2021 年 9 月　　运 筹 学 学 报　　第 25 卷 第 3 期
Sep., 2021　　Operations Research Transactions　　Vol.25 No.3

超大规模集成电路布局的优化模型与算法*

黄志鹏[1]　李兴权[2]　朱文兴[1,2,†]

摘要　布局确定集成电路单元在芯片中的具体位置, 在单元互不重叠的基础上优化一些性能指标。该问题是 NP困难的组合优化问题, 是超大规模集成电路物理设计的核心问题之一, 对集成电路的性能指标, 如线网可布通性、时延、功耗、电路可靠性等有重大影响。在现代的集成电路设计中, 布局问题通常包含数百万个集成电路单元, 以及大小相异的异质性模块, 和各种复杂的布局约束。目前的超大规模集成电路布局算法通常分解为总体布局、布局合法化和详细布局三个步骤。根据近年来集成电路布局算法的研究进展, 综述并分析集成电路的总体布局、布局合法化和详细布局的相关优化模型和算法, 并展望进一步的研究方向。

关键词　超大规模集成电路, 总体布局, 合法化, 优化算法

中图分类号　TP302.1, O221

2010 数学分类号　68W35, 90C90

Optimization models and algorithms for placement of very large scale integrated circuits*

HUANG Zhipeng[1]　LI Xingquan[2]　ZHU Wenxing[1,2,†]

Abstract　Placement is one of the critical stages in the physical design of very large scale integrated circuits (VLSI), which has significant impact on the performance of integrated circuits, such as routability, delay, power consumption, circuit reliability, etc. Placement determines the specific positions of cells of a chip, by optimizing some performance metrics under the constraint of cells non-overlapping, which is an NP-hard combinatorial optimization problem. Modern placement algorithms need to deal with large-scale designs with millions of cells, heterogeneous modules with different sizes, and various complex placement constraints. Modern placement algorithms usually consist of three steps: global placement, legalization, and detailed placement. Based on the recent research progress of VLSI placement algorithms, this paper surveys related optimization models and algorithms for VLSI global placement, legalization and detailed placement, and discusses possible research directions.

Keywords　VLSI, global placement, legalization, optimization algorithm

Chinese Library Classification　TP302.1, O221

2010 Mathematics Subject Classification　68W35, 90C90

收稿日期: 2021-03-15

* 基金项目: 国家自然科学基金 (Nos. 61672005, 61907024)

1. 福州大学离散数学与理论计算机科学研究中心, 福建福州 350116; Center for Discrete Mathematics and Theoretical Computer Science, Fuzhou University, Fuzhou 350116, Fujian, China

2. 鹏城实验室, 广东深圳 518055; Peng Cheng Laboratory, Shenzhen 518055, Guangdong, China

† 通信作者　E-mail: wxzhu@fzu.edu.cn

近年来, 随着集成电路制造工艺的迅速发展, 集成电路单元的几何尺寸不断缩小, 集成度不断提高, 加上存储空间的局限性和封装工艺的限制, 超大规模集成电路 (Very Large Scale Integration, VLSI) 设计的复杂性急剧增加。物理设计是 VLSI 设计的关键环节之一, 也是集成电路设计自动化 (EDA) 工具的核心。VLSI 物理设计主要分为以下几个阶段[1]: 电路划分、版图规划、布局、布线和封装. 布局是 VLSI 物理设计的核心之一, 是典型的超大规模 NP困难组合优化问题, 对集成电路的性能指标, 如时延、线网可布通性、功耗、电路可靠性等有重大影响[2]。布局是将集成电路单元放置在固定的布局区域内, 在单元之间相互不重叠的条件下优化一些度量指标 (如线长, 线网可布通性等)。即便布局问题有着数十年的研究历史, 最近的研究表明当前的布局工具仍然不能生成接近最优的布局结果[3], 因此如何设计高性能的布局工具仍然引起广泛关注。此外, 先进制程下的设计规则产生了许多新的布局约束, 同时为了得到更好的设计效果, 许多物理设计规则需要在布局阶段考虑, 例如线网的可布通性、栅栏区域约束、时序要求、模糊效应和近邻效应等。日益增长的设计复杂性对 VLSI 布局问题的模型及其算法设计提出了更高的要求, 因此, 设计出快速、有效地处理超大规模集成电路布局问题的模型及其算法, 仍然是当前及今后很长一段时间 VLSI 布局问题面临的挑战[3]。

目前, 求解 VLSI 布局问题的算法主要可分为三类[3, 4]: 基于模拟退火的算法、基于划分的算法和基于解析方法的算法。基于模拟退火的算法利用模拟退火机制改变单元在布局区域中的位置来优化布局。由于在较小的解空间中进行搜索, 通常在小型设计上可以获得良好的布局质量。但基于模拟退火的布局工具效率较低, 并且缺乏可扩展性, 不适用于大规模电路。基于模拟退火的布局工具见文献 [5-7] 等。基于划分的布局算法的主要思想是递归地划分电路和布局区域, 然后以自顶向下的方式将子电路分配到子区域中, 直到子电路足够小为止。该算法通常非常有效且易于扩展, 但子问题之间缺少用于交互的全局信息, 从而限制了解的质量。基于划分的布局工具见文献 [8-15] 等。

基于解析方法的布局算法将布局问题转化为由线长目标函数和单元不重叠约束组成的数学规划问题, 然后采用不同的光滑化技术对目标及约束进行光滑化处理, 并构造各种优化算法或启发式方法求解所得到的布局优化模型[4]。近期的文献和布局竞赛表明, 对于超大规模集成电路设计, 基于解析方法的布局算法可以得到更好的解质量。基于解析方法的布局工具见文献 [16-46] 等。尽管集成电路布局问题的研究取得了很大进展, 目前尚无关于哪种算法是最佳算法的共识。工具性能的比较仍然主要依靠经验[47], 并且经常受到代码实现的质量、并行处理和高性能库的使用以及报告方法等的影响。特别地, 学术类工具与商业工具相比在针对特定目标的优化上具有竞争优势, 但在更普遍存在的多目标优化方面效果不佳[48]。

由于布局问题通常包含数百万个单元, 且要求在可接受的计算时间内得到高质量的解, 现代的基于解析方法的布局算法通常包括三个步骤[4]: 总体布局, 布局合法化和详细布局, 如图 1 所示。总体布局在忽略部分布局约束 (例如, 单元重叠) 的情况下, 为每个模块和单元寻找关于某些目标函数的最佳位置; 合法化消除所有单元重叠, 同时尽可能地保留在总体布局阶段得到的 “良好” 的解; 详细布局通常以迭代的方式进一步改善合法的布局解。总体布局很大程度上影响了最终布局结果的质量, 因此在布局中被认为是最至关重要的部分。

本文首先简要介绍集成电路布局问题的两个基本要素: (1) 线长模型, (2) 重叠处理技术。其次, 我们着重综述线长模型、重叠减少技术以及与其他布局约束的集成, 和现有的

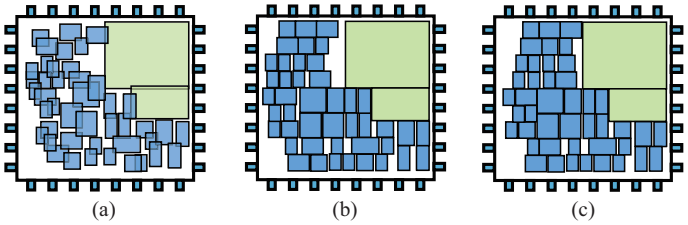

图 1 (a) 总体布局, (b) 布局合法化, (c) 详细布局

布局研究中常用到的优化模型和算法等。进一步, 我们总结常用的布局合法化和详细布局技术, 以将这些技术组合成一个完整的集成电路布局流程。最后, 我们对当前技术做出总结并指出一些未来的研究方向。

1 集成电路布局的优化模型

如前所述, 超大规模集成电路布局是一个具有多目标和多约束的优化问题, 直接求解该问题是非常困难的。最常见的方法是首先构造只优化线长的布局问题的算法, 在可接受的时间内计算出好的解, 并且算法具有良好的可扩展性, 使得其他目标和约束可以同时进行考虑或在下一步处理[3]。

在 VLSI 布局问题中, 将电路中的单元视为顶点, 将线网视为超边。给定一个具有 n 个单元和 m 条线网的电路, 可以将其转化为超图 $H(V,E)$, 其中 $V=\{v_1,\cdots,v_n\}$ 是所有顶点的集合, $E=\{e_1,\cdots,e_m\}$ 是所有超边的集合。单元 v_i 的中心点坐标用 (x_i,y_i) 表示, 单元的宽度、高度分别用 w_i 和 h_i 表示, A_i 是单元 v_i 的面积, $i=1,\cdots,n$。布局区域是矩形区域, 其左下角和右上角顶点分别为 $(0,0)$ 和 (W,H)。布局的基本目标是在满足所有单元互不重叠的条件下, 确定每个单元在布局区域内的位置, 使得某些指标达到最优。由于线长目标与芯片性能高度相关, 布局的目标中最常用的指标是总线长最小, 可以将其描述为如下的约束最小化问题:

$$\begin{aligned}&\min\ \ W(x,y)\\&\text{s.t.}\ \ \text{no}\ \ \text{cell}\ \ \text{overlap},\end{aligned}$$

其中 $W(x,y)$ 是线长函数, 通常采用所有线网的半周长线长 (half-perimeter wirelength, HPWL) 之和[2-4,49]:

$$W(x,y)=\sum_{e\in E}(\max_{v_i,v_j\in e}|x_i-x_j|+\max_{v_i,v_j\in e}|y_i-y_j|)。$$

为了描述单元之间的不重叠约束, 引入区间 $[L1,R1]$ 和 $[L2,R2]$ 来描述重叠长度[49]:

$$\Theta([L1,R1],[L2,R2])=[\min(R1,R2)-\max(L1,L2)]^+,$$

其中

$$[z]^+=\begin{cases}z, & \text{当 } z>0;\\ 0, & \text{当 } z\leqslant 0。\end{cases}$$

进一步, 在布局中单元 v_i 和 v_j 间的重叠面积可以用下式计算:

$$O_{ij}(x_i,y_i,x_j,y_j)=\Theta\Big(\Big[x_i-\frac{w_i}{2},x_i+\frac{w_i}{2}\Big],\Big[x_j-\frac{w_j}{2},x_j+\frac{w_j}{2}\Big]\Big)\\ \times\Theta\Big(\Big[y_i-\frac{h_i}{2},y_i+\frac{h_i}{2}\Big],\Big[y_j-\frac{h_j}{2},y_j+\frac{h_j}{2}\Big]\Big)。$$

因此, VLSI 布局问题的数学模型可描述为[34]:

$$\begin{aligned}\min\ & W(x,y)\\ \text{s.t.}\ & O_{ij}(x_i,y_i,x_j,y_j)=0,\ \ \forall v_i,v_j\in V,\ \ i\neq j,\\ & 0\leqslant x_i-\frac{w_i}{2},x_i+\frac{w_i}{2}\leqslant W,\ \ \forall v_i\in V,\\ & 0\leqslant y_i-\frac{h_i}{2},y_i+\frac{h_i}{2}\leqslant H,\ \ \forall v_i\in V。\end{aligned}\tag{1}$$

在上述问题中, 只优化线长趋向于将单元聚集在一起, 而消除重叠要求单元散开, 因此目标和约束是相互冲突的。此外, 在问题 (1) 中, 约束个数是 $O(n^2)$, 当 n 较大时, 约束个数太多; HPWL 函数是凸函数, 但不是处处可微的。由于超大规模集成电路布局问题中的单元个数通常有几十万甚至数百万, 所以很难直接求解问题 (1)。固然可以用组合优化的方法求解超大规模集成电路布局问题, 但是, 组合优化技术无法与其他优化目标很好地结合, 并且对该问题的可扩展性比连续优化方法差[3]。因此, 针对问题 (1) 我们先简要介绍一些常用的线长光滑化模型、重叠处理技术, 随后进一步讨论布局问题研究中所用的优化模型和算法。

1.1 线长模型

线长光滑化模型主要可分为二次线长模型和非二次线长模型, 下面我们介绍一些较为常见的线长光滑化模型。

1) 二次线长模型

集成电路的二次布局技术使用的是二次线长模型[49]。假设芯片中所有线网都只包含两个引脚, 则二次线长模型为

$$\hat{W}(x,y)=\sum_{e\in E}\frac{1}{2}\Big(\sum_{i,j\in e,i<j}w_{x,ij}(x_i-x_j)^2+\sum_{i,j\in e,i<j}w_{y,ij}(y_i-y_j)^2\Big),$$

其中 $w_{x,ij}(w_{y,ij})$ 为连接单元 i,j 的线网的权重 (没有线网连接则为 0)。假设单元 v_1 到 $v_{\bar{n}}$ 是可移动单元, $v_{\bar{n}+1}$ 到 v_n 是固定位置的单元, 则 $\boldsymbol{x}=(x_1,x_2,\cdots,x_{\bar{n}})^{\mathrm{T}}$, $\boldsymbol{y}=(y_1,y_2,\cdots,y_{\bar{n}})^{\mathrm{T}}$ 是可移动单元在 x 方向和 y 方向的坐标向量。进一步根据可移动单元的坐标以矩阵形式可将上述线长模型简洁地表示为

$$\hat{W}(x,y)=\frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}+d_x^{\mathrm{T}}\boldsymbol{x}+\frac{1}{2}\boldsymbol{y}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{y}+d_y^{\mathrm{T}}\boldsymbol{y}+C。$$

其中 C 是某常数项。

因此, 在不考虑重叠的情况下, 最小化线长的布局结果可以由求解下述线性代数方程组给出:

$$\boldsymbol{Q}\boldsymbol{x}+\boldsymbol{d}_{\boldsymbol{x}}=0,\ \ \boldsymbol{Q}\boldsymbol{y}+\boldsymbol{d}_{\boldsymbol{y}}=0。$$

由于二次线长模型只能处理包含两个引脚的线网, 所以包含多个引脚的线网通常用如图 2 所示的团模型或星模型[50] 转化为只包含两个引脚的线网。

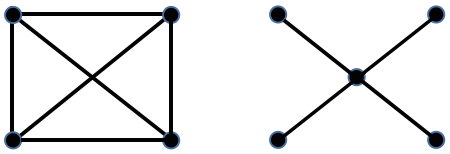

图 2 4 引脚线网的两种模型 (a) 团模型, (b) 星模型

团模型考虑了每条线网的所有可能的两引脚连接, 而星模型为每条线网引入了一个额外的星形引脚, 并将线网的每个引脚连接到星形引脚。由于 HPWL 仅考虑边界引脚之间的距离, 而团模型的内部连接会影响线长, 所以团模型和 HPWL 之间有较大的近似误差 (星模型类似)。文 [26] 在二次线长模型的基础上提出了一种删除所有内部二引脚连接的 Bound2Bound 线网模型, 可以更精确地近似 HPWL。令 p 表示线网中引脚的个数, Bound2Bound 线网模型在 x 方向上表示为:

$$W_{e,x}^{\mathrm{B2B}} = \frac{1}{2}\sum_{i,j\in e} w_{x,ij}(x_i - x_j)^2,$$

其中线网权重 $w_{x,ij}^{\mathrm{B2B}}$ 被定义为

$$w_{x,ij}^{\mathrm{B2B}} = \begin{cases} 0, & v_i, v_j \in inner\ pins, \\ \dfrac{2}{p-1}\dfrac{1}{|x_i - x_j|}, & \text{其他。} \end{cases}$$

y 方向上的表示类似。

2) 非二次线长模型

非二次布局技术使用比二次线长模型更为准确的非二次光滑化模型近似 HPWL, 例如 Log-Sum-Exp (LSE) 线长模型[20], Weighted Average (WA) 线长模型[51], L_p 范数线长模型[52] 等。对含有 $\hat{n}$ 个引脚的线网 e, LSE 线长模型[20] 通过如下函数近似 HPWL:

$$\hat{W}_e^{\mathrm{LSE}}(x,y) = \gamma\Big(\ln\sum_{i\in e}\exp\Big(\frac{x_i}{\gamma}\Big) + \ln\sum_{i\in e}\exp\Big(\frac{-x_i}{\gamma}\Big) + \ln\sum_{i\in e}\exp\Big(\frac{y_i}{\gamma}\Big) + \ln\sum_{i\in e}\exp\Big(\frac{-y_i}{\gamma}\Big)\Big),$$

其中 γ 是用来控制模型精度的光滑化参数。该线长函数的误差上界是 $\gamma\ln\hat{n}$。当 γ 趋于 0 时, LSE 线长趋近半周长线长。由于计算机的精度限制, 通常只能选一个适当小的 γ 以避免在算法实现过程中出现算术溢出。类似地, WA 线长模型[51] 是:

$$\begin{aligned}\hat{W}_e^{\mathrm{WA}}(x,y) = &\frac{\sum_{i\in e} x_i\exp(x_i/\gamma)}{\sum_{i\in e}\exp(x_i/\gamma)} - \frac{\sum_{i\in e} x_i\exp(-x_i/\gamma)}{\sum_{i\in e}\exp(-x_i/\gamma)} \\ &+ \frac{\sum_{i\in e} y_i\exp(y_i/\gamma)}{\sum_{i\in e}\exp(y_i/\gamma)} - \frac{\sum_{i\in e} y_i\exp(-y_i/\gamma)}{\sum_{i\in e}\exp(-y_i/\gamma)},\end{aligned}$$

其误差上界为 $\frac{\gamma\Delta x}{1+\exp(\Delta x/\hat{n})} + \frac{\gamma\Delta y}{1+\exp(\Delta y/\hat{n})}$。另一种较好的近似方法是 L_p 范数线长模型[52]:

$$\hat{W}_e^{L_p}(x,y) = \Big(\sum_{i\in e} x_i^p\Big)^{\frac{1}{p}} - \Big(\sum_{i\in e} x_i^{-p}\Big)^{-\frac{1}{p}} + \Big(\sum_{i\in e} y_i^p\Big)^{\frac{1}{p}} - \Big(\sum_{i\in e} y_i^{-p}\Big)^{-\frac{1}{p}}。$$

以上线长模型中, LSE 线长模型和 WA 线长模型得到了较多的应用。

1.2 重叠处理技术

单元重叠的处理是设计集成电路总体布局算法的关键, 目前的方法主要有基于划分的方法, 基于单元移动的方法, 基于最小费用流指派的方法, 基于扩散方程的方法, 以及基于频率控制和密度控制的方法等[4]。

1) 划分

划分将复杂的电路分解为较小的子电路, 并将这些子电路分配给适当的子区域。在随后的优化步骤中, 每个单元被限制在其分配的子区域内。在基于划分的布局方法中, 重叠的减少通常包括划分和优化两个阶段。在划分阶段, 有以下两种主流的方法: (1) 对于给定的初始布局, 根据单元的物理位置进行排序, 确定割线的位置, 并把单元划分到子区域; (2) 允许将一些单元分配给子区域来构建一个指派问题, 在满足容量约束的同时获得对子区域的单元分配[53]。细化阶段进一步应用启发式算法, 例如 FM 算法[54] 和基于窗口的重新划分[55, 56], 以进一步提高划分质量。

2) 单元移动

文 [50] 提出在保留布局区域中单元的相对顺序的情况下, 通过移动将单元进行扩散。首先将布局区域划分为大小相同的网格, 分别沿 x 和 y 方向移动单元。在沿 x 方向移动单元时, 先根据行中每个网格的当前利用率调整网格结构使得利用率分布较为均匀, 调整后的网格结构通常是不规则的。然后根据初始和调整后的网格边界将该行内的单元进行线性映射, 即根据网格边界的变化等比例地计算出该网格中单元的新的位置。在对每一行进行了单元移动后, 再沿 y 方向对每一列进行单元移动。

3) 最小费用流指派

文 [57] 提出使用最小费用流指派来扩散单元。首先根据给定的初始物理位置, 将附近的单元聚在一起, 保持每个簇的大小尽可能均匀。然后将布局区域划分为均匀的子区域, 并使用最小费用流算法将聚类的簇分配到相应的子区域中。在基于划分的重叠处理技术中, 由于单元大小不同, 允许将单元部分分配给子区域, 而在该方法中通过控制簇的大小来维持与子区域一一对应。

4) 扩散

在文 [58] 中, 减少单元重叠被建模为由密度梯度驱动的物理扩散过程。密度、时间和空间之间的关系表示为:

$$\frac{\partial d_{x,y}(t)}{\partial t} = \bar{D}\nabla^2_{x,y}(t), \tag{2}$$

其中 $d_{x,y}(t)$ 是点 (x,y) 处在时刻 t 的密度, $\bar{D}$ 是扩散率。边界条件为 $\nabla d_{\bar{x},\bar{y}}(t)=0$, 其中 $(\bar{x},\bar{y})$ 为边界上的点。对任意的位置 (x,y) 和时刻 t, 定义由密度和局部密度梯度确定的 2D 扩散速度场 $v_{x,y}=(v^H_{x,y}, v^V_{x,y})$ 如下:

$$v^H_{x,y} = -\frac{\partial d_{x,y}(t)}{\partial x}/d_{x,y}(t),\ v^V_{x,y} = -\frac{\partial d_{x,y}(t)}{\partial y}/d_{x,y}(t)。 \tag{3}$$

获得速度场后, 给定一个单元的初始位置, 可以通过对速度场进行积分来确定该单元在任意时刻的位置。

为了应用于布局, 文 [58] 利用前向时间中心空间 (FTCS)[59] 策略将公式 (2) 离散化, 其中 $t+1$ 时刻一个网格单元的密度由 t 时刻该单元的密度和其四个相邻网格单元的密度决定。速度场方程 (3) 也可以通过 FTCS 策略离散化, 进而可以递归地计算出单元在每个时刻的位置。

5) 频率控制

文 [60] 将布局区域划分为大小相同的网格并计算网格上的密度, 然后通过二维离散余弦变换 (DCT) 将空域上的密度矩阵 $D=\{d_{x,y}\}$ 转化到频域上。令 $F=\{f_{i,j}\}$ 表示频率分布矩阵:

$$f_{i,j}=\frac{2}{N}C(i)C(j)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\Big(d_{x,y}\cos\Big(\frac{(2x+1)i\pi}{2N}\Big)\cos\Big(\frac{(2y+1)j\pi}{2N}\Big)\Big),$$

其中 (x,y) 是空域中的坐标, (i,j) 是频域中的坐标。通过变换, 频率矩阵 F 代表密度矩阵的不同频率分布。为了使单元均匀分布, 密度分布应集中在具有更好均匀性的频域上, 因此, 扩散成本 DIST 被定义为:

$$\mathrm{DIST}=\sum_{i,j}(w_{i,j}f_{i,j}^2),$$

其中 $w_{i,j}$ 是在频域 (i,j) 处的分布权重, 这里 DIST 近似为单元位置的凸二次函数。

6) 密度控制

在基于解析方法的布局算法中, 均匀分布单元的一种最流行的做法是密度控制。将布局区域划分为均匀的不重叠的网格 bin, bin b 的密度函数表示为

$$D_b(x,y)=\sum_{v\in V}\Theta_x(b,v)\Theta_y(b,v), \tag{4}$$

其中 $\Theta_x(b,v)$ 是 bin b 与单元 v 在 x 方向重叠的长度, $\Theta_y(b,v)$ 类似。进一步地, 单元之间的无重叠约束松弛为如下的不等式约束:

$$D_b(x,y)\leqslant M_b, \text{对每一个 bin } b,$$

其中 $D_b(x,y)$ 是所有可移动单元与 bin b 重叠的总面积, M_b 是 bin b 可容纳的可移动单元的面积。由于 bin b 和单元 i 在 x 方向上的重叠函数 $\Theta_x(b,v)$ 是不平滑的, 所以很难直接对其进行优化, 通常采用不同的密度函数光滑化策略对其进行平滑, 主要有以下几种方法:

(1) 钟形函数光滑化

布局工具 APlace[19] 和 NTUplace3[27] 使用如下钟形势函数 p_x 对 $\Theta_x(b,v)$ 进行平滑:

$$p_x(b,v)=\begin{cases}1-ad_x^2, & \text{当 } 0\leqslant d_x<\dfrac{w_v}{2}+w_b;\\ \Big(d_x-\dfrac{w_v}{2}-2w_b\Big)^2, & \text{当 } \dfrac{w_v}{2}+w_b\leqslant d_x<\dfrac{w_v}{2}+2w_b;\\ 0, & \text{当 } \dfrac{w_v}{2}+2w_b\leqslant d_x,\end{cases}$$

其中

$$a=\frac{4}{(w_v+2w_b)(w_v+4w_b)},\quad b=\frac{2}{w_b(w_v+4w_b)},$$

d_x 是 x 方向上单元 v 到 bin b 的中心点距离, w_b 是 bin b 的宽度, y 方向上类似, 则非光滑的密度函数 $D_b(x,y)$ 可以用光滑化后的函数 $\hat{D}_b(x,y)$ 替代:

$$\hat{D}_b(x,y)=\sum_{v\in V}c_vp_x(b,v)p_y(b,v),$$

其中 c_v 是归一化因子使得单元的总势能等于其面积。

(2) 亥姆霍兹光滑化

布局工具 mPL6[21] 使用具有零导数边界的 Helmholtz 方程的解来近似密度函数

$$\Delta\hat{D}_b(x,y)-\varepsilon\hat{D}_b(x,y)=-D_b(x,y),$$

其中 $\varepsilon>0$ 是光滑参数, Δ 是拉普拉斯算子。

(3) 泊松光滑化

在布局工具 FDP[61]、Kraftwerk[26] 和 mFAR[18] 中, 将密度视为静电势, 通过泊松方程的解来光滑近似密度函数:

$$\Delta\hat{D}_b(x,y)=-D_b(x,y),$$

并通过几何多重网格求解器 DiMEPACK[62] 来求解泊松方程。

基于电场能的布局工具 ePlace[38] 在总体布局中将单元视为正电荷, 并将单元分布的密度约束视为总势能约束。基于给定的电荷密度分布和静电平衡, ePlace 使用泊松方程对电势和电场分布进行建模, 其中利用布局的特性设置了 Neumann 边界条件和相容性条件, 构造了如下的偏微分方程组:

$$\begin{cases}\nabla\cdot\nabla\psi(x,y)=-\rho(x,y),\\ \hat{n}\cdot\nabla\psi(x,y)=0,\ (x,y)\in\partial\mathbf{R},\\ \iint_{\mathbf{R}}\rho(x,y)\mathrm{d}x\mathrm{d}y=\iint_{\mathbf{R}}\psi(x,y)\mathrm{d}x\mathrm{d}y,\end{cases}$$

并用谱方法求解。ePlace 可以快速计算电势和电场分布, 从而在最小化线长的同时快速扩散布局单元。

为了避免数值求解泊松方程导致的数值误差, 文 [42] 精确构造了单元分布的密度函数, 得到了泊松方程的如下可以计算任意位置电势的解析解:

$$\psi(\hat{x},\hat{y})=\sum_{u=0}^{\infty}\sum_{p=0}^{\infty}a_{u,p}\cos\left(\frac{u\pi}{W}\hat{x}\right)\cos\left(\frac{p\pi}{H}\hat{y}\right),$$

其中

$$\begin{cases}a_{0,0}=0,\\ a_{\mu,0}=\dfrac{2W^2}{\mu^3\pi^3H}\displaystyle\sum_{i=1}^{n}h_i\left(\sin\frac{\mu\pi(x_i+\frac{w_i}{2})}{W}-\sin\frac{\mu\pi(x_i-\frac{w_i}{2})}{W}\right),\\ a_{0,\eta}=\dfrac{2H^2}{\eta^3\pi^3W}\displaystyle\sum_{i=1}^{n}w_i\left(\sin\frac{\eta\pi(y_i+\frac{h_i}{2})}{H}-\sin\frac{\eta\pi(y_i-\frac{h_i}{2})}{H}\right),\\ a_{\mu,\eta}=\dfrac{4W^2H^2}{(\mu^2H^2+\eta^2W^2)\mu\eta\pi^4}\displaystyle\sum_{i=1}^{n}\left(\sin\frac{\mu\pi(x_i+\frac{w_i}{2})}{W}-\sin\frac{\mu\pi(x_i-\frac{w_i}{2})}{W}\right)\\ \qquad\cdot\left(\sin\dfrac{\eta\pi(y_i+\frac{h_i}{2})}{H}-\sin\dfrac{\eta\pi(y_i-\frac{h_i}{2})}{H}\right)。\end{cases}$$

为了提升求解速度, 基于解析函数将布局区域划分为均匀的网格, 并使用快速傅立叶变换快速求近似解析解。所得到的泊松方程的解不仅在误差上小于原有的数值方法, 计算复杂度也达到了 $O(n\log n)$, 从而在效率和质量上取得了一个良好的权衡。此外, 文 [43] 进一步把泊松方程的解析解扩展到三维情况, 并应用于 2.5D 异质性 FPGA 芯片的布局。

2 集成电路总体布局

基于解析方法的集成电路布局算法通常包括三个步骤[4]: 总体布局, 布局合法化和详细布局。总体布局忽略了部分布局约束 (如单元重叠), 为每个单元寻找最佳摆放位置。合法化消除所有单元重叠, 同时尽可能地保留在总体布局阶段得到的 “良好” 的解。详细布局进一步改进合法化后的解。集成电路总体布局很大程度上影响了最终布局结果的质量, 因此在布局中被认为是最至关重要的部分。本节介绍集成电路总体布局问题的有代表性的优化模型和算法。

2.1 二次规划方法

基于二次规划的总体布局算法一般使用两种方法使得单元分布均匀[49], 在这些方法中, 即使有附加的约束/力, 二次线长最小化问题仍然可以建模为凸二次规划, 因而可以有效解决。第一种方法是对布局区域进行物理划分 (如 BonnPlace[25] 和 hATP[63]), 或者添加线性约束以更改一些单元的重心[16], 以防止它们聚集在一起。给定一个不均匀的二次布局解, 文 [16] 通过以下过程来改善单元的分布: 假设单元必须水平分布, 首先使用一条垂直切割线将电路划分为两个子电路, 并将布局区域划分为两个子区域。对每个子电路在 x 方向上添加一个约束, 以迫使该子电路中所有单元的重心在相应区域的中心, 重心约束会将两个子电路水平拉开。进一步递归求解具有两个附加约束的布局问题, 直到每个子电路所包含的单元数量少于预置的数量。

另一种集成线长模型和重叠处理技术的方法称为定点方法。通过重叠处理技术获得每个单元的目标位置并在该位置上创建一个固定点, 在单元以及固定点之间建立虚拟线网。然后在修改后的网表上求解布局问题, 重复扩散过程, 直到布局密度分布满足要求。根据所选择的重叠处理技术的不同, 应用于不同布局工具的定点方法之间也存在差异[4]。例如, 采用密度扩散技术[64]、泊松密度控制[18, 26, 61] 和单元移动技术[24] 的布局工具是在每个单元的目标位置上创建一个固定点, 并在它们之间建立虚拟线网; 在最小费用流指派[57] 中, 一组单元只能获得粗略的位置指引, 因此这些单元将共享相同的固定点; 此外, FastPlace[23] 仅将从单元原始位置到其目标位置的方向和距离作为参考, 然后把固定点沿与目标位置相同的方向放置在芯片边界上, 并调整虚拟线网权重。

2.2 惩罚函数法/拉格朗日乘子法

基于非凸优化的布局工具通常通过非二次线长函数来近似 HPWL, 通过非二次密度约束来控制单元扩散, 然后用惩罚函数法将由线长目标和密度约束组成的优化问题转换为无约束最小化问题。APlace[20], mPL6[21], NTUplace3[27], ePlace[38] 和 FZUplace[42] 等非二次布局工具均使用 LSE 或 WA 光滑化线长模型。其中, APlace 和 NTUplace3 使用钟形密度函数模型, mPL6 使用 Helmholtz 密度函数模型, 而 ePlace 和 FZUplace 使用电场能模型。

NTUplace3[27] 为了全面了解布局信息并有效地使用非线性规划技术, 将布局区域均匀地划分为不重叠的网格 bin。然后将总体布局问题转化为如下的约束最小化问题:

$$\begin{aligned} \min\ & W(x,y) \\ \text{s.t.}\ & D_b(x,y) = M_b, \forall \text{bin } b。 \end{aligned} \tag{5}$$

上述问题通常采用二次罚方法转化为无约束最小化问题:

$$\min \quad W(x,y)+\lambda_p\sum_b(\hat{D}_b(x,y)-M_b)^2, \tag{6}$$

其中 λ_p 是罚参数, 用于平衡线长和密度惩罚的权重, 并在算法迭代中不断更新 (例如不断增大以满足密度约束)。通常采用共轭梯度法[65] 求解上述总体布局问题, 同时为了保证求解效率, 采用如下的动态步长 α_k 以替代线搜索:

$$\alpha_k=\frac{sw_b}{||d_k||_2},$$

其中 s 是指定的参数, d_k 是第 k 次迭代的共轭方向。由于 $||\alpha_k d_k||_2^2=s^2w_b^2$, 所以该步长可以控制可移动单元的单次总移动量。

由于布局问题的超大规模特性, 直接求解上述超大规模的非线性规划问题通常非常耗时, 所以一般使用多级聚类框架来减少布局算法的计算时间[3]。该类算法首先以多级的方式对电路单元聚类, 得到一系列粗化的、规模逐渐缩小的网表。然后在粗粒度网表上解非线性规划问题, 快速生成粗粒度的布局。随后解聚类, 并以上一层的簇的位置作为初始位置, 解非线性规划问题进行布局优化。在总体布局中, 常用的聚类算法包括 First-Choice[66] (Capo, NTUplace 系列) 和 Best-Choice[67] (APlace, mPL6, FastPlace3, RQL 和 MAPLE 等)。

ePlace[38] 和 FZUplace[42] 等在总体布局中将单元视为正电荷, 并将单元分布的密度约束视为总势能约束 $N(v)$。然后根据静电平衡, 将总体布局问题转化为:

$$\begin{aligned}\min \quad & W(x,y)\\ \text{s.t.} \quad & N(v)=0。\end{aligned} \tag{7}$$

由于电荷两两之间均会产生作用力, 进而形成电势, 而系统总势能等于所有电荷对两两之间产生的势能之和, 所以 $N(v)=\frac{1}{2}\sum_{i\in V}q_i\psi_i$, 其中 $q_i=A_i$ 表示单元 i 的电量。与 (5) 式不同的是, 多个约束转化为单一的零势能约束 $N(v)=0$。通过使用一个拉格朗日乘子 λ_l, 将 (7) 式转化为无约束最小化问题:

$$\min \quad W(v)+\lambda_l N(v)。$$

使用 Nesterov 方法[68] 求解上述问题, 单元移动的方向由线长梯度和电场决定。为了快速计算及收敛, 通常利用前后两次迭代的位置来近似计算 Lipschitz 常数并估计步长:

$$\tilde{L}_k=\frac{||\nabla f(u_k)-\nabla f(u_{k-1})||}{||u_k-u_{k-1}||},$$

其中 u_k 是第 k 步迭代的参考解。同时, 根据线长的改变量动态更新乘子 λ_l。该类布局算法的速度快且效果好, 不需要结合多级框架, 甚至还可以用 GPU 加速[44]。

2.3 增广拉格朗日方法

在 VLSI 总体布局问题 (5) 中, 二次罚方法 (6) 可以随着罚参数值的加大而快速扩散单元。然而对于目标函数中的线长和密度, 随着罚参数值的加大密度约束的惩罚项可能变

得太大, 以至于线长成本对总目标值的贡献太小进而导致总线长急剧增加, 解质量下降。此外, 光滑化密度函数是非凸非线性的。因此, 随着惩罚参数值的增加, 难以通过二次罚方法来解决该约束最小化问题, 进而限制了解的质量。

增广拉格朗日方法可以减少最小化问题病态的可能性[69], 因此, 文 [70] 提出了使用增广拉格朗日方法代替惩罚函数法用于求解集成电路的总体布局问题。根据实际问题的特征, 进一步添加了衰减系数和阻尼系数对增广拉格朗日法进行修正。此外, 文中使用谨慎的动态密度权重策略来平衡线长目标和密度约束, 在共轭梯度法中使用自适应步长策略使得在初始状态能进行广泛的搜索, 并在可能接近最佳位置时缩小步长。如果没有精确线搜索, 通过增广拉格朗日方法获得的解将无法足够接近原始问题的稳定点[70]。

文 [71] 分析了四种处理约束的方法: 二次罚函数法, 拉格朗日乘子法和两种增广拉格朗日方法的优缺点, 然后针对集成电路总体布局问题提出了广义增广拉格朗日方法 (GALM) 如下:

$$\min \hat{W}(x,y) + \frac{1}{\omega}\sum_b \mu_b^k(\hat{D}_b(x,y) - M_b) + \frac{\rho}{2}\sum_b (\hat{D}_b(x,y) - M_b)^2,$$

其中 $\mu_b^{k+1} = \mu_b^k + \frac{\rho}{\omega}(\hat{D}_b - M_b)$。通过适当更新 ω 的值和拉格朗日乘子, GALM 在初始阶段不仅具有二次罚方法快速扩散单元的优点, 且在算法迭代后期保留了增广拉格朗日方法的优点, 并避免了其缺点。文中证明了 GALM 在一定条件下可以收敛到原始问题的 KKT 点。

文 [41] 将 ePlace 的模型扩展到 FPGA 总体布局以处理 FPGA 设计中的异质性模块。不同于 ePlace 使用拉格朗日乘子法来处理密度约束, 文中采用了增广拉格朗日法对每种资源类型的模块赋予不同的权重:

$$\min \hat{W}(x,y) + \sum_{s\in S}\lambda_s(\psi_s(x,y) + \frac{c_s}{2}\psi_s(x,y)^2),$$

其中 $s \in S = \{\text{LUT, FF, DSP, RAM}\}$ 对应不同的资源类型, c_s 是控制二次项的系数。为了更好地控制密度约束的总体行为, 并使算法对初始布局的敏感性降低, 二次项也受 λ_s 控制。通过正确设置参数, 当某种资源类型 s 具有高势能 $\psi_s(x,y)$ (很多重叠) 时, 二次惩罚项占主导地位以增强目标函数的凸度并增强收敛性。当某种资源类型 s 具有相对较小的势能时, 一次项占主导地位因而可以使用乘子方法继续优化, 而不会遇到与惩罚方法相关的病态问题。文中进一步使用预条件技术和基于归一化的次梯度法以更新乘子并提升算法性能。

2.4 交替方向乘子法

交替方向乘子法 (Alternating Direction Method of Multipliers, ADMM) 是一种用于具有可分离结构优化问题的有效算法, 特别是可分离线性约束凸优化问题。该方法将大的全局问题分解为多个较小、较容易求解的局部子问题, 并通过协调子问题的解而收敛到全局问题的高质量 (可行) 的解, 在集成电路总体布局中也得到了应用。

文 [72] 将传统的线长优化的集成电路总体布局问题转化为一个特殊的非凸最小化问题, 其中目标函数是两个凸线长函数的和, 而约束条件中的函数是非凸的密度函数。使用邻近点交替方向乘子法求解该问题, 并在邻近点参数的适当选择和某些假设下证明了算法生成的序列的任何子序列全局收敛到原问题的 KKT 点。

文 [73] 考虑了集成电路制造中的雾化效应和邻近效应, 将集成电路总体布局问题建模为如下具有线性约束的可分离的最小化问题, 以降低所解决问题的复杂性:

$$\begin{aligned} &\min\ \ \theta_1(x,y)+\theta_2(g,h)\\ &\text{s.t.}\ \ x-g=0,\ \ y-h=0, \end{aligned}$$

其中 $\theta_1(x,y)=\lambda_1\hat{W}(x,y)+\lambda_2\hat{D}_b(x,y)$ 表示线长和密度函数, $\theta_2(g,h)=\lambda_3 S_f(g,h)+\lambda_4 S_p(g,h)$ 表示雾化效应和邻近效应值。通过在目标函数中添加强凸的邻近项, 文中提出了一个邻近组 ADMM 来求解问题, 所提出方法的每次迭代都以较低的计算成本解决了两个子问题。第一个子问题 (主要与线长和密度相关) 通过最速下降法进行计算, 而第二个子问题 (主要与雾化和邻近效应相关) 则通过一些近似技术直接求得解析解。文中在两个温和的假设下证明了所提出的方法全局收敛到一阶临界点。

对 2.5D FPGA 布局问题, 文 [43] 考虑超长线 (SLL) 数量的优化, 提出了一个连续成本函数以解决有限离散的 SLL 问题并放松时钟约束, 然后将考虑 SLL 和时钟约束的 2.5D FPGA 总体布局问题表示为连续可微的最小化问题。为了降低所解决的优化问题的复杂性, 进一步将其重构为具有线性约束的可分离的最小化问题。用 ADMM 算法求解该问题需要考虑两个子问题, 第一个子问题包括线长、密度函数和 SLL 约束, 第二个子问题是时钟约束。使用类似的邻近组 ADMM 以交替方式有效地解决子问题, 并提出了一个平滑的罚参数更新方案, 以在解质量和计算时间之间取得较好的平衡。

文 [74] 提出了针对超导电子电路的基于 ADMM 的时序驱动的总体布局算法, 通过对电路的时序关键路径上的所有线网的最大互连长度施加硬约束, 在寻求目标时钟频率的布局同时将电路的总线长最小化。所构造的 ADMM 将时序驱动的总体布局问题分解为两个子问题, 一个子问题优化总线长并减少重叠; 另一个子问题找靠近前一个子问题的满足路径时延约束的布局解。通过迭代过程, 不断修正两个子问题的解以缩小线长驱动和时序驱动布局之间的间隙。通过在总体布局的最后阶段执行少量的 ADMM 步骤, 该算法能产生满足目标时钟频率的时序要求的高质量的解。

2.5 非光滑优化

与其他光滑化近似方法不同的是, 文 [37] 提出了一个精确计算 HPWL 的 l_1 范数线长模型。在线网只连接 2 个或 3 个引脚的情况下, l_1 范数函数恰好是 HPWL; 在线网连接 p_e ($p_e>3$) 个引脚的情况下, 也可以将 HPWL 较精确地近似为加权的 l_1 范数函数。l_1 范数线长模型表示如下:

$$W_{l_1}(x,y)=\sum_{e\in E}(\|A_e x\|_1+\|A_e y\|_1),$$

其中 $A_e=(\omega_{pq}a_{pq})$ 是超边 e 的带权连接矩阵。此外, 重叠函数是通过密度控制模型的定义 (4) 来精确计算的, 而不是光滑近似。例如, 对于 bin b 和单元 i, 令 $d_x=|x_i-x_b|$, 若 $w_b>w_i$, 则

$$\Theta_x(b,i)=\begin{cases} w_i, & \text{当}\ \ 0\leqslant d_x\leqslant\dfrac{w_b-w_i}{2};\\ \dfrac{w_b-2d_x+w_i}{2}, & \text{当}\ \ \dfrac{w_b-w_i}{2}<d_x<\dfrac{w_b+w_i}{2};\\ 0, & \text{当}\ \ \dfrac{w_b+w_i}{2}\leqslant d_x。\end{cases}$$

由于所提出的模型是非光滑的, 文中给出了容易计算的次梯度。例如, 函数 $|x_i - x_j|$ 的次梯度 g_x 计算如下:

$$g_x = \begin{cases} 1, & \text{当 } x_i > x_j; \\ \cos\theta, & \text{当 } x_i = x_j; \\ -1, & \text{当 } x_i < x_j。 \end{cases}$$

为了保持下降性, 在上一步迭代中, 如果 $x_i > x_j$, 则 $\theta \in [0, \pi/3]$; 如果 $x_i < x_j$, 则 $\theta \in [2\pi/3, \pi]$; 否则 $\theta = 0$。进一步地, 将次梯度方法与 PR 共轭梯度法[27] 结合, 提出了共轭次梯度法来求解带二次惩罚项的非光滑优化问题, 并在某些合适的条件下证明了该方法的局部收敛性。在实验中, 该非光滑优化方法还使用了用于控制步长的自适应参数以及用于增加惩罚参数值的谨慎策略以平衡计算时间和解的质量。

2.6 迭代收缩和阈值算法

文 [75] 提出了迭代收缩和阈值算法来有效地求解使用精确罚的混合非整数倍高单元的总体布局问题:

$$\min\ f(z; \lambda_1, \lambda_2) = W_L(z) + \lambda_1 W_L^p(z) + \lambda_2 \sum_k \sum_b \max\{D_{k,b}(z) - M_{k,b}, 0\}, \tag{8}$$

其中 $z := (x, y)$, $W_L^p(z)$ 是虚拟线网的线长, λ_1 随着迭代的进行从 0 增加到 1, λ_2 是罚参数。由于在混合非整数倍高单元的总体布局问题中, 区域是动态更新的, 考虑密度分布更为复杂, 所以采用精确罚来考虑约束违反的情况。由于问题 (8) 是非光滑的, 所以采用在非光滑正则化方面具有优势的迭代收缩和阈值算法 (ISTA) 进行求解。

为简单起见, 令 $\tilde{W}(z; \lambda_1) = W_L(z) + \lambda_1 W_L^p(z)$, $\tilde{D}(z; \lambda_2) = \lambda_2 \sum_k \sum_b \max\{D_{k,b}(z) - M_{k,b}, 0\}$。已知 ISTA 的 s 次迭代的解, 下一步迭代 $z_{(s+1)}$ 通过求解下述问题得到:

$$\begin{aligned} z_{(s+1)} \ = \ & \arg\min_z \ \tilde{W}(z_{(s)}; \lambda_1) + \left\langle \nabla \tilde{W}(z_{(s)}; \lambda_1), z - z_{(s)} \right\rangle \\ & + \frac{\beta_{(s)}}{2} \|z - z_{(s)}\|^2 + \tilde{D}(z; \lambda_2)。 \end{aligned} \tag{9}$$

求解问题 (9) 等价于求解如下的邻近点算子问题:

$$z_{(s+1)} = \arg\min_z \tilde{f}(z; z_{(s)}) := \frac{1}{2}\|z - u_{(s)}\|^2 + \frac{1}{\beta_{(s)}} \tilde{D}(z; \lambda_2),$$

其中 $u_{(s)} = z_{(s)} - \frac{\nabla \tilde{W}(z_{(s)}; \lambda_1)}{\beta_{(s)}}$。在每次迭代中, ISTA 沿负梯度方向下降并以步长 $\frac{1}{\beta_{(s)}}$ 移动。

2.7 基于深度学习的 GPU 加速方法

半导体行业正面临两个挑战, 即越来越大的设计以及复杂的设计约束和目标, 这些挑战导致需要更大的计算量和更多的计算资源来寻找满足所有约束的合法解决方案, 最终导致在后端设计流程中更长的设计周期和较慢的收敛速度。因此, 业界一直在寻求减少包含数十亿个晶体管的现代集成电路的设计时间和工作量, 而 GPU 加速的最新进展为高性能设计自动化带来了新的机遇[76]。

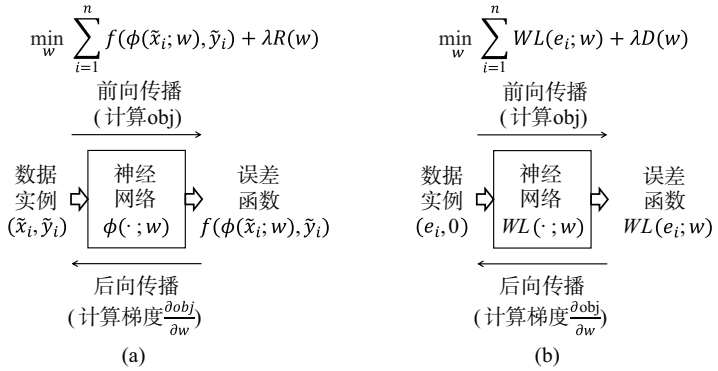

图 3 神经网络训练和基于连续优化的布局的类比
(a) 训练网络权重 w; (b) 求布局问题中的单元位置 $w = (x, y)$

文 [44] 将 ePlace[38] 的基于连续优化的布局方法等效地转换为训练神经网络, 提出了一个 GPU 加速的布局框架 DREAMPlace, 其中线长、密度函数分别类比于神经网络的预测误差和正则项, 如图 3 所示。在神经网络训练中, 每个数据用特征向量 $\tilde{x}_i$ 和标签 $\tilde{y}_i$ 标识, 且被输入神经网络后会得到一个预测的标签 $\phi(\tilde{x}_i; w)$。训练的任务是调整权重 w 使总目标最小化, 其中目标是所有训练数据的预测误差和正则项 $R(w)$ 之和。为了将集成电路单元布局与神经网络训练进行类比, 文 [44] 将单元位置 (x, y) 整合为 w, 每个数据 $(\tilde{x}_i, \tilde{y}_i)$ 都被替换为线网 e_i 和标签为零的数据实例 $(e_i, 0)$, 然后以此计算线长代价 $WL(e_i; w)$。由于线长为非负值, 所以神经网络的预测误差转化为 $\sum_{i=1}^{n} WL(e_i; w)$, 且由于密度代价 $D(w)$ 与线网实例无关, 所以可对应于正则项 $R(w)$。如此, 进一步可以按照神经网络训练过程来求解布局问题, 其中前向传播计算目标函数, 后向传播计算梯度。

广泛使用的深度学习工具包 PyTorch 和 TensorFlow 提供了成熟有效的 GPU 加速方案。基于此, DREAMPlace[44] 构造了在前向传播中快速计算线长和电势函数值以及后向传播中快速计算线长梯度及电场的方法, 并在 PyTorch 上实现。在未出现质量下降的情况下, DREAMPlace[44] 把集成电路的总体布局和合法化的速度提升了 30 倍以上, 且该框架能在一分钟内完成一百万个单元的布局设计, 同时在多达 1 000 万个单元的布局设计上保持了几乎线性的可扩展性。该方法已被扩展到详细布局[45] 以及考虑栅栏区域约束[46] 的布局等问题中。

3 布局合法化与详细布局

为了简化问题, 集成电路的总体布局在将电路单元均匀扩散的同时仍然保留部分重叠, 并且通常不与电源轨道对齐。布局合法化阶段尝试在保留总体布局后单元的相对位置的同时, 以最小的总位移消除所有重叠。常用的布局合法化算法主要有[3]: 类似于 Tetris 的贪婪合法化方法[77] 根据单元的 x 坐标对单元进行排序, 然后将单元以最小的代价从左到右/从右到左分配到其最近的合法位置; Abacus[78] 在给定单元行分配的情况下, 使用动态规划寻找单元的最佳位置; 而网络流算法试图寻找全局最优解等。

集成电路的总体布局通常使用近似线长模型, 并且单元在合法化过程中可能受到过多扰动, 因此总体布局和合法化后得到的解可能还不是很好。给定一个合法的布局解, 详

细布局可在保持解的合法性的同时局部重新排列单元来进一步改善线长[3]。例如, 分支定界法[79] 通过滑动窗口技术对每个窗口内的单元进行最佳排序, 其中窗口内所包含的单元数量是权衡计算时间与解的质量的重要因素; 文 [27] 在给定的窗口内找到一组可交换单元, 并通过将单元分配给窗口内的可用位置来构造二部图匹配问题。分配代价由将单元放置在不同位置的线长给出, 进而应用最短增广路算法[80] 求解二部图匹配问题; FastPlace-DP[81] 在不更改其他单元位置的情况下, 将每个单元移动到可用空白中的最佳位置。而当设计的利用率很高时, 尝试将单元与最佳区域内的单元交换来改善线长。下面介绍一些有代表性的布局合法化和详细布局的优化模型和算法。

3.1 单倍行高单元的布局合法化

1) 动态规划方法

在布局合法化的动态规划方法中, 通常要求单元的高度相同, 都是单倍行高单元。文 [82] 在不改变单元顺序且假定单元宽度均匀的条件下, 采用动态规划算法对单元进行布局合法化。首先将所有单元均按其期望的 y 坐标值排序。然后从最顶层的行开始对单元进行合法化, 根据单元的 y 坐标选择一个备选集, 使得备选集中单元的总面积超过所需的行容量, 接着用动态规划方法选择备选单元集的一个子集分配给该行, 原则是最小化分配到该行的成本 (距离的平方) 且满足容量约束。在行分配后, 对每行中的单元按期望的 x 坐标排序并紧凑排列。如此直到将所有单元分配到行上, 并合法化。

文 [78] 的合法化方法 Abacus 与 Tetris[77] 类似, 先将单元排序, 然后一个个将其合法化。对每个单元, 首先根据总体布局的位置移到最近的行, 然后依次将其移动到此行的上方和下方。在移动到每一行时, 先将单元根据其在总体布局中的 x 坐标进行放置, 然后与该行上已有的单元共同进行合法化以使它们的总移动量最小且互不重叠, 直到找到代价最低的行后对该单元进行最终的放置。在每一行的合法化中, Abacus 会移动在一行中已经不重叠的单元来降低所有单元的总移动量, 其原理是求解如下考虑最小移动量和无重叠约束的单行合法化问题:

$$\min \sum_{i=1}^{N} e_i (x_i - x_i')^2,$$

$$\text{s.t. } x_i \geqslant x_{i-1} + w_{i-1},$$

其中 e_i 是权重, x_i' 和 x_i 分别是合法化前后的位置, w_i 是单元 i 的宽度。该二次规划问题可以用动态规划在线性时间内求解。

2) 网络流方法

在文 [83] 中, 先用一系列重叠子区域覆盖布局区域。对每个子区域, 先找出该子区域中的所有单元, 将每个单元 μ 按宽度的最大公约数分割成大小相同的 S_μ 个子单元, 可放置的位置节点集 (与 μ 同宽) 设为 λ。添加一个源点 S 和一个汇点 D; 从 S 到 μ 连一有向边, 边的容量为 S_μ, 费用为 0; 从 μ 向 λ 中的每个节点连一有向边, 容量为 ∞, 费用是线网长度的一个估计, 该估计可以不依赖于区域中需要同时放置的单元位置计算分配到每个位置的成本。进一步, 从 λ 中的每个节点向 D 连一有向边, 费用为 0, 由于任意一个子单元只能放置在一个可用的与其他子单元不重叠的位置上, 所以边的容量为 1。如此, 把布局的合法化问题转化为求解一个最小费用最大流问题, 并按照计算结果修改单元的位置。

给定总体布局的结果, 文 [84] 提出了一个两阶段合法化方法。第一阶段将芯片细分为区域, 在区域之间移动子电路。包含过多电路的区域被定义为源, 需要被移除的量称作供给, 有空余摆放空间的区域定义为汇, 可以接受的量称为需求。构造一个最小费用流问题来决定必须移动哪些子电路, 其中相邻的两个区域用一对方向相反的有向边连接, 容量为 ∞, 费用为从一个区域移动子电路到另一个区域的代价。然后针对网络中的最小费用流, 用动态规划方法决定移动哪些子电路。第二阶段依据版图每行的单元顺序, 用一个线性时间算法寻找该行所有单元的最佳摆放位置。

3.2 混合高度单元的布局合法化

1) 模迭代方法

对于混合高度单元合法化问题, 需要在单元之间无重叠、单元对齐到行上并与电源导轨对齐的情况下最大程度地减少单元的总位移, 这是 NP-难的组合优化问题。混合高度单元合法化问题比单倍行高单元合法化问题难得多, 因为移动一个单元可能会导致其他行中的单元重叠, 并且可能会导致较大的死空间。由于复杂性高, 以前的工作都在局部区域内进行合法化从而限制了解的质量。与单倍行高单元合法化的算法相似, 给定单元顺序, 混合高度单元合法化问题可以松弛为凸二次规划问题。通常来说, 凸二次规划问题的内点法单步迭代需要立方时间, 而积极集方法单步迭代则需要二次时间, 因此用它们直接求解超大规模凸二次规划问题十分耗时。

文 [85] 对线性互补问题, 构造了基于模的矩阵分裂迭代方法 (MMSIM), 并在一定条件下证明了算法的收敛性。该方法在满足某些条件的情况下只需要求解线性系统, 从而如果可以使得线性系统具有特殊结构, 则单步迭代的计算量小。此外, 该方法的收敛速度仅取决于拆分矩阵 M 的最大和最小特征值, 以及 $M^{-1}N$ 的最大奇异值, 而不是问题规模的大小。基于此方法, 在文 [86] 中, 根据集成电路总体布局的结果将所有单元按最近原则分配到行上, 并固定单元顺序、放宽右边界约束, 将混合行高单元合法化问题松弛为如下的二次规划问题:

$$\begin{aligned} \min \quad & \frac{1}{2}x^{\mathrm{T}}Qx + p^{\mathrm{T}}x \\ \text{s.t.} \quad & Bx \geqslant b, \\ & Ex = 0, \\ & x \geqslant 0, \end{aligned} \tag{10}$$

其中 Q 是单位矩阵, $p = (-x'_1, -x'_2, \cdots, -x'_n)^{\mathrm{T}}$, x'_i 是总体布局后单元 i 的位置。上述问题的第一个约束是单元之间无重叠约束, 第二个约束表示将多倍行高单元切割成多个单倍行高单元且这些单倍行高单元的 x 坐标一致。将等式约束惩罚进目标函数, 则目标矩阵是对称正定的, 约束矩阵是行满秩的。由 KKT 条件, 可将问题 (10) 转化为如下的线性互补问题 (LCP):

$$w = Az + q \geqslant 0, \quad z \geqslant 0 \text{ 且 } z^{\mathrm{T}}w = 0,$$

其中 $w = (u, v)^{\mathrm{T}}$, $z = (x, r)^{\mathrm{T}}$, $q = (p, -b)^{\mathrm{T}}$,

$$A = \begin{bmatrix} Q + \lambda E^{\mathrm{T}}E & -B^{\mathrm{T}} \\ B & 0 \end{bmatrix}。$$

进一步地, 文 [86] 引入了一个足够小的扰动 ε 并取 $A(\varepsilon) = A + \varepsilon I$, 证明了 LCP$(q, A(\varepsilon))$ 的解接近于原始 LCP(q, A) 的解, 且平移总体布局后单元的位置使得问题 (10) 中的非负

约束在最优解处不起作用, 则可以保证 LCP$(q, A(\varepsilon))$ 与对应的 QP 问题是等价的。在此基础上, 文 [86] 构造了一个稳健的 MMSIM (RMMSIM), 在 A 是半正定的条件下 $A(\varepsilon)$ 是正定的, 因此仅需要系统矩阵为半正定即可相对容易地证明 RMMSIM 的收敛性和最优性, 并且 RMMSIM 的参数选择更加灵活。

以适当的方式对问题做稀疏化处理, 且在 LCP$(q, A(\varepsilon))$ 中拆分矩阵, 并使用 RMMSIM 方法求解, 可以大大加快计算速度。如果没有将单元放置在芯片的右边界之外, 则该方法在实际计算中可确保最优性。尤其是, RMMSIM 有效地利用了电路的稀疏特性, 因此可以非常有效地同时处理所有行中混合高度的单元, 并得到一个全局良好的合法化布局结果。

混合行高单元合法化的 MMSIM 方法已得到了一些应用。文 [87] 通过在最小植入面积约束违背的单元之间插入虚拟单元, 使用 MMSIM 求解了考虑最小植入面积的布局合法化问题; 文 [88] 将考虑单元移动和技术约束的布局合法化问题松弛为带权的凸二次规划问题, 并通过 MMSIM 有效求解; 文 [89] 将最小宽度和碎片效应的布局合法化问题松弛为二次规划问题, 并用 MMSIM 求解。上述的合法化方法只考虑了 x 方向的单元移动, 文 [90] 进一步考虑了 x 和 y 方向的单元移动, 通过分析和重构目标函数和约束, 将混合高度标准单元合法化问题转化为混合整数二次规划 (MIQP) 来考虑单元的平均移动量和最大、第二、第三移动量。通过将离散约束放宽为线性约束, 将 MIQP 转换为二次规划问题, 进一步重构为线性互补问题, 并通过 MMSIM 进行求解。

2) 网络流方法

为考虑带可布线性和围栏区域约束的混合行高单元的合法化问题, 文 [91] 先用启发式方法对混合行高单元布局合法化, 优化的目标是单元的平均移动量和最大移动量。其次, 为减少合法化过程中的最大移动量, 该文对相同宽高的单元及其所放置的位置构造一个赋权二部图匹配问题, 权重是移动量的严格递增函数。使用最小费用流算法求解该完美匹配问题, 并修正这些单元的摆放位置。然后, 在不更改单元顺序和行分配的情况下局部移动单元来进一步减少最大移动量和平均移动量。文中将此问题建模为线性规划, 随后将目标函数中 x 方向上的位移分解为一对变量, 并在约束中添加辅助变量, 进一步将问题转化为对偶线性规划, 并根据流守恒建模为最小费用流问题, 使用网络单纯形算法求解。

3.3 基于混合整数规划的详细布局

基于合法化的布局结果, 文 [92] 提出了适用于标准单元详细布局的混合整数规划模型, 其中所有单元的高度相同, 每个单元的宽度是 site 宽度的整数倍。该模型的目标是最小化半周长线长, 并根据 site 的占用情况定义 0-1 变量和构造约束。约束包括: 每个单元可以放置的矩形框与单元重心的定义; 单元占用的 site 数量与单元宽度相同; 对于占据多个 site 的单元, 添加连续性约束以确保该单元占据同一行的连续位置; 在一个特定位置没有两个单元重叠。文中使用分支-割算法求解, 所设计的批处理分支技术可以在优化过程的每个步骤中消除几个整数变量, 而定义的割平面还可以有效地优化可行区域, 从而快速生成整数解。此外, 文中还利用多核计算环境对所设计的算法并行化, 以显著减少计算时间。

4 总结和展望

集成电路布局问题是超大规模的 NP-困难组合优化问题, 现有的布局算法通常分解为总体布局、布局合法化和详细布局三个步骤。经过 30 多年的研究, 已涌现出许多集成电路总体布局和布局合法化问题的算法, 包括连续优化和组合优化算法, 其中基于连续优化的算法目前是集成电路总体布局问题的主流算法, 而布局合法化问题有连续优化和组合优化算法。但是, 当前的许多研究表明, 现有算法所求出的解与最优解还有一定距离, 这可能是由于需要在可接受的计算时间内求尽可能好的解导致的, 所以超大规模集成电路布局问题还有一定的研究空间。

线长驱动的布局是集成电路布局的基础性问题, 其求解算法的每一次较大的改进都可能对布局问题的研究起重要的推动作用。该问题经历了较长时间的研究, 应该已经较为成熟。但是, 集成电路布局还需要考虑线网的可布通性、时序、芯片的功耗等性能指标, 其中线网的可布通性和时序约束通常描述为离散优化问题, 而功耗是用微分方程描述的。因此, 在线长驱动的布局问题中加入这些因素后布局问题不仅是多目标优化问题, 而且可能混合了连续优化、离散优化和微分方程等领域。这些问题很重要, 目前已有不少的研究, 但从最优化理论与算法的角度看, 总体上还比较初步, 因此本文并未涉及。此外, 在集成电路的先进制程下, 约束条件不断增多, 设计规模不断扩大, 使得超大规模集成电路布局问题变得更加复杂, 求解更为困难, 值得深入研究。

参 考 文 献

[1] Alpert C J, Mehta D P, Sapatnekar S S. *Handbook of Algorithms for Physical Design Automation* [M]. Florida: CRC Press, 2009: 109-134.

[2] 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.

[3] Markov I L, Kim M C. Progress and challenges in VLSI placement research [C]//*Proceedings of IEEE*, 2015, **103**(11): 1985-2003.

[4] Chang Y W, Jiang Z W, Chen T C. Essential issues in analytical placement algorithms [J]. *IPSJ Transactions on System LSI Design Methodology*, 2009, **2**: 145-166.

[5] Sechen C, Sangiovanni-Vincentelli A. Timberwolf 3.2: a new standard cell placement and global routing package [C]//*Proceedings of ACM/IEEE Design Automation Conference*, 1986: 432-439.

[6] Wang M, Yang X, Sarrafzadeh M. Dragon2000: standard-cell placement tool for large industry circuits [C]//*Proceedings of IEEE/ACM International Conference on Computer Aided Design*, 2000: 260-263.

[7] Taghavi T, Yang X, Choi B K. Dragon2005: large-scale mixed-size placement tool [C]//*Proceedings of International Symposium on Physical Design*, 2005: 212-217.

[8] Breuer M. Min-cut placement [J]. *Journal of Design Automation Fault-Tolerant Computing*, 1977, **10**: 343-382.

[9] Dunlop A E, Agrawal V D, Deutsch D N, et al. Chip layout optimization using critical path weighting [C]//*Proceedings of Design Automation Conference Proceedings*, 1984: 133-136.

[10] Tsay R S, Kuh E S, Hsu C P. Proud: A fast sea-of-gates placement algorithm [C]//*Proceedings of ACM/IEEE Design Automation Conference*, 1988.

[11] Burkard R E, Karisch S E, Rendl F. QAPLIB - A quadratic assignment problem library [J]. *Journal of Global Optimization*, 1997, **10**: 391-403.

[12] Caldwell A E, Kahng A B, Markov I L. Can recursive bisection produce routable placements [C]//*Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, 2000: 477-482.

[13] Roy J A, Papa D A, Adya S N, et al. Capo: robust and scalable open-source min-cut floorplacer [C]//*Proceedings of International Symposium on Physical Design*, 2005: 224-226.
[14] Roy J A, Adya S N, Papa D A, et al. Min-cut floorplacement [J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2006, **25**(7): 1313-1326.
[15] Agnihotri A R, Ono S, Madden P H. Recursive bisection placement: feng shui 5.0 implementation detail [C]//*Proceedings of International Symposium on Physical Design*, 2005: 230-232.
[16] Kleinhans J M, Sigl G, Johannes F M, et al. GORDIAN: VLSI placement by quadratic programming and slicing optimization [J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1991, **10**(3): 356-365.
[17] Naylor W C, Donelly R, Sha L. Non-linear optimization system and method for wire length and delay optimization for an automatic electric circuit placer [P]. *US Patent 6,301,693*, 2001.
[18] Hu B, Zeng Y, Marek-Sadowska M. mFAR: Fixed-points-addition-based VLSI placement algorithm [C]//*Proceedings of International Symposium on Physical Design*, 2005: 18-25.
[19] Kahng A B, Wang Q. Implementation and extensibility of an analytic placer [J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2005, **24**(5): 734-747.
[20] Kahng A B, Wang Q. A faster implementation of APlace [C]//*Proceedings of International Symposium on Physical Design*, 2006: 218-220.
[21] Chan T F, Cong J, Shinnerl J R, Sze K, Xie M. mPL6: Enhanced multilevel mixed-size placement [C]//*Proceedings of International Symposium on Physical Design*, 2006: 212-214.
[22] Viswanathan N, Pan M, Chu C. FastPlace 2.0: An efficient analytical placer for mixed-mode designs [C]//*Proceedings of Asia and South Pacific Conference on Design Automation*, 2006: 195-200.
[23] Viswanathan N, Pan M, Chu C. FastPlace 3.0: a fast multilevel quadratic placement algorithm with placement congestion control [C]//*Proceedings of Asia and South Pacific Conference on Design Automation*, 2007: 135-140.
[24] Viswanathan N, Nam G J, Alpert C J, Villarrubia P, Ren H, Chu C. RQL: Global placement via relaxed quadratic spreading and linearization [C]//*Proceedings of ACM/IEEE Design Automation Conference*, 2007: 453-458.
[25] Brenner U, Struzyna M, Vygen J. BonnPlace: Placement of leading-edge chips by advanced combinatorial algorithms [J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2008, **27**(9): 1607-1620.
[26] Spindler P, Schlichtmann U, Johannes F M. Kraftwerk2: a fast force-directed quadratic placement approach using an accurate net model [J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2008, **27**(8): 1389-1411.
[27] Chen T C, Jiang Z W, Hsu T C, et al. NTUplace3: An analytical placer for large-scale mixed-size design with preplaced blocks and density constraints [J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2008, **27**(7): 1228-1240.
[28] Hsu M K, Chen Y F, Huang C C, et al. NTUplace4h: A novel routability-driven placement algorithm for hierarchical mixed-size circuit designs [J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2014, **33**(12): 1914-1927.
[29] Huang C C, Lee H Y, Lin B Q, et al. NTUplace4dr: A detailed-routing-driven placer for mixed-size circuit designs with technology and region constraints [J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018, **37**(3): 669-681.
[30] Chen J, Lin Z, Kuo Y C, et al. Clock-aware placement for large-scale heterogeneous FPGAs [J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020, **39**(12): 5042-5055.
[31] Kim M C, Lee D J, Markov I L. SimPL: An effective placement algorithm [J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2012, **31**(1): 50-60.
[32] Kim M C, Lee D J, Markov I L. SimPL: An algorithm for placing VLSI circuits [J]. *Communication of the ACM*, 2013, **56**(6): 105-113.
[33] Kim M C, Viswanathan N, Alpert C J, et al. MAPLE: Multilevel adaptive placement for mixed-size designs [C]//*Proceedings of ACM International Symposium on Physical Design*, 2012: 193-200.

[34] Chen J, Zhu W. An analytical placer for VLSI standard cell placement [J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2012, **31**(8): 1208-1221.

[35] Lin T, Chu C. POLAR 2.0: An effective routability-driven placer [C]//*Proceedings of ACM/ESDA/IEEE Design Automation Conference*, 2014: 1-6.

[36] Lin T, Chu C, Shinnerl J R, et al. POLAR: a high performance mixed-size wirelength-driven placer with density constraints [J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2015, **34**(3): 447-459.

[37] Zhu W, Chen J, Peng Z, et al. Nonsmooth optimization method for VLSI global placement [J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2015, **34**(4): 642-655.

[38] Lu J, Chen P, Chang C C, et al. ePlace: electrostatics based placement using fast fourier transform and Nesterov's method [J]. *ACM Transactions on Design Automation of Electronic Systems*, 2015, **20**(2): 1-34.

[39] Lu J, Zhuang H, Chen P, et al. ePlace-MS: Electrostatics based placement for mixed-size circuits [J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2015, **34**(5): 685-698.

[40] Cheng C K, Kahng A B, Kang I, et al. RePlAce: Advancing solution quality and routability validation in global placement [J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018, **38**(9): 1717-1730.

[41] Li W, Lin Y, Pan D Z. elfPlace: Electrostatics-based placement for large-scale heterogeneous FPGAs [C]//*Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, 2019: 1-8.

[42] Zhu W, Huang Z, Chen J, et al. Analytical solution of Poisson's equation and its application to VLSI global placement [C]//*Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, 2018.

[43] Chen J, Zhu W, Yu J, et al. Analytical placement with 3D Poisson's equation and ADMM based optimization for large-scale 2.5D heterogeneous FPGAs [C]//*Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, 2019.

[44] Lin Y, Dhar S, Li W, et al. DREAMPIace: Deep learning toolkit-enabled GPU acceleration for modern VLSI placement [C]//*Proceedings of ACM/IEEE Design Automation Conference*, 2019.

[45] Lin Y, Pan D Z, Ren H, et al. DREAMPlace 2.0: Open-source GPU-accelerated global and detailed placement for large-scale VLSI designs [C]//*Proceedings of China Semiconductor Technology International Conference*, 2020.

[46] Gu J, Jiang Z, Lin Y, et al. DREAMPlace 3.0: Multi-electrostatics based robust VLSI placement with region constraints [C]//*Proceedings of IEEE/ACM International Conference On Computer Aided Design*, 2020.

[47] Nam G J, Cong J. *Modern Circuit Placement: Best Practices and Results* [M]. New York, NY, USA: Springer-Verlag, 2007.

[48] Kahng A B, Lee H, Li J. Horizontal benchmark extension for improved assessment of physical CAD research [C]//*Proceedings of Great Lakes Symposium on VLSI*, 2014: 27-32.

[49] Chu C. *Placement, in electronic design automation: synthesis, verification, and testing* [M]. Elsevier/Morgan Kaufmann, 2009: 635-686.

[50] Viswanathan N, Chu C C N. Fastplace: Efficient analytical placement using cell shifting, iterative local refinement and a hybrid net model [J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2005, **24**(5): 722-733.

[51] Hsu M K, Chang Y W, Balabanov V. TSV-aware analytical placement for 3D IC design [C]//*Proceedings of ACM/IEEE Design Automation Conference*, 2011: 664-669.

[52] Chan T, Cong J, Sze K. Multilevel generalized force-directed method for circuit placement [C]//*Proceedings of International Symposium on Physical Design*, 2005: 185-192.

[53] Brenner U, Struzyna M. Faster and better global placement by a new transportation algorithm [C]//*Proceedings of ACM/IEEE Design Automation Conference*, 2005: 591-596.

[54] Fiduccia C M, Mattheyses R M. A linear-time heuristic for improving network partitions [C]//*Proceedings of ACM/IEEE Design Automation Conference*, 1982: 175-181.
[55] Huang D J H, Kahng A B. Partitioning-based standard-cell global placement with an exact objective [C]//*Proceedings of ACM International Symposium on Physical Design*, 1997: 18-25.
[56] Vygen J. Algorithms for large-scale flat placement [C]//*Proceedings of ACM/IEEE Design Automation Conference*, 1997: 746-751.
[57] Agnihotri A R, Madden P H. Fast analytic placement using minimum cost flow [C]//*Proceedings of IEEE/ACM Asia South Pacific Design Automation Conference*, 2007: 128-134.
[58] Ren H, Pan D Z, Alpert C J, et al. Diffusion-based placement migration with application on legalization [J]. *IEEE Transations on Computer-Aided Design of Integrated Circuits and Systems*, 2007, **26**(12): 2158-2172.
[59] Press W H, Teukolsky S A, Vetterling W T, et al. *Numerical recipes in C++* [M]. Cambridge University Press, 2002.
[60] Yao B, Chen H, Cheng C K, et al. Unified quadratic programming approach for mixed mode placement [C]//*Proceedings of ACM International Symposium on Physical Design*, 2005: 193-199.
[61] Vorwerk K, Kennings A, Vannelli A. Engineering details of a stable force-directed placer [C]//*Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, 2004: 573-580.
[62] Kowarschik M, Weiβ C. DiMEPACK — A cache-optimized multigrid library [C]//*Proceedings of Conference on Parallel and Distributed Processing Techniques and Applications*, 2001: 425-430.
[63] Nam G J, Reda S, Alpert C J, et al. A fast hierarchical quadratic placement algorithm [J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2006, **25**(4): 678 – 691.
[64] Luo T, Pan D Z. DPlace2.0: A stable and efficient analytical placement based on diffusion [C]//*Proceedings of IEEE/ACM Asia South Pacific Design Automation Conference*, 2008: 346-351.
[65] Luenberger D G. *Linear and Nonlinear Programming* [M]. Addison Wesley, Reading, MA, 1984.
[66] Karypis G, Kumar V. Multilevel k-way hypergraph partitioning [C]//*Proceedings of ACM/IEEE Design Automation Conference*, 1999: 343-348.
[67] Alpert C J, Kahng A B, Nam G J, et al. A semi-persistent clustering technique for VLSI circuit placement [C]//*Proceedings of International Symposium on Physical Design*, 2005: 200-207.
[68] Nesterov Y E. A method of solving a convex programming problem with convergence rate $O(1/k^2)$ [J]. *Dokl. Akad. Nauk SSSR*, 1983, **269**(3): 543-547.
[69] Hestenes M R. Multiplier and gradient methods [J]. *Journal of Optimization Theory and Applications*, 1969, **4**(5): 303-320.
[70] Zhu W, Chen J, Li W. An augmented Lagrangian method for VLSI global placement [J]. *The Journal of Supercomputing*, 2014, **69**: 714-738.
[71] Zhu Z, Chen J, Peng Z, et al. Generalized augmented Lagrangian and its applications to VLSI global placement [C]//*Proceedings of ACM/ESDA/IEEE Design Automation Conference*, 2018.
[72] Peng Z, Chen J, Zhu W. A proximal alternating direction method of multipliers for a minimization problem with nonconvex constraints [J]. *Journal of Global Optimization*, 2015, **62**: 711-728.
[73] Chen J, Yang L, Peng Z, et al. Novel proximal group ADMM for placement considering fogging and proximity effects [C]//*Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, 2018.
[74] Shahsavani S N, Pedram M. TDP-ADMM: A timing driven placement approach for superconductive electronic circuits using alternating direction method of multipliers [C]//*Proceedings of ACM/IEEE Design Automation Conference*, 2020.

[75] Chen J, Huang Z, Huang Y, et al. An efficient EPIST algorithm for global placement with non-integer multiple-height cells [C]// ACM/IEEE Design Automation Conference, 2020.

[76] Lin Y. GPU acceleration in VLSI back-end design: overview and case studies [C]//*Proceedings of IEEE/ACM International Conference On Computer Aided Design*, 2020.

[77] Hill D. Method and system for high speed detailed placement of cells within an intergrated circuit design [P]. *US patent 6,370,673*, 2002.

[78] Spindler P, Schlichtmann U, Johannes F M. Abacus: Fast legalization of standard cell circuits with minimal movement [C]//*Proceedings of International Symposium on Physical Design*, 2008: 47-53.

[79] Caldwell A E, Kahng A B, Markov I L. Optimal partitioners and end-case placers for standard-cell layout [J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2000, **19**(11): 1304-1313.

[80] Jonker R, Volgenant A. A shortest augmenting path algorithm for dense and sparse linear assignment problems [J]. *Computing*, 1987, **38**(4): 325-340.

[81] Pan M, Viswanathan N, Chu C. An efficient and effective detailed placement algorithm [C]//*Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, 2005: 48-55.

[82] Agnihotri A, Yildiz M C, Khatkhate A, et al. Fractional cut: improved recursive bisection placement [C]//*Proceedings of IEEE International Conference on Computer-Aided Design*, 2003: 307-310.

[83] Doll K, Johannes F M, Antreich K J. Iterative placement improvement by network flow methods [J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1994, **13**(10): 1189-1200.

[84] Brenner U, Vygen J. Legalizing a placement with mimimum total movement [J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2004, **23**(12): 1597-1613.

[85] Bai Z Z. Modulus-based matrix splitting iteration methods for linear complementarity problems [J]. *Numerical Linear Algebra with Applications*, 2010, **17**(6): 917-933.

[86] Chen J, Zhu Z, Zhu W, et al. A robust modulus-based matrix splitting iteration method for mixed-cell-height circuit legalization [J]. *ACM Transactions on Design Automation of Electronic Systems*, 2021, **26**(2): 1-28.

[87] Chen J, Yang P, Li X, et al. Mixed-cell-height placement with complex minimum-implant-area constraints [C]//*Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, 2018.

[88] Zhu Z, Chen J, Zhu W, et al. Mixed-cell-height legalization considering technology and region constraints [J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020, **39**(12): 5128-5141.

[89] Zhu Z, Huang Z, Yang Peng, et al. Mixed-cell-height legalization considering complex minimum width constraints and half-row fragmentation effect [J]. *Integration*, 2020, **71**: 1-10.

[90] Li X, Chen J, Zhu W, et al. Analytical mixed-cell-height legalization considering average and maximum movement minimization [C]//*Proceedings of International Symposium on Physical Design*, 2019: 27-34.

[91] Li H, Chow W K, Chen G, et al. Routability-driven and fence-aware legalization for mixed-cell-height circuits [C]//*Proceedings of ACM/ESDA/IEEE Design Automation Conference*, 2018.

[92] Cauley S, Balakrishnan V, Hu Y C, et al. A parallel branch-and-cut approach for detailed placement [J]. *ACM Transactions on Design Automation of Electronic Systems*, 2011, **16**(2): 1-19.