PeF: Poisson's Equation-Based Large-Scale Fixed-Outline Floorplanning

Ximeng Li, Keyu Peng, Fuxing Huang, and Wenxing Zhu

***Abstract*—Floorplanning is the first stage of VLSI physical design. An effective floorplanning engine definitely has a positive impact on chip design speed, quality, and performance. In this article, we present a novel mathematical model to characterize nonoverlapping of modules, and propose a flat fixed-outline floorplanning algorithm based on the VLSI global placement approach using Poisson's equation. The algorithm consists of global floorplanning and legalization phases. In global floorplanning, we redefine the potential energy of each module based on the novel mathematical model for characterizing nonoverlapping of modules and an analytical solution of Poisson's equation. In this scheme, the widths of soft modules appear as variables in the energy function and can be optimized. Moreover, we design a fast approximate computation scheme for partial derivatives of the potential energy. In legalization, based on the defined horizontal and vertical constraint graphs, we eliminate overlaps between modules remained after global floorplanning, by modifying relative positions of modules. Experiments on the MCNC, GSRC, HB+, and ami49_x benchmarks show that, our algorithm improves the average wirelength by at least 2% and 5% on small and large-scale benchmarks with certain whitespace, respectively, compared to state-of-the-art floorplanners.**

***Index Terms*—Constraint graph, fixed-outline floorplanning, global floorplanning, legalization, Poisson's equation.**

I. Introduction

FLOORPLANNING is the first stage of VLSI physical design flow. With widespread usage of IP cores and huge number of cells integrated in modern chips, floorplanning needs to handle hundreds or even thousands of circuit modules within a fixed outline. Especially, some more complex issues need to be considered during floorplanning, such as timing, thermal, power, and voltage island [1], [2]. However, it is well known that the fundamental floorplanning problem is NP-hard. Hence, designing a high-performance algorithm for the fundamental large-scale floorplanning is a challenge, which will have an impact on the performance and yield of the final ICs. In this article, we focus on designing a flat analytical engine for the fixed-outline floorplanning.

Fixed-outline floorplanning can be described as placing a set of rectangular modules into a fixed rectangular area without overlap, while minimizing the total wirelength among the modules. Floorplanning usually includes two types of circuit modules: 1) hard modules and 2) soft modules. The width and height of a soft module can be changed within a certain range meanwhile maintaining a certain area. So far, works for fixed-outline floorplanning can be divided into three categories: 1) heuristic methods for a small scale problem; 2) multilevel heuristic methods for a large-scale problem; and 3) analytical approaches.

TABLE I
Summary of Floorplan Representations

Representation	Solution Space	Packing Time	Flexibility
Normalized Polish Expression	$O(n!2^{2.6n}/n^{1.5})$	$O(n)$	slicing
Corner Block List [3]	$O(n!2^{3n})$	$O(n)$	Mosaic
Twin Binary Sequence [4]	$O(n!2^{3n}/n^{1.5})$	$O(n)$	Mosaic
O-tree [5]	$O(n!2^{2n}/n^{1.5})$	$O(n)$	compacted
B*-tree [6]	$O(n!2^{2n}/n^{1.5})$	$O(n)$	compacted
Corner Sequence [7]	$\leq (n!)^2$	$O(n)$	compacted
Sequence Pair [8]	$(n!)^2$	$O(n^2)$	general
TCG-S [9]	$(n!)^2$	$O(n \log n)$	general
ACG [10]	$O((n!)^2)$	$O(n^2)$	general

Heuristic methods for small-scale fixed-outline floorplanning problem first adopt a floorplan representation, then use heuristic algorithms to search for an optimal solution in the representation space, and finally decode the optimal representation to the corresponding floorplan. Floorplan representation method also determines the complexity of transforming a floorplan into its representation, and the complexity of decoding a representation into its corresponding floorplan. Floorplan representation methods can be summarized in Table I.

In Table I, O-tree [5] and B*-tree [6] are two commonly used floorplan representations, followed by the most preferred simulated annealing algorithm. Parquet [11] used sequence pair as the representation, on which a floorplan is also optimized using simulated annealing. Since the running time of this kind of algorithms grows very fast with the problem scale, they cannot be applied to large-scale floorplanning directly. A methodology to cope with this issue is the multilevel framework with the heuristic search methods.

The core idea of multilevel heuristic search methods is reducing the size of solution space multilevelly by clustering or partitioning, and thus can handle the large-scale problem. Clustering-based floorplanning methods, e.g., MB*-tree [12], merge modules from the bottom to up, until the number of modules is small enough, and then solve the floorplanning problem. The solution is refined in each level of de-clustering, leading finally to a solution of the original problem. A disadvantage of clustering-based floorplanning is that, modules cannot be clustered from a global perspective in the early

Manuscript received 18 April 2022; revised 21 August 2022; accepted 7 October 2022. Date of publication 11 October 2022; date of current version 19 May 2023. This work was supported by the National Natural Science Foundation of China under Grant 62174033. This article was recommended by Associate Editor C. Zhuo. *(Corresponding author: Wenxing Zhu.)*

The authors are with the Center for Discrete Mathematics and Theoretical Computer Science, Fuzhou University, Fuzhou 350108, China (e-mail: wxzhu@fzu.edu.cn).

Digital Object Identifier 10.1109/TCAD.2022.3213609

TABLE II
SUMMARY OF ANALYTICAL FLOORPLANNING METHODS

method	global floorplanning	legalization
Analytical [18]	bell-shaped smoothing and Density control	function *pl*2*sp*() in Parquet-4 [11]
F-FM [1]	placement model based on NTUplace [21]	gradual partition based on position and area + ST-Tree [22]
Ref. [2]	placement model based on NTUplace [21]	SAINT [23]
AR [19]	Attractor-Repeller model	SOCP based on relative position matrix
UFO [20]	Push-Pull model	SOCP based on constraint graph

clustering stage, which limits the solution quality. Instead, partitioning-based methods decompose the original large-scale floorplanning problem into several small-scale subproblems, until they are small enough and solved directly. Finally, solutions of subproblems are merged and optimized level by level, leading to a solution of the original problem. Capo [13], DeFer [14], IMF [15], and QinFer [16] are state-of-the-art partitioning-based floorplanning methods. Most of them use hMetis [17] for the partitioning. Compared to clustering-based floorplanning methods, partitioning-based ones can consider wirelength information in the early floorplanning stage. However, they cannot accurately calculate the wirelength in the partitioning stage.

Since floorplanning is similar to the VLSI placement problem in some degree, a number of analytical floorplanning methods have been proposed based on the frameworks of VLSI placement approaches. They include analytical [18], AR [19], UFO [20], F-FM [1], etc. Generally, these methods analogously have the global floorplanning stage which determines "rough" positions of modules, and a legalization stage which eliminates the overlaps between modules and determines the exact positions of all modules and the widths of soft modules. Furthermore, combining with a multilevel framework, these analytical methods can solve large-scale floorplanning problems well. Table II summarizes the global floorplanning and legalization stages of the above methods, respectively.

Floorplanners analytical [18], F-FM [1], and Lin et al. [2] use density-control in the global floorplanning stage. In analytical [18], cell density is calculated according to the current positions and widths of modules, in which soft modules are treated as hard ones. Then, the density function is smoothed using the bell-shaped function. During legalization, overlaps between modules are removed using the function *pl*2*sp*() in Parquet-4 [11] to get a legal floorplan. Similarly, F-FM [1] and Lin et al. [2] treat all soft modules as hard blocks, and use global placement for global floorplanning. In legalization, F-FM [1] uses the ST-tree [22] with the gradual partition based on positions and areas of modules, and then adopts the simulated annealing algorithm to optimize the positions and widths of modules. As for the floorplanner in Lin et al. [2], SAINT [23] is used for legalization. The main idea is using polygon to approximate the curve of area of every soft module, and building an ILP model to achieve legalization of modules.

Earlier floorplanners AR [19] and UFO [20] directly regard all soft and hard modules as squares, and further abstract them as circles. Then, Attractor-Repeller and Push-Pull models are proposed for global floorplanning, respectively. Both of them use second-order cone programming (SOCP) to obtain legal solutions in the legalization stage. The difference is that, AR uses a relative position matrix, while UFO uses constraint graph. It must be remarked that solving SOCP is rather time-consuming when the problem size is moderate or large.

Note that floorplanning problem contains hard and soft modules. However, state-of-the-art analytical floorplanners analytical [18], F-FM [1], and Lin et al. [2] take soft modules as hard ones in global floorplanning. Moreover, they must be incorporated in the multilevel framework to handle large-scale floorplanning problem. Apparently, these will affect the solution quality.

In this article, we focus on the fundamental fixed-outline floorplanning problem. This is because algorithms for floorplanning with complex issues are extended from those for the fundamental floorplanning. Notice that Poisson's equation-based VLSI global placement methods [24], [25] do not use multilevel framework. They are flat while achieve state-of-the-art results. In this article, we will propose a high-performance floorplanning engine by extending this type of methods to large-scale fixed-outline floorplanning with hard and soft modules.

To achieve this goal, we must not only consider optimizing the shapes of soft modules in global floorplanning, but also design a high-performance legalization algorithm correspondingly. Contributions of this article are summarized as follows.

1) We propose a novel mathematical model to characterize nonoverlapping of modules. Then under the concept of electrostatic system [24], we redefine the potential energy of modules based on the analytical solution of Poisson's equation in [25]. This allows the widths of soft modules to appear as variables in the established fixed-outline global floorplanning model, and the derivative with respect to the widths of soft modules can be easily obtained.
2) We give a fast approximate computation scheme for the partial derivative of the potential energy, and design a fixed-outline global floorplanning algorithm based on the global placement method using Poisson's equation.
3) By defining horizontal and vertical constraint graphs, we present a fast legalization algorithm to remove overlaps between modules after global floorplanning. The algorithm modifies relative positions of modules and compresses the floorplan to obtain finally a legal solution.
4) Within acceptable running time, our floorplanning algorithm obtains better results on large-scale benchmarks. The average wirelength is at least 5% less than state-of-the-art multilevel heuristic search algorithms, and is at least 8% less than state-of-the-art analytical floorplanning algorithms. On small-scale benchmarks, it has similar improvements.

The remainder of this article is organized as follows. Section II is the preliminaries. Section III proposes a novel mathematical model to characterize nonoverlapping of modules, gives our global floorplanning model based on Poisson's equation, and provides a fast computation scheme for the partial derivatives. In Section IV, we detail our fixed-outline global floorplanning algorithm and legalization algorithm. Experimental results and comparisons are presented in Section V, and conclusion is summarized in Section VI.

II. Preliminaries

This section presents the problem of fixed-outline floorplanning, introduces the wirelength model and principle of Poisson's equation method for handling modules overlap in VLSI global placement.

A. Fixed-Outline Floorplanning Problem

The floorplanning area is a rectangle with four edges parallel to the coordinate axes, with the lower left corner at the origin, and the upper right corner at (W, H). Let $V = V_s \cup V_h$ be the set of rectangular modules, where V_s and V_h represent the sets of soft and hard modules, respectively. For each module $v_i \in V$, its width, height, area, and the coordinate of the center point are w_i, h_i, A_i, and (x_i, y_i), respectively. For each module $v_i \in V_s$, the aspect ratio, calculated as the ratio of height to width, is between AR_i^l and AR_i^u. Moreover, let E be the set of nets e_i, $i = 1, 2, \ldots, m$, for which $WL(e_i)$ is the half-perimeter wirelength. Let $WL(E) = \sum_{e_i \in E} WL(e_i)$. Then the fixed-outline floorplanning problem can be described as follows:

$$\begin{aligned} \min\ & WL(E) \\ \text{s.t.}\ & \text{overlap} = 0 \\ & \text{aspect ratio is suitable} \\ & \text{all modules are within the fixed-outline.} \end{aligned} \tag{1}$$

In this article, the objective of model (1) adopts the Log-Sum-Exp (LSE) wirelength function. Moreover, the first constraint in (1) ensures that all modules do not overlap. The second constraint means that the aspect ratio of every soft module meets the upper and lower bounds. The third constraint indicates that all modules are located within the fixed outline.

B. VLSI Global Placement Based on Poisson's Equation

ePlace [24] is a representative VLSI global placement method. In ePlace, each module v_i is regarded as a positively charged particle with electric quantity q_i, which is proportional to its area A_i. Thus, the placement problem is modeled as a 2-D electrostatic system, in which the electric potential $\psi(x, y)$ and the electric field $\boldsymbol{\xi}(x, y)$ satisfy that $\boldsymbol{\xi}(x, y) = (\xi_x, \xi_y) = -\nabla\psi(x, y)$. ePlace [24] characterizes $\psi(x, y)$ as the solution of Poisson's equation

$$\begin{cases} \nabla^2\psi(x, y) = -\rho(x, y), & (x, y) \in \boldsymbol{R} \quad \text{(2a)} \\ \hat{\boldsymbol{n}}\nabla\psi(x, y) = 0, & (x, y) \in \partial\boldsymbol{R} \quad \text{(2b)} \\ \iint_{\boldsymbol{R}} \rho(x, y)\mathrm{d}x\mathrm{d}y = \iint_{\boldsymbol{R}} \psi(x, y)\mathrm{d}x\mathrm{d}y = 0 & \text{(2c)} \end{cases}$$

where $\boldsymbol{R}$ is the rectangular placement area. Equation (2a) is Poisson's equation, $\rho(x, y)$ is the charge density function on $\boldsymbol{R}$. Equation (2b) is the Neumann boundary condition, where $\partial\boldsymbol{R}$ and $\hat{\boldsymbol{n}}$ represent the boundary of area $\boldsymbol{R}$ and the outer normal vector of the boundary, respectively. Equation (2c) is the compatibility condition to ensure that the equation has a unique solution.

The electric forces $\boldsymbol{F_i} = q_i\boldsymbol{\xi(x_i, y_i)}$, $i = 1, 2, \ldots, n$, guide the modules movement and henceforth reduce modules overlaps. Define the system potential energy $N(V) = \sum_{v_i \in V} N(v_i)$, where $N(v_i) = q_i\psi(v_i)$ is the potential energy of module v_i. Then by ePlace [24], a necessary condition for *overlap* $= 0$ is that the system potential energy $N(V)$ is the smallest. This allows to use the regularizing term $N(V)$ to transform the VLSI global placement problem into

$$\begin{aligned} \min\ & \mathrm{LSE}(E) + \lambda N(V) \\ \text{s.t.}\ & \text{all modules are in the placement area.} \end{aligned} \tag{3}$$

Zhu et al. [25] defines the density function

$$\begin{aligned} \rho(x, y) &= \sum_{v_i \in V} \rho_i(x, y), \quad (x, y) \in [0, W] \times [0, H] \\ \rho_i(x, y) &= \begin{cases} 1, & \text{if } (x, y) \in R_i \\ 0, & \text{else} \end{cases} \end{aligned} \tag{4}$$

where $R_i = [x_i - (w_i/2), x_i + (w_i/2)] \times [y_i - (h_i/2), y_i + (h_i/2)]$, $h_i = (A_i/w_i)$. Then, by redefining

$$\rho(x, y) \triangleq \rho(x, y) - \frac{1}{WH}\iint_{\boldsymbol{R}} \rho(x, y)\mathrm{d}x\mathrm{d}y.$$

Zhu et al. [25] obtains an analytical solution of (2). The truncated version is

$$\psi(x, y) = \sum_{u=0}^{K}\sum_{p=0}^{K} a_{up}\cos\left(\frac{u\pi}{W}x\right)\cos\left(\frac{p\pi}{H}y\right) \tag{5}$$

where K is the truncation factor that controls the precision.

For fast global placement, assume that the placement area is divided in to $K \times K$ bins, and the density of each grid is

$$\hat{\rho}(i, j) = \sum_{v_k \in V} \frac{\mathrm{Area}\left(\mathrm{bin}_{i,j} \cap R_k\right)}{\mathrm{Area}\left(\mathrm{bin}_{i,j}\right)}, \quad i,\ j = 0, 1, \ldots, K - 1. \tag{6}$$

By a similar redefinition, Zhu et al. [25] obtained an analytical solution of (2), for which the truncated solution value at $(i + (1/2), j + (1/2))$ is

$$\hat{\psi}(i, j) = \sum_{u,p=0}^{K-1} a_{up}\cos\left(\frac{u\left(i + \frac{1}{2}\right)\pi}{K}\right)\cos\left(\frac{p\left(j + \frac{1}{2}\right)\pi}{K}\right). \tag{7}$$

The coefficient calculations for (5) and (7) can be found in [25].

III. Global Floorplanning Model and Gradient Calculation

In this article, the fixed-outline floorplanning is decomposed into two stages: 1) global floorplanning and 2) legalization. In global floorplanning, partial overlaps between modules are allowed and a rough floorplanning is built. In legalization, the overlaps are eliminated and the final legal floorplan is obtained. To be able to optimize the shapes of soft modules in global floorplanning, we propose in this section a novel mathematical model to characterize nonoverlapping of modules, and then modify the VLSI global placement model (3) for fixed-outline global floorplanning. Based on the global floorplanning model, we present a fast approximate scheme for calculating partial derivatives of potential energy.

A. Global Floorplanning Model

First, we present a sufficient and necessary condition for nonoverlapping of modules. According to (4), we can prove the following results. The details will be clarified in the Appendix.

Lemma 1: Any two different modules v_i and $v_j \in V$ are nonoverlapping if and only if $\iint_{R_i} \rho_j(u, v)\mathrm{d}u\mathrm{d}v = 0$.

Let Area(R_i) be the area of module v_i, we have the following lemma.

Lemma 2: $\iint_{R_i} \rho_i(u, v)\mathrm{d}u\mathrm{d}v = \text{Area}(R_i)$ for all modules $v_i \in V$.

Theorem 1: Any module $v_i \in V$ is nonoverlapping with other modules if and only if

$$\iint_{R_i} \sum_{v_j \in V} \rho_j(u, v)\mathrm{d}u\mathrm{d}v = \text{Area}(R_i). \tag{8}$$

According to Lemma 2 and Theorem 1, it is obvious that

$$\iint_{R_i} \sum_{v_j \in V} \rho_j(u, v)\mathrm{d}u\mathrm{d}v \geq \text{Area}(R_i).$$

Moreover, by Theorem 1, the nonoverlapping constraint in (1) is equivalent to (8). Therefore, by the regularization technique in machine learning [26], the fixed-outline floorplanning problem (1) can be transformed to

$$\begin{aligned} \min \quad & WL(E) + \lambda \sum_{v_i \in V} \iint_{R_i} \sum_{v_j \in V} \rho_j(u, v)\mathrm{d}u\mathrm{d}v \\ \text{s.t.} \quad & \text{aspect ratio is suitable} \\ & \text{all modules are within the fixed-outline.} \end{aligned} \tag{9}$$

In the floorplanning model (9), $\sum_{v_j \in V} \rho_j(u, v)$ is interpreted as the density function within the floorplanning region. According to VLSI global placement, there are three optional methods to relax the density function for facilitating optimization as follows.

1) Divide the floorplanning area into bins, smooth the corresponding density function with the Bell-shaped function, and then make integration.
2) Divide the floorplanning area into bins, smooth the corresponding density function with Poisson's equation, and then make integration.
3) Make integration of $\sum_{v_j \in V} \rho_j(u, v)$ directly.

In this article, we use scheme 2) since Poisson's equation method has achieved state-of-the-art results in VLSI global placement. Let $\psi(u, v)$ be the function given by Poisson's equation (2) to smooth the density function $\sum_{v_j \in V} \rho_j(u, v)$ defined in (5). Suggested by (9), for each module $v_i \in V$, we define the potential energy N_i as follows:

$$N_i = \iint_{R_i} \psi(u, v)\mathrm{d}u\mathrm{d}v \tag{10}$$

where the integration area $R_i = [x_i - (w_i/2), x_i + (w_i/2)] \times [y_i - (h_i/2), y_i + (h_i/2)]$, $h_i = (A_i/w_i)$.

Observing (10), for a soft module, the potential energy N_i is a function of variables $x_i, y_i,$ and w_i. While, for a hard module, the potential energy N_i is a function of variables x_i and y_i. Therefore, for convenience we denote

$$\begin{aligned} g_i(x_i, y_i, w_i) = N_i &= \iint_{R_i} \psi(u, v)\mathrm{d}u\mathrm{d}v \ \forall v_i \in V_s \\ g_i(x_i, y_i) = N_i &= \iint_{R_i} \psi(u, v)\mathrm{d}u\mathrm{d}v \ \forall v_i \in V_h. \end{aligned} \tag{11}$$

According to (11) and the fixed-outline floorplanning problem (9), we can establish the global floorplanning model

$$\min_{x,y,w} \ \text{LSE}(E) + \lambda \left[\sum_{v_i \in V_s} g_i(x_i, y_i, w_i) + \sum_{v_i \in V_h} g_i(x_i, y_i) \right] \tag{12a}$$

$$\text{s.t.} \ AR_i^l \leq \frac{A_i}{w_i^2} \leq AR_i^u \quad \forall v_i \in V_s \tag{12b}$$

$$\frac{w_i}{2} \leq x_i \leq W - \frac{w_i}{2} \quad \forall v_i \in V \tag{12c}$$

$$\frac{h_i}{2} \leq y_i \leq H - \frac{h_i}{2} \quad \forall v_i \in V. \tag{12d}$$

In (12), the first constraint ensures the aspect ratio of each soft module be between the upper and lower bounds. The last two constraints indicate that all modules are located within the floorplanning area.

In order to use a nonlinear optimization method to solve problem (12), the partial derivative of the objective function with respect to each variable needs to be calculated. Since the wirelength function LSE(E) is in analytical form, the partial derivative is easy to calculate. Moreover, the difference between the potential energy functions of soft modules and hard modules is only in the variable w_i. Therefore, we only show calculations of the partial derivatives of potential energy functions of soft modules in the next section.

B. Partial Derivative of Potential Energy

In order to calculate the partial derivative of $g_i(x_i, y_i, w_i)$, first we need to calculate $g_i(x_i, y_i, w_i)$ itself. According to Section II-B, Poisson's equation has the solution as in (5). Substituting it into (11) can get

$$\begin{aligned} g_i(x_i, y_i, w_i) &= \iint_{R_i} \psi(z, v)\mathrm{d}z\mathrm{d}v \\ &= \sum_{u=0}^{K} \sum_{p=0}^{K} a_{u,p} \int_{x_i - \frac{w_i}{2}}^{x_i + \frac{w_i}{2}} \cos\left(\frac{u\pi}{W} z\right)\mathrm{d}z \int_{y_i - \frac{h_i}{2}}^{y_i + \frac{h_i}{2}} \cos\left(\frac{p\pi}{H} v\right)\mathrm{d}v \\ &= \sum_{u=1}^{K} \sum_{p=1}^{K} a_{u,p} \frac{WH}{up\pi^2} \left[\sin \frac{u\pi\left(x_i + \frac{w_i}{2}\right)}{W} - \sin \frac{u\pi\left(x_i - \frac{w_i}{2}\right)}{W} \right] \\ &\quad \left[\sin \frac{p\pi\left(y_i + \frac{h_i}{2}\right)}{H} - \sin \frac{p\pi\left(y_i - \frac{h_i}{2}\right)}{H} \right] \\ &\quad + \sum_{u=1}^{K} a_{u,0} \frac{Wh_i}{u\pi} \left[\sin \frac{u\pi\left(x_i + \frac{w_i}{2}\right)}{W} - \sin \frac{u\pi\left(x_i - \frac{w_i}{2}\right)}{W} \right] \\ &\quad + \sum_{p=1}^{K} a_{0,p} \frac{Hw_i}{p\pi} \left[\sin \frac{p\pi\left(y_i + \frac{h_i}{2}\right)}{H} - \sin \frac{p\pi\left(y_i - \frac{h_i}{2}\right)}{H} \right]. \end{aligned} \tag{13}$$

Next, calculate the partial derivatives of (13) with respect to x_i, y_i, and w_i, respectively

$$\begin{aligned} &\frac{\partial g_i(x_i, y_i, w_i)}{\partial x_i} \\ &= \sum_{u=1}^{K} \sum_{p=1}^{K} a_{u,p} \frac{H}{p\pi} \left[\cos \frac{u\pi\left(x_i + \frac{w_i}{2}\right)}{W} - \cos \frac{u\pi\left(x_i - \frac{w_i}{2}\right)}{W} \right] \\ &\left[\sin \frac{p\pi\left(y_i + \frac{h_i}{2}\right)}{H} - \sin \frac{p\pi\left(y_i - \frac{h_i}{2}\right)}{H} \right] \end{aligned}$$

$$+\sum_{u=1}^{K} a_{u,0} h_i \left[\cos \frac{u\pi\left(x_i+\frac{w_i}{2}\right)}{W} - \cos \frac{u\pi\left(x_i-\frac{w_i}{2}\right)}{W}\right] \tag{14}$$

$$\frac{\partial g_i(x_i, y_i, w_i)}{\partial y_i} = \sum_{u=1}^{K}\sum_{p=1}^{K} a_{u,p} \frac{W}{u\pi}\left[\sin \frac{u\pi\left(x_i+\frac{w_i}{2}\right)}{W} - \sin \frac{u\pi\left(x_i-\frac{w_i}{2}\right)}{W}\right] \left[\cos \frac{p\pi\left(y_i+\frac{h_i}{2}\right)}{H} - \cos \frac{p\pi\left(y_i-\frac{h_i}{2}\right)}{H}\right] + \sum_{p=1}^{K} a_{0,p} w_i \left[\cos \frac{p\pi\left(y_i+\frac{h_i}{2}\right)}{H} - \cos \frac{p\pi\left(y_i-\frac{h_i}{2}\right)}{H}\right] \tag{15}$$

$$\begin{aligned}\frac{\partial g_i(x_i, y_i, w_i)}{\partial w_i} &= \sum_{u=1}^{K}\sum_{p=1}^{K} a_{u,p} \frac{WH}{2up\pi}\Bigg\{\frac{u}{W}\left[\cos \frac{u\pi\left(x_i+\frac{w_i}{2}\right)}{W} + \cos \frac{u\pi\left(x_i-\frac{w_i}{2}\right)}{W}\right]\\ &\quad \left[\sin \frac{p\pi\left(y_i+\frac{h_i}{2}\right)}{H}) - \sin \frac{p\pi\left(y_i-\frac{h_i}{2}\right)}{H}\right]\\ &\quad - \frac{pA_i}{Hw_i^2}\left[\sin \frac{u\pi\left(x_i+\frac{w_i}{2}\right)}{W} - \sin \frac{u\pi\left(x_i-\frac{w_i}{2}\right)}{W}\right]\\ &\quad \left[\cos \frac{p\pi\left(y_i+\frac{h_i}{2}\right)}{H} + \cos \frac{p\pi\left(y_i-\frac{h_i}{2}\right)}{H}\right]\Bigg\}\\ &\quad + \sum_{u=1}^{K} a_{u,0} \frac{W}{u\pi}\Bigg\{\frac{u\pi h_i}{2W}\left[\cos \frac{u\pi\left(x_i+\frac{w_i}{2}\right)}{W} + \cos \frac{u\pi\left(x_i-\frac{w_i}{2}\right)}{W}\right]\\ &\quad - \frac{A_i}{w_i^2}\left[\sin \frac{u\pi\left(x_i+\frac{w_i}{2}\right)}{W} - \sin \frac{u\pi\left(x_i-\frac{w_i}{2}\right)}{W}\right]\Bigg\}\\ &\quad + \sum_{p=1}^{K} a_{0,p} \frac{H}{p\pi}\Bigg\{\left[\sin \frac{p\pi\left(y_i+\frac{h_i}{2}\right)}{H} - \sin \frac{p\pi\left(y_i^0-\frac{h_i}{2}\right)}{H}\right]\\ &\quad - \frac{p\pi A_i}{2Hw_i}\left[\cos \frac{p\pi\left(y_i+\frac{h_i}{2}\right)}{H} + \cos \frac{p\pi\left(y_i-\frac{h_i}{2}\right)}{H}\right]\Bigg\}.\end{aligned} \tag{16}$$

At this point, the partial derivative calculations are completed.

For each module v_i, according to (14)–(16), the time complexity required to calculate the partial derivatives with respect to the variables x_i, y_i, and w_i are $O(K^2)$, when the coefficients are given. Here, K is the truncation constant in (5), which is used to control the accuracy of the solution. Therefore, the time complexity of calculating partial derivatives for all n modules is $O(nK^2)$, which is linear with n. However, experiments show that the constant K^2 is very large. Thus, even if the run-time increases linearly with the scale of the problem, it still takes a large run-time to solve a large-scale problem. Based on this, the next section looks for a way to further reduce the run-time complexity.

C. Fast Approximate Calculation of Partial Derivatives of Potential Energy

From (6) and (7), the floorplanning area $[0, W] \times [0, H]$ has been discretized into $K \times K$ bins. The width and height of each bin are $w_b = (W/K)$ and $h_b = (H/K)$, respectively. Note that R_i is the rectangular area occupied by module v_i. Let B_{v_i} be the set of bins occupied by R_i, and denote the right, left, upper, and lower boundaries of B_{v_i} as X_U^i, X_L^i, Y_U^i, and Y_L^i, respectively. That is, $B_{v_i} = \{\text{bin}_{p,q} | X_L^i \le p \le X_U^i,\ Y_L^i \le q \le Y_U^i\}$.

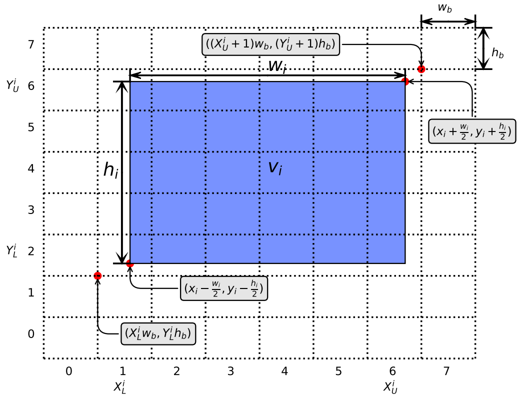

w_b
h_b
$((X_U^i+1)w_b, (Y_U^i+1)h_b)$
w_i
$(x_i+\frac{w_i}{2}, y_i+\frac{h_i}{2})$
h_i
v_i
$(x_i-\frac{w_i}{2}, y_i-\frac{h_i}{2})$
$(X_L^i w_b, Y_L^i h_b)$
7
Y_U^i 6
5
4
3
Y_L^i 2
1
0
0
1 X_L^i
2
3
4
5
6 X_U^i
7

Fig. 1. Rectangular area R_i and the squares it occupies.

First, we use auxiliary functions to determine the minimum number of bins covering the rectangular area R_i. The auxiliary functions are defined as follows:

$$\begin{aligned}\text{up}(x) &= \begin{cases} 0, & \text{if } x = 0 \\ \lceil x \rceil - 1, & \text{else } x > 0 \end{cases}\\ \text{down}(x) &= \lfloor x \rfloor, \quad x \ge 0.\end{aligned} \tag{17}$$

Then, the four boundaries of the minimum bins covering the rectangular area R_i can be calculated by

$$X_L^i = \text{down}\left(\frac{x_i - \frac{w_i}{2}}{w_b}\right), X_U^i = \text{up}\left(\frac{x_i + \frac{w_i}{2}}{w_b}\right)$$
$$Y_L^i = \text{down}\left(\frac{y_i - \frac{h_i}{2}}{h_b}\right), Y_U^i = \text{up}\left(\frac{y_i + \frac{h_i}{2}}{h_b}\right).$$

And, the bins enclosed in the rectangle with lower left corner $(X_L^i w_b, Y_L^i h_b)$ and upper right corner $((X_U^i + 1)w_b, (Y_U^i + 1)h_b)$ is exactly the set B_{v_i} of bins occupied by module v_i. Further, let $\hat{X_U^i} = X_U^i + 1$, $\hat{Y_U^i} = Y_U^i + 1$.

Fig. 1 shows the specific meaning of the above notations. Based on the rectangle rule in numerical integration, the integration in (11) can be approximated as follows:

$$\iint_{R_i} \psi(z, v)\mathrm{d}z\mathrm{d}v \approx \sum_{\text{bin}_{p.q} \in B_{v_i}} \text{Area}\left(\text{bin}_{p,q} \cap R_i\right)\hat{\psi}(p, q).$$

Expanding the right term of the above equation, we get

$$\begin{aligned} g_i(x_i, y_i, w_i) &\approx \sum_{\text{bin}_{p.q} \in B_{v_i}} \text{Area}\left(\text{bin}_{p,q} \cap R_i\right)\hat{\psi}(p, q)\\ &= \sum_{p=X_L^i}^{X_U^i}\sum_{q=Y_L^i}^{Y_U^i} w_b h_b \hat{\psi}(p, q)\\ &\quad - \sum_{q=Y_L^i}^{Y_U^i} h_b\left[\left(x_i - \frac{w_i}{2}\right) - X_L^i w_b\right]\hat{\psi}(X_L^i, q)\\ &\quad - \sum_{q=Y_L^i}^{Y_U^i} h_b\left[\hat{X_U^i} w_b - \left(x_i + \frac{w_i}{2}\right)\right]\hat{\psi}(X_U^i, q)\end{aligned}$$

$$
\begin{aligned}
&- \sum_{p=X_L^i}^{X_U^i} w_b\left[\left(y_i - \frac{h_i}{2}\right) - Y_L^i h_b\right]\hat{\psi}(p, Y_L^i)\\
&- \sum_{p=X_L^i}^{X_U^i} w_b\left[\hat{Y}_U^i h_b - \left(y_i + \frac{h_i}{2}\right)\right]\hat{\psi}(p, Y_U^i)\\
&+ \left[\left(x_i - \frac{w_i}{2}\right) - X_L^i w_b\right]\left[\hat{Y}_U^i h_b - \left(y_i + \frac{h_i}{2}\right)\right]\hat{\psi}(X_L^i, Y_U^i)\\
&+ \left[\left(x_i - \frac{w_i}{2}\right) - X_L^i w_b\right]\left[\left(y_i - \frac{h_i}{2}\right) - Y_L^i h_b\right]\hat{\psi}(X_L^i, Y_L^i)\\
&+ \left[\hat{X}_U^i w_b - \left(x_i + \frac{w_i}{2}\right)\right]\left[\hat{Y}_U^i h_b - \left(y_i + \frac{h_i}{2}\right)\right]\hat{\psi}(X_U^i, Y_U^i)\\
&+ \left[\hat{X}_U^i w_b - \left(x_i + \frac{w_i}{2}\right)\right]\left[\left(y_i - \frac{h_i}{2}\right) - Y_L^i h_b\right]\hat{\psi}(X_U^i, Y_L^i).
\end{aligned}
$$

In the above equation, w_i appears as a variable explicitly. Moreover, $h_i = (A_i/w_i)$. Hence, by taking the partial derivative of the last equality with respect to w_i, we get

$$
\begin{aligned}
\frac{\partial g_i(x_i, y_i, w_i)}{\partial w_i} &\approx \sum_{q=Y_L^i}^{Y_U^i} \frac{h_b}{2}\left[\hat{\psi}(X_L^i, q) + \hat{\psi}(X_U^i, q)\right]\\
&- \sum_{p=X_L^i}^{X_U^i} \frac{A_i w_b}{2w_i^2}\left[\hat{\psi}(p, Y_L^i) + \hat{\psi}(p, Y_U^i)\right]\\
&- \frac{1}{2}\left[\left(y_i - \frac{h_i}{2}\right) - Y_L^i h_b\right]\left[\hat{\psi}(X_L^i, Y_L^i) + \hat{\psi}(X_U^i, Y_L^i)\right]\\
&- \frac{1}{2}\left[\hat{Y}_U^i h_b - \left(y_i + \frac{h_i}{2}\right)\right]\left[\hat{\psi}(X_L^i, Y_U^i) + \hat{\psi}(X_U^i, Y_U^i)\right]\\
&+ \frac{A_i}{2w_i^2}\left[\left(x_i - \frac{w_i}{2}\right) - X_L^i w_b\right]\left[\hat{\psi}(X_L^i, Y_L^i) + \hat{\psi}(X_L^i, Y_U^i)\right]\\
&+ \frac{A_i}{2w_i^2}\left[\hat{X}_U^i w_b - \left(x_i + \frac{w_i}{2}\right)\right]\left[\hat{\psi}(X_U^i, Y_L^i) + \hat{\psi}(X_U^i, Y_U^i)\right].
\end{aligned}
\tag{18}
$$

Similarly, the partial derivatives of $g_i(x_i, y_i, w_i)$ with respect to x_i and y_i can be approximated by

$$
\begin{aligned}
\frac{\partial g_i(x_i, y_i, w_i)}{\partial x_i} &\approx \sum_{q=Y_L^i}^{Y_U^i} h_b\left[-\hat{\psi}(X_L^i, q) + \hat{\psi}(X_U^i, q)\right]\\
&+ \left[\left(y_i - \frac{h_i}{2}\right) - Y_L^i h_b\right]\left[\hat{\psi}(X_L^i, Y_L^i) - \hat{\psi}(X_U^i, Y_L^i)\right]\\
&+ \left[\hat{Y}_U^i h_b - \left(y_i + \frac{h_i}{2}\right)\right]\left[\hat{\psi}(X_L^i, Y_U^i) - \hat{\psi}(X_U^i, Y_U^i)\right]
\end{aligned}
\tag{19}
$$

$$
\begin{aligned}
\frac{\partial g_i(x_i, y_i, w_i)}{\partial y_i} &\approx \sum_{p=X_L^i}^{X_U^i} w_b\left[-\hat{\psi}(p, Y_L^i) + \hat{\psi}(p, Y_U^i)\right]\\
&+ \left[\left(x_i - \frac{w_i}{2}\right) - X_L^i w_b\right]\left[\hat{\psi}(X_L^i, Y_L^i) - \hat{\psi}(X_L^i, Y_U^i)\right]\\
&+ \left[\hat{X}_U^i w_b - \left(x_i + \frac{w_i}{2}\right)\right]\left[\hat{\psi}(X_U^i, Y_L^i) - \hat{\psi}(X_U^i, Y_U^i)\right].
\end{aligned}
\tag{20}
$$

According to (18), (19), and (20), the computing cost for $([\partial g_i(x_i, y_i, w_i)]/\partial x_i)$, $([\partial g_i(x_i, y_i, w_i)]/\partial y_i)$, and $([\partial g_i(x_i, y_i, w_i)]/\partial w_i)$ is a linear function of $(X_U^i - X_L^i)+(Y_U^i - Y_L^i)$, if all $\hat{\psi}(p, q)$ have been computed. Therefore, the time complexity of calculating the partial derivatives for all modules is $O(n(\mathrm{AVG}/n))$, where $\mathrm{AVG} = \sum_{i=1}^{n}(X_U^i - X_L^i + Y_U^i - Y_L^i)$, if all $\hat{\psi}(p, q)$ have been computed.

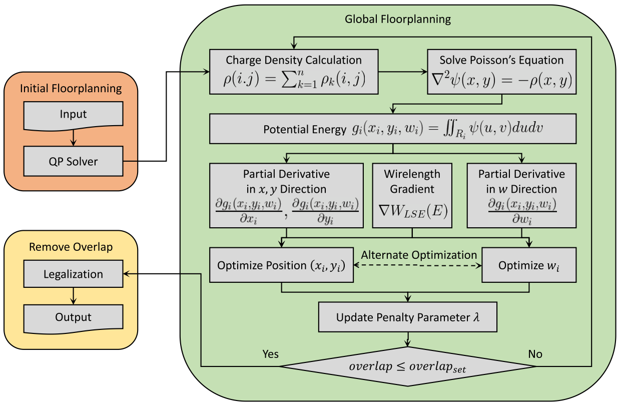

Global Floorplanning
Initial Floorplanning
Input
QP Solver
Charge Density Calculation
$\rho(i.j) = \sum_{k=1}^{n} \rho_k(i, j)$
Solve Poisson's Equation
$\nabla^2\psi(x, y) = -\rho(x, y)$
Potential Energy $g_i(x_i, y_i, w_i) = \iint_{R_i} \psi(u, v)dudv$
Partial Derivative in x, y Direction $\frac{\partial g_i(x_i,y_i,w_i)}{\partial x_i}, \frac{\partial g_i(x_i,y_i,w_i)}{\partial y_i}$
Wirelength Gradient $\nabla W_{LSE}(E)$
Partial Derivative in w Direction $\frac{\partial g_i(x_i,y_i,w_i)}{\partial w_i}$
Remove Overlap
Legalization
Output
Optimize Position (x_i, y_i)
Alternate Optimization
Optimize w_i
Update Penalty Parameter λ
Yes
No
$overlap \leq overlap_{set}$

Fig. 2. Framework of our fixed-outline floorplanning algorithm.

According to (7), we first invoke the fast Fourier transform (FFT) one time to calculate all the coefficients a_{up}, and then invoke the FFT another time to calculate all values of $\hat{\psi}(p, q)$. Since the time complexity of one FFT is $O(K^2 \log K)$, all the values $\hat{\psi}(p, q)$ can be calculated in time $O(K^2 \log K)$ by the FFT on (7). Hence, the total time complexity of calculating the partial derivatives for all modules is $O(K^2 \log K) + O(n(\mathrm{AVG}/n))$. It must be remarked that, in ePlace [24] each iteration requires invoking the FFT three times for calculating the partial derivatives for all modules [see (21) and (24) in [24]].

Further, in our fast approximate calculation of partial derivatives, it is apparently that the average number (AVG/n) of squares occupied by all modules is much smaller than the total number K^2 of squares. So the time complexity of this fast approximate calculation of partial derivatives still has a linear relationship with the total number of modules, but the difference is that the constant (AVG/n) is no longer related to the truncation constant K^2 as in (14)–(16). Therefore, the cost of calculating all partial derivatives is reduced.

For the convenience of description, we will call the method in Section III-B the partial derivative calculation method before acceleration (PDBA), and the method in this section the accelerated partial derivative calculation method.

IV. Fixed-Outline Floorplanning Algorithm

In this section, we present our algorithm for fixed-outline floorplanning, as shown in Fig. 2. It contains three phases: 1) initial floorplanning; 2) global floorplanning; and 3) legalization. We directly adopts the QP method in ePlace [24] to produce an initial floorplan. Hence, we mainly introduce in this section the global floorplanning and legalization algorithms.

A. Global Floorplanning Algorithm

In the global floorplanning phase of Fig. 2, Charge Density Calculation, Solve Poisson's Equation, Potential Energy Calculation, Partial derivative in x, y directions, Partial derivative in w direction, and Wirelength Gradient have been explained in Section III. In this section, we present other parts of the global floorplanning algorithm for problem (12), as in Algorithm 1.

Algorithm 1 Global Floorplanning Algorithm

```
Input: initial solution $u_0 = (x_0, y_0, w_0)$, $\lambda_0$, maximum iterations $k_{\max}$,
    minimum overlap $O_{\min}$;
Output: solution $u_{gp}$.
 1: for $k = 0 \rightarrow k_{\max}$ do
 2:     $f_k = LSE(E) + \lambda_k[\sum_{v_i \in V_s} g_i(x_i, y_i, w_i) + \sum_{v_i \in V_h} g_i(x_i, y_i)]$;
 3:     calculate partial derivatives $\nabla_x f_k$, $\nabla_y f_k$ and $\nabla_w f_k$ of $f_k$;
 4:     $(x_{k+1}, y_{k+1})$ = Nesterov-Solver($(x_k, y_k), f_k$);
 5:     $w_{k+1}$ =Normal-Solver($w_k, 1, f_k$);
 6:     $u_{gp} = u_{k+1} = (x_{k+1}, y_{k+1}, w_{k+1})$;
 7:     project $u_{k+1}$ onto the feasible region;
 8:     calculate overlap $O_{k+1}$ and update $\lambda_k$ to $\lambda_{k+1}$;
 9:     if $O_{k+1} \leq O_{\min}$ then
10:         break;
11: return $u_{gp}$.
```

Algorithm 2 Nesterov-Solver(u_k,f_k) // Nesterov's Method at the kth Iteration [24]

```
Input: major solution $u_k$, reference solution $v_k$, optimization
    parameter $a_k$ and objective function $f_k = f(v_k)$;
Output: $u_{k+1}$, $v_{k+1}$.
1: calculate the gradient in the $x, y$ direction: $\nabla_{x,y} f_k = \nabla_{x,y} f_k(v_k)$;
2: steplength $a_k = \arg\max_a\{f_k - f(v_k - a\nabla_{x,y} f_k) \geq 0.5a\|\nabla_{x,y} f_k\|^2\}$;
3: new solution $u_{k+1} = v_k - a_k \nabla_{x,y} f_k$;
4: parameter update $a_{k+1} = \frac{1+\sqrt{4a_k^2+1}}{2}$;
5: new reference solution $v_{k+1} = u_{k+1} + \frac{(a_k-1)(u_{k+1}-u_k)}{a_{k+1}}$;
6: return $u_{k+1}$, $v_{k+1}$.
```

In Algorithm 1, we use projected gradient method to solve problem (12). Line 2 constructs the objective function f_k required in the kth iteration. Line 3 calculates the partial derivative of the objective function in each direction, which will be used in lines 4 and 5.

We use Nesterov's method [24] to optimize the module positions (x_i, y_i). In line 4 of Algorithm 1, we first fix the widths of soft modules and call Nesterov's method to optimize module positions (Algorithm 2), which was first adopted for VLSI placement in ePlace [24]. Nesterov's method is the first-order optimization algorithm with convergence rate $O(1/k^2)$, where k is the number of iterations. We set $a_0 = 1$, $u_0 = v_0$, and v_0 is the solution given by the initial floorplanning.

In Nesterov's method, since we have fixed the widths of soft modules when we call the Nesterov-Solver, we use the same diagonal Jacobi preconditioner as ePlace [24] to get faster convergence and a better solution.

Line 5 of Algorithm 1 uses normalized gradient descent method to optimize the widths of soft modules. By fixing the positions of modules obtained in line 4, we calculate $\nabla_w f_k = \nabla_w f_k(u_k)$. Then, we set $u_{k+1} = u_k - \text{step}\nabla_w f_k$, in which $\text{step} = a/\|\nabla_w f_k\|$, and, here, we set the constant $a = 1$. Note that, after optimizing the widths w_i of soft modules, the positions of pins will be affected. Hence, in order to ensure the optimization accuracy, the positions of pins will be updated proportionally before the next time we call the Nesterov-Solver.

Since u_{k+1} may not satisfy the constraints in problem (12). In order to obtain a feasible solution, line 7 of Algorithm 1 projects u_{k+1} onto the feasible region of problem (12), such that it satisfies the constraints (12b)–(12d). First, to meet the constraint (12b), line 7 projects the widths w_i of soft modules in u_{k+1} as follows:

$$w_i = \begin{cases} \sqrt{\frac{A_i}{AR_L^i}}, & \text{if } \frac{A_i}{w_i^2} < AR_L^i \\ \sqrt{\frac{A_i}{AR_U^i}}, & \text{if } \frac{A_i}{w_i^2} > AR_U^i \\ w_i, & \text{else.} \end{cases}$$

Next, line 7 of Algorithm 1 projects in the same way the positions of modules such that they satisfy the constraints (12c) and (12d)

$$x_i = \begin{cases} \frac{w_i}{2}, & \text{if } x_i < \frac{w_i}{2} \\ W - \frac{w_i}{2}, & \text{if } x_i > W - \frac{w_i}{2} \\ x_i, & \text{else} \end{cases}$$

$$y_i = \begin{cases} \frac{h_i}{2}, & \text{if } y_i < \frac{h_i}{2} \\ H - \frac{h_i}{2}, & \text{if } y_i > H - \frac{h_i}{2} \\ y_i, & \text{else.} \end{cases}$$

Line 8 of Algorithm 1 updates the value of penalty parameter λ and calculates the overlaps between modules. Here, the method for updating the penalty parameter value is the same as that in ePlace [24]. First, set an expected increase of wirelength $\Delta\text{HPWL}_{\text{ref}}$ for each iteration. Then, the penalty parameter is updated as follows:

$$\lambda_{k+1} = \mu_k \lambda_k, \quad \mu_k = \mu_0^{-\frac{\Delta\text{HPWL}_k}{\Delta\text{HPWL}_{\text{ref}}}+1}$$

where $\Delta\text{HPWL}_k = \text{HPWL}(v_k) - \text{HPWL}(v_{k-1})$. The constant μ_0 is set to 1.1, and μ_k is usually truncated to the interval [0.75, 1.1].

Finally, in lines 9 and 10, Algorithm 1 will stop if the current total overlap between modules is less than the threshold $O_{\min}$, which ensures that the overlapping area must be less than $O_{\min}$ before legalization begins.

B. Legalization

After global floorplanning, we can get a rough floorplanning of modules with a small overlap. However, there is still a little overlap in the floorplan (more details can be found in the experimental results section). In other words, it does not satisfy the constraint that there is no overlap between modules. Hence, legalization will be used to make the global floorplan be a legal floorplan. Therefore, this section will discuss how to eliminate the overlap between modules.

In legalization, we use the global floorplanning result as the initial solution, and treat all soft modules as hard ones. If a floorplan is small scale, then it is legalized using the *pl2sp* function in Parquet [11] directly; otherwise it is legalized using the method proposed in this section.

Abacus [27], SAINT [23], and Floorist [28] are state-of-the-art legalization methods to eliminate overlaps between modules. Abacus [27] is used in VLSI placement, in which standard cells can be aligned to rows. However, in fixed-outline floorplanning, the modules are of various heights. Hence, it is difficult to extend Abacus to legalize a floorplan. SAINT [23] uses polygon to approximate the curve of the area of every soft module, and solve an ILP for legalization, which is time consuming and is difficult to scale up for large-scale floorplanning problem. Floorist [28] is a floorplan repair method,

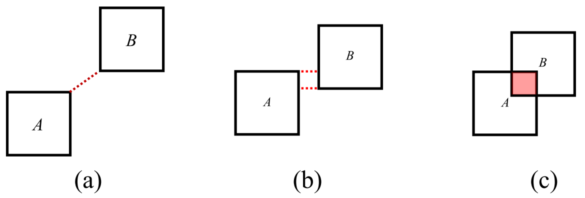


Fig. 3. Three overlapping relationships between modules which are used to construct VCG and HCG.

its main idea is as follows. First, a nonoverlapping floorplan is constructed based on the global floorplanning result. If it exceeds the floorplanning area, then the positional relationship of modules is modified and the floorplan is compressed, until all modules are located legally in the floorplanning area.

We propose a new legalization algorithm based on Floorist [28]. The biggest difference between our legalization algorithm and Floorist is as follows. Floorist considers how to insert constraints into the constraint graphs so that all modules finally are nonoverlapping, but does not delete constraints in the constraint graphs. During insertion, the fixed outline is satisfied but the nonoverlapping condition may be violated. A major issue is that the constraint graphs may become very dense finally. In contrast, our algorithm not only inserts constraints but also deletes constraints in the constraint graphs. Moreover, our algorithm ensures that the nonoverlapping condition is satisfied but the fixed outline may be violated. Eventually, our algorithm will satisfy the fixed-outline condition by modifying the constraint graphs.

1) Horizontal and Vertical Constraint Graphs and Related Definitions: First, we construct horizontal and vertical constraint graphs HCG and VCG, according to the positional relationship of the modules after global floorplanning. Assume that the coordinate of the lower left corner of module A is smaller than that of B in both the horizontal and vertical directions, as Fig. 3 shows. Then, we have the following processing for modules A and B.

1) If A and B do not overlap both horizontally and vertically, then we have $A \uparrow B \in VCG$ and $A \rightarrow B \in HCG$.
2) If A and B overlap vertically but do not overlap horizontally, then we have $A \rightarrow B \in HCG$.
3) Assume that A and B overlap both horizontally and vertically. If the horizontal overlap is greater than the vertical overlap. Then there is $A \uparrow B \in VCG$, otherwise $A \rightarrow B \in HCG$.

Basically, our algorithm legalizes a floorplan alternatively in the horizontal and vertical directions. In the horizontal direction, our legalization algorithm modifies a floorplan by deleting an edge between two modules in HCG, then adding an edge between the two modules in VCG, and compress the floorplan. The vertical legalization operates in a similar way. Generally, we can arbitrarily delete or add an edge between two modules, if there is no contradiction. Moreover, it is not needed to ensure that HCG and VCG are transitive, for effectively reducing the number of edges in HCG and VCG.

However, choosing an edge for adjustment is an important step that affects the effectiveness of our legalization algorithm. During horizontal legalization, we tend to select an edge that has the least impact on VCG, which is achieved by comparing the weights of the edges in HCG defined in (24). The vertical legalization selects an edge in VCG similarly. The specific details will be given in Algorithm 4.

During horizontal legalization, according to the HCG, we first move the modules as far as possible to the left in the floorplanning area, while keeping the HCG unchanged. The method is, we calculate the abscissa x_A of the lower left corner of every module A

$$x_A = \begin{cases} \max_{\forall B(B \rightarrow A)}(x_B + w_B), & \text{if } \exists B(B \rightarrow A) \\ 0, & \text{else.} \end{cases} \tag{21}$$

Similar to (21), the ordinate y_A of the lower left corner of every module A can be calculated. After the above calculations, the coordinate (x_A, y_A) of the lower left corner of each module is determined, and all overlaps between modules are eliminated.

Next, let noRight $= \{A | \forall B, A \not\rightarrow B\}$, we calculate the width W_d of the smallest rectangle enclosing all modules

$$W_d = \max_{A \in \text{noRight}} (x_A + w_A). \tag{22}$$

Similar to (22), we can calculate the height H_d of the smallest rectangle enclosing all modules.

Then, we change temporarily the origin of the coordinate system to $(W_d, 0)$, and change the positive direction of the x-axis to left. In the temporary coordinate system, we construct the horizontal constraint graph HCG' of the floorplan. According to the HCG', by calculating (21), the lower left corner coordinate x'_A of each module A can be obtained. After the above calculations, the horizontal slack S_A^x of a module A is defined as follows:

$$S_A^x = W - \left(x_A + x'_A + w_A\right) \tag{23}$$

here W is the width of the floorplanning area.

Similarly, we can define the vertical slack S_A^y of module A. Then, we say a horizontal relationship $A \rightarrow B$ is critical if $S_A^x = 0$ and $S_B^x = 0$. Similarly, we say a vertical relationship $A \uparrow B$ is critical if $S_A^y = 0$ and $S_B^y = 0$.

Next, we define the compressible relationship between modules. Modules A and B has a compressible relationship in the horizontal direction, denoted as $A \cap_y B = \emptyset$, if $A \rightarrow B$ and $[y_A, y_A + h_A] \cap [y_B, y_B + h_B] = \emptyset$. This definition means that, if there are two modules A and B such that $A \cap_y B = \emptyset$, then A or B may be moved horizontally such that the floorplan is more compact. Similarly, we can define the vertical compressible relationship, and denote it as $A \cap_x B = \emptyset$. Moreover, for a horizontal critical relationship $A \rightarrow B$, we define the weight of $A \rightarrow B$ as follows:

$$\text{Weight}(A \rightarrow B) = \begin{cases} S_A^y - h_B, & \text{if } y_A \leq y_B \\ S_B^y - h_A, & \text{else.} \end{cases} \tag{24}$$

For $y_A \leq y_B$, if we delete the horizontal critical relationship $A \rightarrow B$ in HCG and add the vertical relationship $A \uparrow B$ in VCG, then the vertical slack of the module A is $S_A^y - h_B$. Similarly, for $y_A > y_B$, after deleting $A \rightarrow B$ in HCG and adding $B \uparrow A$ in VCG, the vertical slack of the module B is about $S_B^y - h_A$. By setting this weight, when we need to delete an edge in the HCG and add a new edge in the VCG, the algorithm will preferentially select the edge with the smallest increase of the height H_d of the smallest rectangle enclosing all modules. Similarly, we can define the weight of a vertical critical relationship in the vertical constraint graph.

Algorithm 3 Legalization Algorithm

```
Input: width and height of floorplanning area $(W, H)$, horizontal and vertical
    constraint graphs $(HCG, VCG)$, maximum number of iterations $N$;
Output: coordinate $(\hat{x}_A, \hat{y}_A)$ of lower left corner of each module.
 1: for $i = 1 \to N$ do
 2:     $(HCG, VCG) = LG_x(W, HCG, VCG)$;
 3:     $(HCG, VCG) = LG_y(H, HCG, VCG)$;
 4:     calculate the smallest rectangle $(W_d, H_d)$ enclosing all modules
        according to $(HCG, VCG)$;
 5:     if $W_d \le W$ and $H_d \le H$ then
 6:         break;
 7: return $(\hat{x}_A, \hat{y}_A)$.
```

Algorithm 4 $LG_x(W, HCG, VCG)$ // Horizontal Legalization

```
Input: width $W$ of floorplanning area, horizontal and vertical constraint
    graphs $(HCG, VCG)$;
Output: $(HCG, VCG)$ after horizontal legalization.
 1: calculate $(x_A, y_A)$ and $(S_A^x, S_A^y)$ of all modules and the width $W_d$ based
    on the current $(HCG, VCG)$;
 2: while $W_d > W$ do
 3:     let $S = \{A \to B \in HCG | S_A^x = 0 \ and \ S_B^x = 0\}$ be the set of horizontal
        critical relationships;
 4:     if $\exists (A \to B) \in S$ and $A \cap_y B = \emptyset$ then
 5:         delete relationship $A \to B$ from the current $HCG$;
 6:     else
 7:         for $\forall (A \to B) \in S$ do
 8:             calculate $Weight(A \to B)$;
 9:         find modules $A$ and $B$ with largest $Weight(A \to B)$;
10:         delete relationship $A \to B$ from the current $HCG$;
11:         if $y_A \le y_B$ then
12:             add relationship $A \uparrow B$ to the current $VCG$;
13:         else
14:             add relationship $B \uparrow A$ to the current $VCG$;
15:     do line 1 again;
16: return $(HCG, VCG)$.
```

2) Legalization Algorithm: Algorithm 3 shows the overall process of our legalization algorithm, which finds a legalized floorplan within a given number of iterations.

Lines 2 and 3 of Algorithm 3 legalize the floorplan in the horizontal and vertical directions alternatively, which will be introduced in Algorithm 4. According to the current HCG and VCG, line 4 uses (21) to calculate the coordinate $(\hat{x}_A, \hat{y}_A)$ of the lower left corner of each module, and uses (22) to calculate the smallest rectangle enclosing these modules. Line 5 of Algorithm 3 determines whether the smallest rectangle is within the floorplanning area. If so, exit the iteration.

Algorithm 4 presents our legalization algorithm in the horizontal direction. The vertical legalization algorithm is similar, and will not be given here.

According to the current HCG and VCG, lines 1 and 15 use (21) to move all modules to the left and bottom in the floorplanning area, and make them nonoverlapping. Then, lines 1 and 15 use (23) and (22) to calculate horizontal and vertical slacks (S_A^x, S_A^y) of all modules, and the width W_d of the smallest rectangle enclosing these modules, respectively.

Line 2 determines whether the floorplan obtained from lines 1 or 15 is already in the floorplanning area. If so, a feasible solution in the horizontal direction has been obtained, and the algorithm stops. Line 3 of Algorithm 4 finds all critical relationships in the HCG. Lines 4 and 5 determine whether there is a compressible relationship $A \to B$ in the set of horizontal critical relationships. If so, it indicates that module B can be moved left so that A and B still do not overlap. Therefore, we delete relationship $A \to B$ from the current HCG.

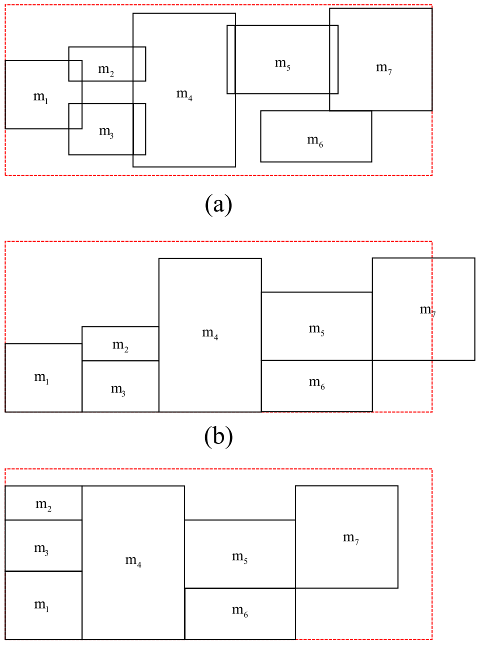


Fig. 4. Distribution of modules: (a) modules with overlaps [29]; (b) floorplan after compression; and (c) legal floorplan after horizontal legalization by Algorithm 4.

Lines 7–14 deal with the situation that the floorplan cannot be compressed directly in the horizontal direction. In this situation, we use (24) to calculate Weight$(A \to B)$ of all horizontal critical relationships, and find the largest one between lines 7 and 9. Then we delete the $A \to B$ corresponding to the largest Weight$(A \to B)$ in HCG at line 10, and finally add a new corresponding VCG edge between lines 11–14. The purpose of this is to minimize the increase in the height H_d of the floorplan after changing the relationship.

Let HCG be $G(V, E)$. According to the topological-sort and (21)–(23), the time complexity of lines 1 and 15 in Algorithm 4 is $O(|E| + |V|)$. Note that, $|S| \le |E|$ holds for the set of horizontal critical relationships. So, the time complexity of lines 4, 5, and 7–9 of Algorithm 4 are all $O(|E|)$. Therefore, the time complexity of each iteration of Algorithm 4 is $O(|E| + |V|)$.

3) Example of Horizontal Legalization: To illustrate the effectiveness of our legalization algorithm, we implement it on the example of XDP [29] as shown in Fig. 4(a), which was used in [29] for mixed-size placement to show that it is nontrivial to identify a right set of edges for adjustment.

Fig. 5(a) presents the horizontal constraint graph corresponding to the floorplan in Fig. 4(a). First, according to the current HCG and VCG, line 1 of Algorithm 4 will obtain a floorplan without overlapping that is compressed in the horizontal and vertical directions, which is shown in Fig. 4(b).

Then because $W_d > W$, we have to legalize the modules in the horizontal direction, such that they are all in the horizontal outline. First, we need to extract critical relationships from HCG through line 3 of Algorithm 4, and line 4 will delete all compressible relationships until $W_d \le W$. In this

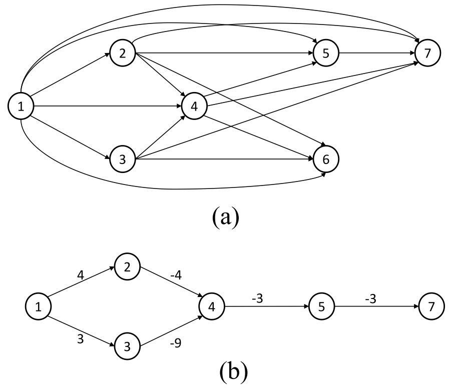


Fig. 5. (a) HCG of floorplan in Fig. 4(a), in which vertex i represents module m_i, $i = 1, 2, \ldots, 7$. (b) Horizontal critical relationships of HCG and the weights of the edges.

example, since there are no compressible relationships, we use (24) to calculate the edge weights. Fig. 5(b) is the corresponding critical relationships of HCG and the weights of the edges.

Now we need to select an edge from HCG with the largest weight to delete, and add a new vertical edge in VCG. We sequentially delete $m_1 \rightarrow m_2$ and then add $m_1 \uparrow m_2$, delete $m_1 \rightarrow m_3$ and then add $m_1 \uparrow m_3$. It is worth to note that when we delete a relationship and create a new relationship, the weights of related edges will be updated. Finally, after executing line 15 of Algorithm 4, we get a horizontally legal floorplan, which is shown in Fig. 4(c).

V. Experimental Results

In this section, we compare our fixed-outline floorplanning algorithm with state-of-the-arts experimentally on MCNC [30], GSRC [31], HB+ [32], and ami49_x [12], [15] benchmarks. Our algorithm was implemented in C++ and run on a Linux machine with 3.60-GHz Intel Core i3-9100 CPU and 4-GB memory. We focus on floorplanning with hard and soft modules. In comparisons, wirelength is calculated using the HPWL, and all compared data are cited from relevant literatures directly.

In experiments of our floorplanning algorithm, parameters were set as follows: $k_{\max} = 1200$ and $O_{\min}$ is set to about 1% in Algorithm 1, and $N = 10$ in Algorithm 3.

A. MCNC and GSRC Benchmarks

These benchmarks are ami33, ami49, n100, n200, and n300. The numbers of modules are 33, 49, 100, 200, and 300, respectively, and all modules are soft. We conducted two experiments. The first experiment fixed the whitespace at 15%, and changed the aspect ratio of floorplanning area from 1:1 to 4:1. The purpose is to test the adaptability of the algorithms to aspect ratio. The second experiment reduced the whitespace to 10%, and fixed the aspect ratio as 1:1. In addition, for the two sets of experiments, we set $(1/3) = AR_i^l \leq (h_i/w_i) \leq AR_i^u = 3$ for all soft modules.

TABLE III
HPWL Results of Three Algorithms on the MCNC and GSRC Benchmarks, Whitespace 15%

Name	Algorithm	Aspect ratio γ			
		1:1	2:1	3:1	4:1
ami33	Parquet-4	82149	79131	91721	101274
	Analytical	74072	75168	75180	79529
	ours	64134	65094	67457	69615
ami49	Parquet-4	928597	942117	1092771	1003220
	Analytical	799239	829888	880387	939049
	ours	668608	674155	727192	761057
n100	Parquet-4	342103	351542	351338	392118
	Analytical	291628	290158	298894	313060
	ours	281655	281294	288535	289199
n200	Parquet-4	630014	645219	639803	685057
	Analytical	572145	565927	583282	608074
	ours	509680	510658	523230	539112
n300	Parquet-4	770354	780406	838600	872501
	Analytical	702822	722527	793771	858346
	ours	576158	575100	602310	656570
Ratio	Parquet-4	1.31	1.33	1.36	1.32
	Analytical	1.16	1.18	1.19	1.21
	ours	1.00	1.00	1.00	1.00

TABLE IV
HPWLs of GSRC Benchmarks, Aspect Ratio 1:1, and Whitespace 10%

Name	Parquet-4	Capo 10.2	AR	IMF	IMF +AFF	Ref. [2]	DeFer	ours
n100	242050	224390	203700	207852	208772	198649	208650	194273
n200	432882	385594	367880	369888	372845	351193	372546	349301
n300	647452	522968	492830	489868	494480	483757	498909	468235
Ratio	1.29	1.13	1.05	1.06	1.07	1.02	1.06	1.00

In the first experiment, we compare the test results of our floorplanning algorithm using the accelerated partial derivative calculation method ("ours," the same meaning in the sequel) with analytical [18], which is an analytical approach, and with Parquet-4 [11] which is based on simulated annealing. In this experiment, none of the three floorplanners uses a multilevel framework. For all test benchmarks, the I/O pads are fixed at the location given originally. The detailed test results are put in Table III.

In the table, the first and second columns present the names of the test benchmarks and the floorplanning algorithms, respectively. Columns 3–6 give the test results of respective floorplanners. The last three rows of the table show the average results of the two floorplanners proportional to that of our floorplanning algorithm, respectively. From Table III, it can be seen that for different aspect ratio, the average HPWL of our algorithm is at least 16% less than that of analytical [18], and at least 31% less than that of Parquet-4 [11].

In the second experiment, for each benchmark, the aspect ratio was set to $\gamma = 1$, and the I/O pads were shifted to the boundary of the floorplanning area. Compared floorplanning algorithms are: analytical-based methods AR [19] and Lin et al. [2] without using multilevel framework, Capo 10.2 [13], IMF [15], and IMF + AMF [12] based on a multilevel framework, DeFer [14], and Parquet-4 [11] based on simulated annealing.

Table IV lists the HPWL results obtained by respective algorithms. The data of DeFer are cited from [14], and the data of the other compared algorithms are cited from [2] and [19] directly. On average, our HPWL result is 6%, 6%, 7%, 13%, and 29% less than that of DeFer [14], IMF [15], IMF + AMF [12], Capo 10.2 [13], and Parquet-4 [11], respectively. Compared to the analytical algorithms Lin et al. [2] and AR [19], our HPWL result is 2% and 5% less.

TABLE V
RESULTS ON THE HB+ BENCHMARKS

Basic Information					HPWL (10^6)					time (s)				
Name	Hard	Soft	I/O	white-space	QinFer	Ref. [2]	DeFer	ours PDBA	ours	QinFer	Ref. [2]	DeFer	ours PDBA	ours
HB+01	665	246	246	26%	2.91	3.18	3.09	3.03	3.00	5.0	150	1.8	19.49	4.3
HB+02	1200	271	259	25%	6.47	6.80	6.17	6.53	6.02	11.3	338	15.3	100.05	4.3
HB+03	999	290	283	30%	9.17	9.68	9.19	8.27	8.17	8.6	262	4.0	88.77	8.1
HB+04	1289	295	287	25%	11.16	9.87	10.26	10.03	9.72	12.4	209	14.2	137.16	7.5
HB+06	571	178	166	25%	8.03	8.50	8.78	7.95	7.91	7.8	125	5.0	24.36	7.1
HB+07	829	291	287	25%	14.88	15.10	15.48	14.73	14.07	8.5	230	4.6	74.21	7.2
HB+08	968	301	286	26%	16.49	17.60	18.73	18.35	17.19	22.8	324	19.3	73.76	9.7
HB+09	860	253	285	25%	17.24	18.30	16.66	16.32	15.81	8.7	259	4.2	58.82	9.0
HB+10	809	786	744	20%	42.33	46.70	45.12	41.58	40.61	11.4	191	6.3	151.73	32.9
HB+11	1124	373	406	25%	25.7	28.20	26.99	25.38	24.58	10.1	494	7.1	77.92	9.3
HB+12	582	651	637	26%	52.83	53.60	50.17	49.26	48.96	16.6	136	5.5	54.36	15.0
HB+13	830	424	490	25%	35.44	35.40	35.51	32.51	32.65	23.7	90	5.9	41.56	14.2
HB+14	1021	614	517	25%	60.68	63.40	64.50	61.57	59.62	14.9	208	12.0	76.22	16.7
HB+15	1019	393	383	25%	77.48	79.10	84.29	74.97	73.93	37.1	532	14.7	89.37	24.8
HB+16	633	458	504	25%	98.53	92.90	98.66	87.66	87.32	11.2	160	8.1	47.49	20.0
HB+17	682	760	743	25%	140.84	140.00	144.56	136.37	138.44	15.2	169	14.7	51.71	20.5
HB+18	658	285	272	25%	70.36	70.70	71.86	68.76	68.05	12.4	66	11.3	32.84	14.5
Ratio					1.05	1.08	1.08	1.02	1.00	1.23	24.43	0.89	7.06	1.00

B. HB+ Benchmarks

HB+ benchmarks were generated from HB benchmarks, with the largest hard macros being enlarged by 100% and the area of remaining soft macros reduced to preserve the total cell area. Therefore, HB+ benchmarks are very hard to handle. In this experiment, we compare the results by our algorithm to multilevel-based floorplanners DeFer [14], QinFer [16], and Lin et al. [2].

DeFer [14] has the characteristics of fast speed and high solution quality. It can solve large-scale fixed-outline slicing floorplanning problems. QinFer [16] is a newly published partitioning-based fixed-outline floorplanning algorithm with high solution quality. Ref. [2] is an analytical floorplanner based on multilevel framework. Note that we have two versions of calculating partial derivatives of potential energy. One is the PDBA presented in Section III-B, and another one is the accelerated partial derivative calculation method presented in Section III-C. Hence, we have two versions of the floorplanning algorithm, respectively. "Ours PDBA" means our floorplanning algorithm using the PDBA, and ours indicates our floorplanning algorithm using the accelerated partial derivative calculation method.

Table V gives the basic information of the HB+ benchmarks and the test results of compared floorplanners. The first 5 columns of Table V present the basic information of the benchmarks, including the names of benchmarks, the numbers of hard modules and soft modules, the number of terminals (I/O pads), and the whitespace fractions. The other information of each benchmark, such as the width and height of the floorplanning area, the upper and lower bounds of aspect ratios of soft modules, can be found in the corresponding literature. In the table, columns 6–10 are the HPWL results obtained by the respective algorithms. According to the last row of Table V, the average HPWL of our algorithm is 5%, 8%, and 8% less than that of QinFer, DeFer, and Lin et al. [2], respectively.

In Table V, columns 11–15 present the run-times of different algorithms, in which the run-times of DeFer, Lin et al. [2], and QinFer are directly cited from [2], [14], and [16], respectively. The computing environment of DeFer was Core Duo 1.86-GHz CPU with 2-GB memory, QinFer was Intel core i5-7500 3.40 GHz with 8-GB memory, and [2] was Intel Xeon 2.00-GHz CPU. On average, the run times of DeFer and QinFer are 89% and 123% of our algorithm, respectively. However, the run-time of Lin et al. [2] is quite long. Although the computing environments are different, the speed of our floorplanning algorithm can still be compared roughly.

Note that both of our floorplanning algorithm and Lin et al. [2] are modified from VLSI placement algorithms, respectively, and both of them do not use a multilevel framework for the small-scale benchmarks in Table IV. According to Table IV, the average HPWL of our algorithm is 2% less than that of Lin et al. [2]. However, for large-scale benchmarks in Table V, our floorplanning algorithm still does not use a multilevel framework, but the algorithm in [2] does. According to Table V, the HPWL of our algorithm is 8% less than that of [2]. This further improvement is obviously due to that our floorplanning algorithm is flat, i.e., does not use the multilevel framework.

Theoretically, the results of "our PDBA" should be better than ours, since it calculates exactly the partial derivatives of the potential energy. However, according to Table V, the average HPWL result of our PDBA has a slight increase of 2% over ours. This seems to be inconsistent with the theory. In fact, the fixed-outline floorplanning is a nonconvex optimization problem, and its objective function has multiple local minima. Therefore, a better solution may not be obtained by searching in the exact gradient descent direction. While, each iteration in ours has a certain error of partial derivative calculation, which makes our algorithm have limited ability to jump out of local optima, and find a better solution.

C. ami49_x Benchmarks

In this experiment, we test our floorplanning algorithm on ami49_x benchmarks [12], [15] to see its performance on large-scale fixed-outline floorplanning. The benchmarks ami49_x were designed as follows. First, duplicate each module v_i and net x times, and get new modules $v_{i,1}, v_{i,2}, \ldots, v_{i,x}$. Then, add $x - 1$ nets $(v_{i,1}, v_{i,2}), (v_{i,1}, v_{i,3}), \ldots, (v_{i,1}, v_{i,x})$. Finally, reduce the width and height of each module by 5

TABLE VI
RESULTS ON THE AMI49_X BENCHMARKS AND WHITESPACE 15%

Basic Information			wirelength (10^6)				time (min)			
Name	Module	Net	MB*-tree	IMF +AFF	Capo 10.2	ours	MB*-tree	IMF +AFF	Capo 10.2	ours
ami49_10	490	4521	3.79	3.31	3.20	2.68	2.1	0.2	0.9	0.08
ami49_20	980	9091	10.20	8.13	7.66	6.80	4.8	0.5	1.9	0.09
ami49_40	1960	18231	26.70	20.20	19.30	19.03	12.7	1.3	3.7	0.22
ami49_60	2940	27371	49.90	34.20	33.70	32.30	25.7	2.2	6.4	0.43
ami49_80	3920	36511	74.80	50.50	49.50	47.30	36.7	3.5	7.8	1.08
ami49_100	4900	45651	103.00	68.60	67.90	65.32	66.2	5.1	10.4	1.57
ami49_150	7350	68501	184.00	121.00	119.00	113.86	107.5	11.0	18.4	2.57
ami49_200	9800	91351	334.00	180.00	177.00	170.02	224.8	19.1	22.4	5.02
Ratio			1.58	1.10	1.07	1.00	43.43	3.88	6.50	1.00

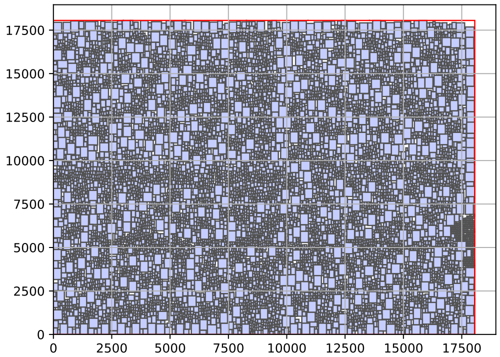


Fig. 6. Result of ami49_200 by our floorplanning algorithm (9800 modules, 91351 nets, and $HPWL = 170.02 \times 10^6$).

times to avoid overflow in computing the wirelength. In this experiment, the whitespace is 15%, the aspect ratio of floorplanning area is $\gamma = 1$, and the I/O pads are shifted to the boundary of the floorplanning area.

Table VI presents the test results of our algorithm, Capo 10.2 [13], MB*-tree [15], and IMF+AFF [12]. The first three columns of Table VI give the basic information. Columns 4–7 and 8–11 are the HPWL and run-time results of each algorithm. The computing environments of Capo 10.2, MB*-tree, and IMF+AFF were Intel Pentium-4 3.20-GHz CPU with 3-GB memory. The test results of Capo 10.2, MB*-tree, and IMF+AFF are cited from [12] directly. Note that, our algorithm does not employ a multilevel framework used in the compared floorplanners.

From the last row of Table VI, the average HPWL of our floorplanning algorithm is 7%, 10%, and 58% less than that of Capo 10.2, IMF+AFF, and MB*-tree, respectively. In terms of run-time, we cannot make direct comparisons due to different computing environment. Despite that, our floorplanning algorithm can succeed in solving every instance with at most 9800 modules in about no more than 5 min. While, the compared floorplanners require at least 19 min of computing. Therefore, the run-time of our algorithm is acceptable.

Fig. 6 shows the results of our algorithm on the largest test case ami49_200. It can be seen that the quality of our algorithm is reliable.

D. Run-Time Analysis

Next, we make the running time component analysis of our floorplanning algorithm with an accelerated partial derivative calculation method. Note that, the total running time of our algorithm = initial solution generation time by QP + run-time of our global floorplanning + run-time of our legalization.

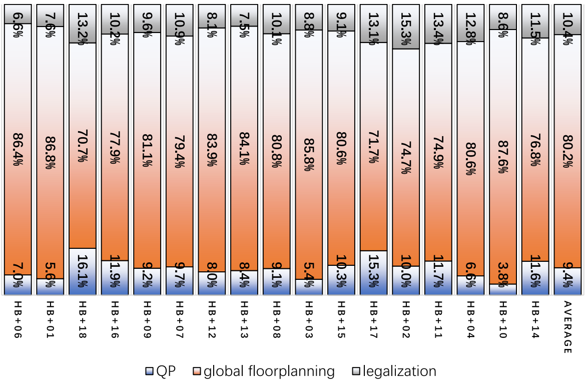


Fig. 7. Proportions of run-time spent by QP, global floorplanning, and legalization on the HB+ benchmarks.

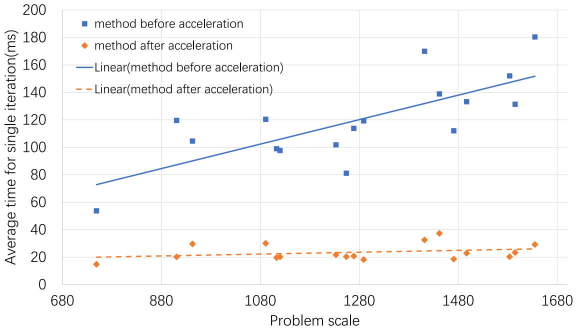


Fig. 8. Relationship between the average run-time of a single iteration of our global floorplanning algorithm with exact and with approximate partial derivative calculation methods.

Fig. 7 presents the run-time proportion of our algorithm on all HB+ benchmarks. In the figure, the abscissa represents the benchmark name, sorted from the left to right in an increasing scale. For each benchmark, from the bottom to top, the figure shows the run-time proportions of QP, global floorplanning, and legalization, respectively. On average, our floorplanning algorithm uses 9.4% of run-time on generating the initial solution, uses 80.2% of run-time on global floorplanning, and uses 10.4% of run-time on legalization. Therefore, our global floorplanning consumes most of run-time among the three components.

Since the run-time of our global floorplanning is approximately equal to the number of global floorplanning iterations × the average run-time of a single iteration of the global floorplanning. Fig. 8 gives the trend chart of the average run-time of a single iteration of our global floorplanning on HB+ benchmarks with respect to the benchmark scale.

In the figure, the solid line and the dashed line are the liner fittings of the average run-times of a single iteration of our global floorplanning algorithm using the exact partial derivative calculation method, and the approximate partial derivative calculation method, respectively. It can be seen that both of them have linear time complexity, but the latter one is much faster.

E. Results of Global Floorplanning and Legalization on HB+ and ami49_x Benchmarks

Next, we analyze the overlap rate after global floorplanning and the wire-length increase after the legalization of our floorplanning algorithm. Table VII gives the overlap rates

TABLE VII
RESULTS OF OVERLAP RATE AFTER GLOBAL FLOORPLANNING AND HPWL INCREASE AFTER LEGALIZATION

Name	Overlap rate	HPWL(10^6)		Rate of change	Name	Overlap rate	HPWL(10^6)		Rate of change
		Before	After				Before	After	
HB+01	0.97%	2.99	3.00	0.48%	HB+15	1.00%	75.18	73.93	-1.66%
HB+02	1.00%	6.05	6.02	-0.42%	HB+16	1.07%	88.54	87.32	-1.38%
HB+03	0.94%	8.23	8.17	-0.63%	HB+17	1.02%	141.38	138.44	-2.07%
HB+04	0.93%	9.80	9.72	-0.86%	HB+18	1.06%	69.47	68.05	-2.04%
HB+06	0.92%	8.02	7.91	-1.33%	ami49_10	0.95%	2.69	2.68	-0.36%
HB+07	0.99%	14.09	14.07	-0.12%	ami49_20	0.98%	6.71	6.80	1.20%
HB+08	0.97%	17.44	17.16	-1.64%	ami49_40	0.99%	18.88	19.03	0.77%
HB+09	0.93%	16.06	15.81	-1.50%	ami49_60	1.00%	31.98	32.30	1.00%
HB+10	1.08%	40.57	40.61	0.10%	ami49_80	1.00%	46.84	47.30	0.99%
HB+11	1.07%	24.58	24.58	-0.03%	ami49_100	0.70%	65.03	65.32	0.44%
HB+12	0.93%	48.30	48.96	1.37%	ami49_150	0.70%	113.58	113.86	0.25%
HB+13	0.95%	32.93	32.65	-0.85%	ami49_200	0.89%	168.79	170.02	0.73%
HB+14	1.09%	60.21	59.62	-0.99%	Average	0.96%			-0.34%

(columns 2 and 7) and HPWLs (columns 3 and 8) before legalization, the HPWLs after legalization (columns 4 and 9), and the wirelength increase of HPWL (columns 5 and 10) after our legalization algorithm for each benchmark, respectively. Here, the overlap rate is calculated by

$$\text{Overlap Rate} = \sum_{i=0}^{K-1}\sum_{j=0}^{K-1} \max\{\text{OR}(i,j),\ 0\} \times 100\%$$

in which

$$\text{OR}(i,j) = \frac{\sum_{v_k\in V} \text{Area}\big(\text{bin}_{i,j} \cap R_k\big)}{\text{Area}\big(\text{bin}_{i,j}\big)} - 1.$$

The wirelength increase of HPWL is calculated using

$$\frac{\text{HPWL After Legal.} - \text{HPWL Before Legal.}}{\text{HPWL Before Legal.}} \times 100\%$$

for which a positive number means that the legalization has increased wirelength, and a negative number means that wirelength has been decreased.

From Table VII, it can be seen that our global floorplanning optimizes the wirelength while effectively reduces the overlap. On average, the average overlap rate is 0.96% after global floorplanning. Further, our legalization reduces on average the wirelength by 0.34%. The reduction in wirelength in fact is due to compressing the floorplan. Note that, there is no significant change in wirelength before and after legalization. This is due to that our legalization is designed as a local modification algorithm which does not modify massively the global floorplanning result.

VI. Conclusion

Based on a novel mathematical model for characterizing nonoverlapping of modules and an analytical solution of Poisson's equation, this article has proposed an algorithm without a multilevel framework for fixed-outline floorplanning. The algorithm consists of global floorplanning and legalization phases. In global floorplanning, it uses redefined potential energy based on the novel mathematical model and the analytical solution of Poisson's equation, to realize spreading of modules and optimizing the widths of soft modules. In legalization, the algorithm uses constraint graphs to alternatively eliminate overlaps between modules in the horizontal and vertical directions. Experimental results on the MCNC, GSRC, HB+, and ami49_x benchmarks show that the average HPWLs by our algorithm are at least 2% and 5% less than compared state-of-the-arts on small and large-scale benchmarks, respectively, within acceptable run-time. Considering other issues in our floorplanning algorithm, such as routability, timing, and thermal, are under investigation.

Appendix

Lemma 3: Any two different modules v_i and $v_j \in V$ are nonoverlapping if and only if $\iint_{R_i} \rho_j(u,v)\mathrm{d}u\mathrm{d}v = 0$.

Proof: Since $\rho_j(u,v) \geq 0$, $\iint_{R_i} \rho_j(u,v)\mathrm{d}u\mathrm{d}v = 0$ is equivalent to $\rho_j(u,v) = 0$, for all $(u,v) \in R_i$. This is further equivalent to $R_i \cap R_j = \emptyset$, i.e., modules v_i and $v_j \in V$ are nonoverlapping. ■

Lemma 4: $\iint_{R_i} \rho_i(u,v)\mathrm{d}u\mathrm{d}v = \text{Area}(R_i)$ for all modules $v_i \in V$.

Proof: According to the definition of $\rho_i(u,v)$, it holds that

$$\iint_{R_i} \rho_i(u,v)\mathrm{d}u\mathrm{d}v = \iint_{R_i} 1\mathrm{d}u\mathrm{d}v = \text{Area}(R_i).$$

■

Based on Lemmas 3 and 4, the following theorem can be proved.

Theorem 2: Any module $v_i \in V$ is nonoverlapping with other modules if and only if

$$\iint_{R_i} \sum_{v_j\in V} \rho_j(u,v)\mathrm{d}u\mathrm{d}v = \text{Area}(R_i). \tag{25}$$

Proof: If module $v_i \in V$ is nonoverlapping with other modules, then by Lemmas 1 and 2,

$$\begin{aligned}\iint_{R_i} \sum_{v_j\in V} \rho_j(u,v)\mathrm{d}u\mathrm{d}v &= \sum_{v_j\in V} \iint_{R_i} \rho_j(u,v)\mathrm{d}u\mathrm{d}v\\ &= \iint_{R_i} \rho_i(u,v)\mathrm{d}u\mathrm{d}v + \sum_{\substack{v_j\in V\\ v_j\neq v_i}} \iint_{R_i} \rho_j(u,v)\mathrm{d}u\mathrm{d}v\\ &= \text{Area}(R_i) + 0 = \text{Area}(R_i).\end{aligned}$$

Conversely, if (25) holds, we can have

$$\begin{aligned}\text{Area}(R_i) &= \iint_{R_i} \sum_{v_j\in V} \rho_j(u,v)\mathrm{d}u\mathrm{d}v\\ &= \iint_{R_i} \rho_i(u,v)\mathrm{d}u\mathrm{d}v + \sum_{\substack{v_j\in V\\ v_j\neq v_i}} \iint_{R_i} \rho_j(u,v)\mathrm{d}u\mathrm{d}v\\ &= \text{Area}(R_i) + \sum_{\substack{v_j\in V\\ v_j\neq v_i}} \iint_{R_i} \rho_j(u,v)\mathrm{d}u\mathrm{d}v\end{aligned}$$

where the last equality comes from Lemma 2. Then

$$\sum_{\substack{v_j\in V\\ v_j\neq v_i}} \iint_{R_i} \rho_j(u,v)\mathrm{d}u\mathrm{d}v = 0.$$

Since $\rho_j(u,v) \geq 0$, the above equality implies that $\iint_{R_i} \rho_j(u,v)\mathrm{d}u\mathrm{d}v = 0$ holds for modules $v_j \neq v_i$. Hence by Lemma 1, the conclusion holds. ■

REFERENCES

[1] J.-M. Lin and J.-H. Wu, "F-FM: Fixed-outline floorplanning methodology for mixed-size modules considering voltage-island constraint," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 11, pp. 1681–1692, Nov. 2014.

[2] J.-M. Lin et al., "A fast thermal-aware fixed-outline floorplanning methodology based on analytical models," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, 2018, pp. 1–8.

[3] X. Hong et al., "Corner block list: An effective and efficient topological representation of non-slicing floorplan," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, 2000, pp. 8–12.

[4] E. F. Young, C. C. Chu, and Z. C. Shen, "Twin binary sequences: A nonredundant representation for general nonslicing floorplan," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 4, pp. 457–469, Apr. 2003.

[5] P.-N. Guo, C.-K. Cheng, and T. Yoshimura, "An O-tree representation of non-slicing floorplan and its applications," in *Proc. Design Autom. Conf. (DAC)*, 1999, pp. 268–273.

[6] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu, "B*-trees: A new representation for non-slicing floorplans," in *Proc. Des. Autom. Conf. (DAC)*, 2000, pp. 485–463.

[7] J.-M. Lin, Y.-W. Chang, and S.-P. Lin, "Corner sequence—A P-admissible floorplan representation with a worst case linear-time packing scheme," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 4, pp. 679–686, Aug. 2003.

[8] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "Rectangle-packing-based module placement," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, 1995, pp. 472–479.

[9] J.-M. Lin and Y.-W. Chang, "TCG-S: Orthogonal coupling of P*-admissible representations for general floorplans," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 23, no. 6, pp. 968–980, Jun. 2004.

[10] H. Zhou and J. Wang, "ACG-adjacent constraint graph for general floorplans," in *Proc. Int. Conf. Comput. Design (ICCD)*, 2004, pp. 572–575.

[11] S. N. Adya and I. L. Markov, "Fixed-outline floorplanning: Enabling hierarchical design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 6, pp. 1120–1135, Dec. 2003.

[12] H.-C. Lee, Y.-W. Chang, J.-M. Hsu, and H. Yang, "Multilevel floorplanning/placement for large-scale modules using B*-trees," in *Proc. Design Autom. Conf. (DAC)*, 2003, pp. 812–817.

[13] S. Adya, S. Chaturvedi, J. Roy, D. Papa, and I. Markov, "Unification of partitioning, placement and floorplanning," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, 2004, pp. 550–557.

[14] J. Z. Yan and C. Chu, "DeFer: Deferred decision making enabled fixed-outline floorplanning algorithm," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 3, pp. 367–381, Mar. 2010.

[15] T.-C. Chen, Y.-W. Chang, and S.-C. Lin, "A new multilevel framework for large-scale interconnect-driven floorplanning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 2, pp. 286–294, Feb. 2008.

[16] P. Ji, K. He, Z. Wang, Y. Jin, and J. Wu, "A quasi-Newton-based floorplanner for fixed-outline floorplanning," *Comput. Oper. Res.*, vol. 129, May 2021, Art. no. 105225.

[17] G. Karypis and V. Kumar, "Multilevel *k*-way hypergraph partitioning," in *Proc. Des. Autom. Conf. (DAC)*, 1999, pp. 343–348.

[18] Y. Zhan, Y. Feng, and S. S. Sapatnekar, "A fixed-die floorplanning algorithm using an analytical approach," in *Proc. Asia South Pac. Design Autom. Conf. (ASP-DAC)*, 2006, pp. 771–776.

[19] C. Luo, M. F. Anjos, and A. Vannelli, "Large-scale fixed-outline floorplanning design using convex optimization techniques," in *Proc. Asia South Pac. Des. Autom. Conf. (ASP-DAC)*, 2008, pp. 198–203.

[20] J.-M. Lin and Z.-X. Hung, "UFO: Unified convex optimization algorithms for fixed-outline floorplanning considering pre-placed modules," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 7, pp. 1034–1044, Jul. 2011.

[21] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen, and Y.-W. Chang, "NTUplace3: An analytical placer for large-scale mixed-size designs with Preplaced blocks and density constraints," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 7, pp. 1228–1240, Jul. 2008.

[22] D. Wong and C. Liu, "A new algorithm for floorplan design," in *Proc. Des. Autom. Conf. (DAC)*, 1986, pp. 101–107.

[23] J.-M. Lin, P.-Y. Chiu, and Y.-F. Chang, "SAINT: Handling module folding and alignment in fixed-outline floorplans for 3D ICs," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, 2016, pp. 1–7.

[24] J. Lu et al., "ePlace: Electrostatics-based placement using fast fourier transform and Nesterov's method," *ACM Trans. Design Autom. Electron. Syst.*, vol. 20, no. 2, pp. 1–34, 2015.

[25] W. Zhu, Z. Huang, J. Chen, and Y.-W. Chang, "Analytical solution of Poisson's equation and its application to VLSI global placement," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, 2018, pp. 1–8.

[26] J. Suykens, M. Signoretto, and A. Argyriou, *Regularization, Optimization, Kernels, and Support Vector Machines*. Boca Raton, FL, USA: Chapman Hall/CRC, 2014.

[27] P. Spindler, U. Schlichtmann, and F. M. Johannes, "Abacus: Fast legalization of standard cell circuits with minimal movement," in *Proc. Int. Symp. Phys. Design (ISPD)*, 2008, pp. 47–53.

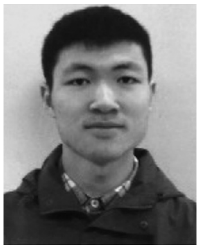
[28] M. D. Moffitt, A. N. Ng, I. L. Markov, and M. E. Pollack, "Constraint-driven floorplan repair," in *Proc. Design Autom. Conf. (DAC)*, 2006, pp. 1103–1108.

[29] J. Cong and M. Xie, "A robust mixed-size legalization and detailed placement algorithm," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 8, pp. 1349–1362, Aug. 2008.

[30] "MCNC floorplan benchmarks." Accessed: Jul. 2001. [Online]. Available: http://vlsicad.eecs.umich.edu/BK/MCNCbench/

[31] "GSRC floorplan benchmarks." Accessed: Jul. 2001. [Online]. Available: http://vlsicad.eecs.umich.edu/BK/GSRCbench/

[32] A. N. Ng, I. L. Markov, R. Aggarwal, and V. Ramachandran, "Solving hard instances of floorplacement," in *Proc. Int. Symp. Phys. Des. (ISPD)*, 2006, pp. 170–177.



Ximeng Li received the B.Sc. degree in computer science from Hangzhou Dianzi University, Hangzhou, China, in 2019, and the M.Sc. degree in applied mathematics from Fuzhou University, Fuzhou, China, in 2022.

He is currently working with Synopsis, Shanghai, China. His research interest is VLSI floorplanning.

Keyu Peng is currently pursuing the master's degree with the Center for Discrete Mathematics and Theoretical Computer Science, Fuzhou University, Fuzhou, China.

His research interests are VLSI placement and floorplanning.

Fuxing Huang is currently pursuing the master's degree with the Center for Discrete Mathematics and Theoretical Computer Science, Fuzhou University, Fuzhou, China.

His research interests are VLSI placement and floorplanning.

Wenxing Zhu received the B.Sc. degree in applied mathematics and the M.Sc. and Ph.D. degrees in operations research from Shanghai University, Shanghai, China, in 1989, 1992, and 1996, respectively.

He joined Fuzhou University, Fuzhou, China, in 1996, and was promoted to professor in 2004. His research interests include algorithms for VLSI physical design automation, optimization theory, and applications.