时间的阶段是第3阶段的拆线重布，并且从Case8与Case10 (图8)的布线结果可以看出，拆线重布的次数过多会导致边的代价过高使得算法无法找到最短路径，因此，算法对于引脚的位置与顺序较为敏感。

在通常情况下，专业设计的电路板器件，在于可布线性方面都进行过精心调整，器件内部要逃逸的引脚不会出现大量无序、混乱的情况。综上所述，算法无论是在学术方面，还是在工业设计中，都具有一定的优势。

5 结束语

针对现有的逃逸布线算法大都采用整数线性规划等并行布线方法，在大规模引脚阵列中由于约束式过多导致运行速度慢，并且求得的解效果不佳甚

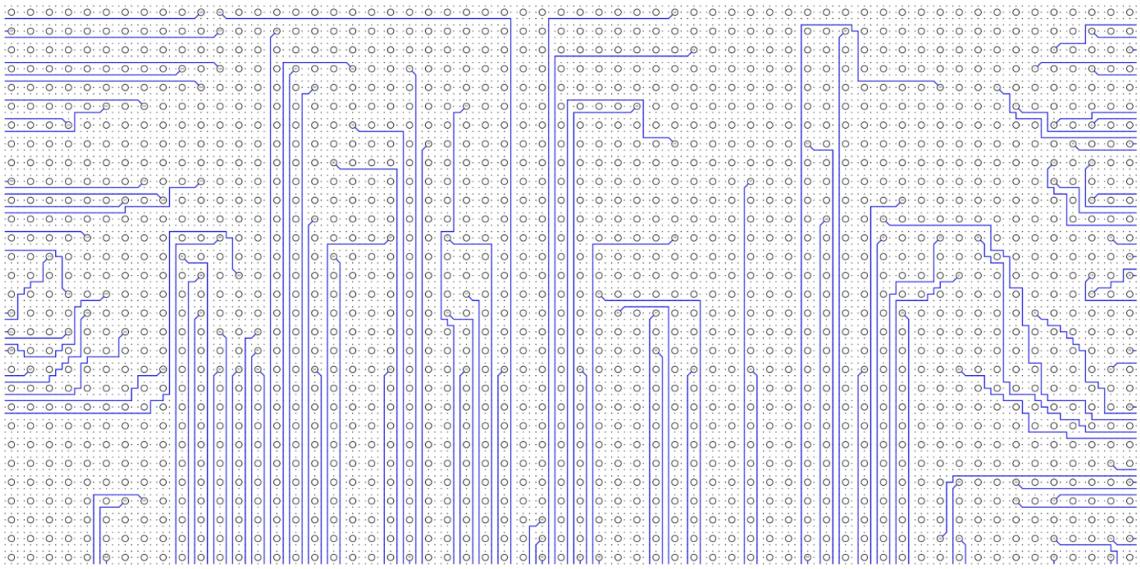

图 8 Case10最终布线结果图

至无法求解的情况。本文提出一种基于A*算法的串行有序逃逸布线算法。用预估函数判断目标终点，用改进的A*算法快速构建初始解，用拆线重布对所有引脚进行布线。多个测试用例结果表明，总体质量较高的逃逸布线能够在较短时间内完成，在规模最大的测试用例Case8中，效果提升最为明显。本文提出的方法能够有效提升布线速度与质量。

随着VLSI技术的发展，电路板器件集成度不断提高，逃逸布线规模也不断增大，求解时间急剧增加。因此，探索并改进串行布线算法，使算法能够更好地求解更大规模的逃逸布线问题是我们今后研究的一个重点。此外，逃逸布线中的差分对以及分层问题的求解也是亟待解决的重点问题，设计一个合理的算法来综合解决PCB自动布线方面的问题将是我们未来的研究方向。

参 考 文 献

[1] 徐宁, 洪先龙. 超大规模集成电路物理设计理论与算法[M]. 北京: 清华大学出版社, 2009: 147–172.
XU Ning and HONG Xianlong. Very Large Scale Integration Physical Design Theory and Method[M]. Beijing: Tsinghua University Press, 2009: 147–172.

[2] YAN Tan and WONG M D F. Recent research development in PCB layout[C]. 2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD'10), San Jose, USA, 2010: 398–403. doi: 10.1109/ICCAD.2010.5654190.

[3] WANG Kan, WANG Huaxi, and DONG Sheqin. Escape routing of mixed-pattern signals based on staggered-pin-array PCBs[C]. 2013 ACM International Symposium on Physical Design, Stateline, United States, 2013: 93–100. doi: 10.1145/2451916.2451941.

[4] FANG Jiawei, LIN I J, CHANG Y W, *et al*. A network-flow-based RDL routing algorithmz for flip-chip design[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2007, 26(8): 1417–1429. doi: 10.1109/TCAD.2007.891364.

[5] MA Qiang, YOUNG E F Y, and WONG M D F. An optimal algorithm for layer assignment of bus escape routing on PCBs[C]. The 48th Design Automation Conference, San Diego, United States, 2011: 176–181. doi: 10.1145/2024724.2024764.

[6] CHO B G, KAM D G, and KOO H I. Mixed-signal escape routing algorithm for multilayer PCBs[J]. *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 2019, 9(8): 1576–1586. doi: 10.1109/TCPMT.2019.2900463.

[7] YAN Jintai. Direction-constrained rectangle escape routing[J]. *ACM Transactions on Design Automation of Electronic Systems*, 2018, 23(3): 34. doi: 10.1145/3178047.

[8] ALPERT C J, MEHJTA D P, and SAPATNEKAR S S. Handbook of Algorithms for Physical Design Automation[M]. Boca Raton: CRC Press, 2008: 469–475.

[9] TOMIOKA Y and TAKAHASHI A. Routing of monotonic parallel and orthogonal netlists for single-layer ball grid array packages[J]. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 2006, E89-A(12): 3551–3559. doi: 10.1093/ietfec/e89-a.12.3551.

[10] FANG Jiawei, HSU C H, and CHANG Yaowen. An integer-linear-programming-based routing algorithm for flip-chip designs[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2009, 28(1): 98–110. doi: 10.1109/TCAD.2008.2009151.

[11] LIAO Zhaopo and DONG Sheqin. A constraint-driven

compact model with partition strategy for ordered escape routing[C]. 2020 on Great Lakes Symposium on VLSI, Virtual Event, New York, USA, 2020: 393–398. doi: 10.1145/3386263.3406942.

[12] TOMIOKA Y and TAKAHASHI A. Monotonic parallel and orthogonal routing for single-layer ball grid array packages[C]. Proceedings of 2006 Asia and South Pacific Conference on Design Automation, Yokohama, Japan, 2006: 6–12. doi: 10.1109/ASPDAC.2006.1594758.

[13] LUO Lijuan and WONG M D F. Ordered escape routing based on Boolean satisfiability[C]. 2008 Asia and South Pacific Design Automation Conference, Seoul, Korea (South), 2008: 244–249. doi: 10.1109/ASPDAC.2008.4483950.

[14] LUO Lijuan and WONG M D F. On using SAT to ordered escape problems[C]. 2009 Asia and South Pacific Design Automation Conference, Yokohama, Japan, 2009: 594–599. doi: 10.1109/ASPDAC.2009.4796545.

[15] JIAO Fengxian and DONG Sheqin. Ordered escape routing with consideration of differential pair and blockage[J]. *ACM Transactions on Design Automation of Electronic Systems*, 2018, 23(4): 46. doi: 10.1145/3185783.

[16] 朱自然, 陈建利, 朱文兴. 基于多阶段拆线重布的总体布线算法[J]. 计算机辅助设计与图形学学报, 2016, 28(11): 2000–2008. doi: 10.3969/j.issn.1003-9775.2016.11.023.

ZHU Ziran, CHEN Jianli, and ZHU Wenxing. A global routing algorithm based on multistage rip-up and reroute[J]. *Journal of Computer-Aided Design & Computer Graphics*, 2016, 28(11): 2000–2008. doi: 10.3969/j.issn.1003-9775.2016.11.023.

[17] SHI Rui and CHENQ C K. Efficient escape routing for hexagonal array of high density I/Os[C]. The 43rd Design Automation Conference, San Francisco, USA, 2006: 1003–1008. doi: 10.1109/DAC.2006.229424.

[18] 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. doi: 10.3724/SP.J.1146.2012.00983.

ZHAO Qi and ZHAO Aqun. A multi-path routing algorithm base on A* algorithm[J]. *Journal of Electronics & Information Technology*, 2013, 35(4): 952–957. doi: 10.3724/SP.J.1146.2012.00983.

[19] 高庆吉, 于咏生, 胡丹丹. 基于改进A*算法的可行性路径搜索及优化[J]. 中国民航学院学报, 2005, 23(4): 42–45. doi: 10.3969/j.issn.1001-5590.2005.04.010.

GAO Qingji, YU Yongsheng, and HU Dandan. Advanced A* algorithm for path-finding and optimization[J]. *Journal of Civil Aviation University of China*, 2005, 23(4): 42–45. doi: 10.3969/j.issn.1001-5590.2005.04.010.

邓新国：男，1975年生，博士，副教授，硕士生导师，研究方向为VLSI电子设计自动化.

叶似锦：男，1997年生，硕士生，研究方向为VLSI布线算法设计.

陈家瑞：男，1981年生，博士，讲师，硕士生导师，研究方向为VLSI电子设计自动化.

陈传东：男，1984年生，博士，讲师，硕士生导师，研究方向为VLSI电子设计自动化.

责任编辑：马秀强