Global Floorplanning via Semidefinite Programming

Wei Li*, Fangzhou Wang†, José M. F. Moura*, R.D. Blanton*
*Electrical and Computer Engineering Department, Carnegie Mellon University
†Department of Computer Science and Engineering, The Chinese University of Hong Kong

***Abstract*— A major task in chip design involves identifying the location and shape of each major design block/module in the footprint of the chip. This is commonly known as floorplanning. The first step of this task is known as global floorplanning and involves identifying a location for each module that minimizes wire length and leaves sufficient area for each module. Existing global floorplanning methods either have non-convex problem formulation or have trivial global solutions with no guarantee on the quality of the result. We propose to model the global floorplanning problem as a Semi-Definite Programming (SDP) problem with a rank constraint. We replace the rank constraint with a direction matrix and convexify the problem, whose solution is shown to be a global optimum if an appropriate direction matrix is chosen. To calculate the direction matrix, a convex iteration algorithm is used where the problem is decomposed into two SDP sub-problems. Furthermore, we introduce a series of techniques that enhance the flexibility, accuracy, and efficiency of our algorithm. The results show that our proposed method reduces the average wirelength by at least from 3.02% to 20.01% on different benchmarks and outline aspect ratios.**

I. Introduction

As the first stage of VLSI physical design, the quality of floorplanning is critical for the quality of the final design. However, floorplanning is NP-hard. It is not easy to find high-quality locations and shapes at the same time. Most existing methods [1]–[5] divide the problem into two steps: global floorplanning and legalization[1].

Global floorplanning finds the module locations that minimize the wirelength and leaves enough space for each module. The legalization step gives modules concrete shapes. Wirelength is highly related to the module locations, therefore, a good global floorplan is critical for the final floorplanning quality.

In the real world, floorplanning is usually done by highly experienced experts. The problem size is reduced to a manageable level, usually less than 100 IP-cores/blocks. However, floorplanning is still open to better solution: the expert may not have enough time to explore many alternative floorplans. Time pressure and short physical design circle force finishing the floorplanning on just a few days. Providing the expert with a high-quality global floorplan as a starting point will significantly accelerate the design cycle.

Existing global floorplanning methods can be roughly divided into two categories: packing-based methods [3]–[5] and analytical-based methods [1], [2], [7]. Packing-based methods translate the global floorplanning problem into a packing problem. The target (locations in the 2-D plane) is represented by a delicate designed representation such as B*-tree [5], sequence-pair [4], and corner sequence [3].

However, as pointed by Kahng [6], many "complete" representations have difficulty to deal with natural requirements, such as the Pre-Placed Module (PPM) constraint. Moreover, working with a floorplan representation leads to inevitable accuracy loss and evaluation overhead: the wirelength is calculated in the 2-D plane, instead of the designed representation. Analytical-based methods minimize the wirelength by optimizing an objective function in the 2-D plane. But, most existing global floorplanning methods model the floorplanning as a non-convex problem, requiring non-linear optimization methods, which get trapped in a local optimum. Also, many lead to a trivial global optimal solution, for example, where all modules are placed in the same location.

In this paper, we model global floorplanning problem as a Semi-definite Programming (SDP) problem with rank constraint. We replace the rank constraint with an inner product between the target matrix and a direction matrix, which is shown to be global optimal if an appropriate direction matrix is chosen. To calculate the direction matrix, a convex iteration algorithm is used. That decomposes the problem into two SDP sub-problems.

We summarize the contributions of this paper as follows:

- To the best of our knowledge, this is the first time that a global floorplanning problem is formulated as an SDP problem, and also the first time the objective function is convex with a non-trivial global optimal solution.
- We propose a framework that uses a convex iteration algorithm to solve the global floorplanning problem. Furthermore, our framework introduces techniques that enhance the flexibility, accuracy, and efficiency of the algorithm.
- We evaluate the described framework under different conditions. Our results show that the proposed algorithm achieves a better solution than previous global floorplanning algorithms.

The remainder of this paper is organized as follows. Section II formally defines the global floorplanning problem. Section III reviews related work on global floorplanning, specifically analytical-based methods, along with analysis of their limitations. Section IV presents our SDP-based global floorplanning method, comparing it to previous methods. Section V evaluates the proposed method on different conditions and benchmarks. Finally, Section VI concludes the paper.

II. Problem Formulation

In this section, we formally define the global floorplanning problem. The input of the global floorplanning problem is:

- A set of n modules $\{p_0, ..., p_{n-1}\}$, each module is associated with a minimal area constraint s_i, that is, the module is expected to be assigned at least s_i area space in the final floorplan.
- A weighted directed connectivity graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, represented by its adjacency matrix $\boldsymbol{A}$, where A_{ij} is the number of signals passed from p_i to p_j.

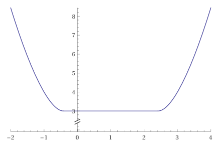
Different from a final floorplan that requires specification of module shapes, the global floorplanning problem only focuses on the locations. It determines the "rough" position of each module, such that wirelength is minimized (wirelength objective) and each module has enough area to be placed in the next step (area constraint). Let $\boldsymbol{x}_i = (x_i, y_i) \in \mathbb{R}^2$ be the center coordinate of module p_i. Collect the center coordinates as the columns of a matrix $\boldsymbol{X}$, i.e., $\boldsymbol{X} = [\boldsymbol{x}_0, ..., \boldsymbol{x}_{n-1}]^T \in \mathbb{R}^{2\times n}$, where n is the number of modules. Formally, the objective is:

$$\min_{\boldsymbol{X}} \sum_{i=0}^{n}\sum_{j=0}^{n} A_{ij} \cdot ||\boldsymbol{x}_i - \boldsymbol{x}_j||_1 \tag{1}$$

$$\text{s.t. each module has enough space to be placed} \tag{2}$$

where A_{ij} is the entry of the adjacency matrix $\boldsymbol{A}$.

[1]In some papers, the global floorplanning is also called initial floorplanning/rough floorplanning/global distribution; the legalization step is sometimes referred to as shape optimization. We follow the naming rule by Kahng [6].


(a) Plot x_i-f_{ij} of AR model

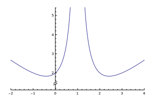
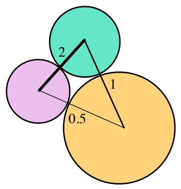
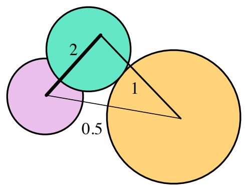

(b) Plot x_i-f_{ij} of PP model

Fig. 1 Plot x_i-f_{ij}, all other variables and parameters are set to 1.



(a)



(b)

Fig. 2 The minimal objective value is expected to happen when circles are tangent (Fig a). But in AR and PP, it may happen when circles are far if the corresponding A_{ij} is small (Fig b). Both numbers and line thickness represent the value of A_{ij}.

III. Related Work

In this section, we introduce the previous global floorplanning methods, and discuss how they represent the area constraint. Moreover, limitations of these methods are discussed.

A. Attractor-Repeller (AR) model [1], [8]

In the AR model, each module p_i is represented by a circle with radius r_i, where r_i is proportional to $\sqrt{s_i}$. As the name suggests, the AR model is composed of two parts: an attractor and a repeller. The attractor is designed so that two modules tend to attract each other if they are connected in the connectivity graph. On the contrary, the repeller prevents two modules from being too close to each other to satisfy the area constraint. Formally, the objective function F is the sum of the attractor and repeller terms of each module pair, i.e., $F = \sum_{i\neq j} f_{ij}$, where f_{ij} is defined as:

$$f_{ij} = \begin{cases} A_{ij}d_{ij} + \frac{t_{ij}}{d_{ij}} - 1 & \text{when } d_{ij} \geq T_{ij} \\ 2\sqrt{A_{ij}t_{ij}} - 1 & \text{when } 0 \leq d_{ij} < T_{ij} \end{cases} \tag{3}$$

Here, $t_{ij} = \sigma \times (r_i + r_j)^2$, σ is a hyperparameter, $d_{ij} = ||\boldsymbol{x}_i - \boldsymbol{x}_j||^2$ is the Euclidean distance square between module p_i and p_j, $T_{ij} = \sqrt{t_{ij}/(A_{ij}+\epsilon)}$, and ϵ is a sufficiently small number. The term $A_{ij}d_{ij}$ is the attractor to minimize the distance, while the term $\frac{t_{ij}}{d_{ij}} - 1$ is the repeller (area constraint). $d_{ij} \rightarrow f_{ij}$ from the first equation in Equation (3) is a monotonic increasing function in the range $[T_{ij}, \infty)$. Therefore, the minimal value of the first equation occurs when $d_{ij} = T_{ij}$, and it is exactly $2\sqrt{c_{ij}t_{ij}} - 1$, the same term in the second equation. This way, the objective function is shown to be convex [8]. Fig. 1(a) is an example plot of x_i-f_{ij} showing its convexity.

Despite being convex, the AR model has several limitations. First, there is no penalty term when two circles (modules) overlap, which causes serious overlap in the global floorplan. Second, the convex AR problem has a trivial global optimal solution, namely, all the circles are placed at the same place. Since the AR optimal solution is trivial, to obtain a valid and good result, a carefully designed line search procedure is required [8] to avoid the trivial global optimum, which is not feasible in practice. In their practical implementation [1], [8], the second equation is not used and a non-linear solver using a gradient-based algorithm is adopted. The final problem of the AR model occurs with its optimal solution. If a module is close to be a square, we expect the optimal value happens when circles are tangent, which is when the distance is minimal and the overlap does not occur. One example is shown is Fig. 2(a). However, for AR, the optimal solution happens when $d_{ij} = T_{ij} = \sqrt{t_{ij}/(A_{ij}+\epsilon)} = \sqrt{\sigma/(A_{ij}+\epsilon)}(r_i + r_j)$. This means that the minimum is related to the value of A_{ij}: when A_{ij} is small, the minimum is when two circles are far away from each other; when A_{ij} is large, the optimal is when two circles overlap, see Fig. 2(b).

B. Push-Pull (PP) model [2], [9]

Similar to the AR model, the PP model also represents the module by a circle, and the objective function is also composed of an attractor (Pull) and a repeller (Push). Let the objective function $F = \sum_{i\neq j} f_{ij}$, f_{ij} be defined as:

$$f_{ij} = \begin{cases} A_{ij}d_{ij} + s_{ij}\left(\frac{r_i+r_j}{d_{ij}} - 1\right) & \text{when } r_i + r_j \geq d_{ij} \\ A_{ij}d_{ij} + \frac{r_i+r_j}{d_{ij}} - 1 & \text{when } r_i + r_j < d_{ij} \end{cases} \tag{4}$$

where $s_{ij} = (r_i \times r_j)^2$, and $d_{ij} = ||\boldsymbol{x}_i - \boldsymbol{x}_j||$ is the Euclidean distance between module p_i and p_j.

The function f_{ij} (Equation (4)) is convex only within an open convex set excluding the intersection between hyperplane $x_i = x_j$ and hyperplane $y_i = y_j$[2]. Fig. 1(b) is a simple example of this non-convexity. In this setting, except for x_i, all other variables and parameters (including x_j) are set to 1, it is clear to see that f_{ij} is not convex with respect to x_i: it is convex only when x_i is restricted to the range $(1, \infty)$ or $(-\infty, 1)$, but not to $\mathbb{R}$. In other words, only when $x_i > x_j$ or $x_i < x_j$ can we say that f_{ij} is convex. Because the objective is non-convex, using a gradient-based method obtains only a local optimum. The second problem with PP is similar to AR (Fig. 2(b)): where the optimal value happens is related to the value of A_{ij}. A small A_{ij} leads to two circles that are far away from each other, which is not desired.

C. Quadratic programming

Quadratic programming (QP) is widely used in global placement [10]–[12] as an initial solution. Li [13] adopts QP in the fixed-outline floorplanning for an initial floorplan. Simply speaking, the objective function in Equation (1) is approximated by a quadratic function using the Euclidean distance square:

$$\min_{\boldsymbol{x},\boldsymbol{y}} \frac{1}{2}\boldsymbol{x}^T\boldsymbol{C}\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{d} + \frac{1}{2}\boldsymbol{y}^T\boldsymbol{C}\boldsymbol{y} + \boldsymbol{y}^T\boldsymbol{d} \tag{5}$$

Here, $\boldsymbol{x}, \boldsymbol{y}$ are the x-coordinates and y-coordinates of the modules, respectively, $\boldsymbol{C}$ represents the connectivity between movable modules, and $\boldsymbol{d}$ is the vector defining the connections between fixed and movable modules. Reference [11] provides details about $\boldsymbol{C}$ and $\boldsymbol{d}$.

Objective (5) is convex, and the global optimal solution can be obtained efficiently. However, it only contains an attraction force, resulting in significant overlap in the final result. Moreover, when all modules are movable, i.e., $\boldsymbol{b} = 0$, the convex objective is reduced to: $\frac{1}{2}\boldsymbol{x}^T\boldsymbol{C}\boldsymbol{x} + \frac{1}{2}\boldsymbol{y}^T\boldsymbol{C}\boldsymbol{y}$ that always gives a trivial solution: all modules are placed at the same position.

IV. SDP-based Convex Iteration

In this section, we introduce our solution (Section IV-A) and enhancing algorithm (Section IV-B), and we present an overall algorithm (Section IV-C). Finally, we compare with previous methods (Section IV-D).

A. Iterative SDP-based optimization

We will first introduce how the problem is reformulated as a "nearly-convex" optimization problem and then solve it by an SDP-based convex iteration [14].

In global floorplanning, shapes of modules are unknown, each module p_i is modeled as a circle whose center is $\boldsymbol{x}_i \in \mathbb{R}^2$ with radius

[2]The proof in [2] does not hold when $d_{ij} = 0$. The derivative and Hessian of f_{ij} do not exist when $d_{ij} = 0$ (Lemma 1 in [2]), therefore: f_{ij} is not a twice continuously differentiable function over $\mathbb{R}^4$ (Corollary 1 in [2]).

$r_i = \sqrt{s_i/4}$, where s_i is the minimum area constraint of module p_i. Due to the absence of shapes, we only care about minimizing the wirelength. At the same time, the area constraint is expected to be satisfied as much as possible, i.e., each module should be left with enough area for the final floorplan. Formally, the wirelength among modules is estimated by:

$$\langle \boldsymbol{A}, \boldsymbol{D} \rangle \tag{6}$$

where $\langle \cdot, \cdot \rangle$ is an inner product, and $\boldsymbol{A}$ is the adjacent matrix of the connectivity graph. $\boldsymbol{D}$ is the Euclidean distance matrix, and the entry D_{ij} is the Euclidean distance square between p_i and p_j, i.e, $D_{ij} = ||\boldsymbol{x}_i - \boldsymbol{x}_j||^2$.

Define the *Gram matrix* $\boldsymbol{G} = \boldsymbol{X}^T\boldsymbol{X} \in \mathbb{S}_+^n$, where $\mathbb{S}_+^n$ represents the positive semi-definite matrix. Since $D_{ij} = G_{ii} + G_{jj} - G_{ij} - G_{ji}$, the objective can be rewritten as:

$$\langle \boldsymbol{B}, \boldsymbol{G} \rangle \tag{7}$$

where $\boldsymbol{B} \in \mathbb{R}^{n \times n}$ is defined by:

$$B_{ij} = \begin{cases} \sum_{k=0}^{n-1} A_{ik} + \sum_{k=0}^{n-1} A_{kj} & \text{when } i = j \\ -2A_{ij} & \text{when } i \neq j \end{cases} \tag{8}$$

Note that $\boldsymbol{B}$ is a constant known matrix since $\boldsymbol{A}$ is given. We further define $\boldsymbol{Z}$ as:

$$\boldsymbol{Z} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{X} \\ \boldsymbol{X}^T & \boldsymbol{G} \end{bmatrix} \in \mathbb{S}_+^{2+n} \tag{9}$$

Using $\boldsymbol{Z}$, the constraint $\boldsymbol{G} = \boldsymbol{X}^T\boldsymbol{X}$ is relaxed to $\boldsymbol{G} \succeq \boldsymbol{X}^T\boldsymbol{X}$, which is exactly $\boldsymbol{Z} \succeq 0$ by Schur's complement. Given all definitions described, the objective and constraints can be formulated in matrix form as:

Main problem:

$$\min_{\boldsymbol{Z}} \langle \boldsymbol{B}, \boldsymbol{G} \rangle \tag{10}$$

$$\text{s.t.} \quad D_{ij} \geq (r_i + r_j)^2 \quad \forall i, j \in \{0, ..., n-1\} \tag{11}$$

$$\boldsymbol{Z} \succeq 0$$

$$\text{rank}(\boldsymbol{Z}) = 2 \tag{12}$$

where Equation (11) guarantees that the Euclidean distance between two circles is larger than the sum of their radius (area constraint). $\boldsymbol{Z} \succeq 0$ is a relaxation of $\boldsymbol{G} = \boldsymbol{X}^T\boldsymbol{X}$, and Equation (12) makes sure that the equation ($\boldsymbol{G} = \boldsymbol{X}^T\boldsymbol{X}$) holds.

Without the rank constraint in Equation (12), the problem is obviously an SDP problem[3]. However, the rank constraint makes the problem non-convex. We replace the rank constraint using a direction matrix $\boldsymbol{W}_{\text{opt}}$. Formally, the main problem can be rewritten as:

Convexified problem:

$$\min_{\boldsymbol{Z}} \langle \boldsymbol{B}, \boldsymbol{G} \rangle + \alpha \langle \boldsymbol{W}_{\text{opt}}, \boldsymbol{Z} \rangle \tag{13}$$

$$\text{s.t.} \quad D_{ij} \geq (r_i + r_j)^2 \quad \forall i, j \in \{0, ..., n-1\}$$

$$\boldsymbol{Z} \succeq 0$$

Here, α is a hyper-parameter controlling the rank constraint penalty. With an appropriate $\boldsymbol{W}_{\text{opt}}$, i.e., $\langle \boldsymbol{W}_{\text{opt}}, \boldsymbol{Z} \rangle = 0$, the two problems are *equivalent*, which means the global optimal solution in the convexified problem is also equivalent to the global optimum in the main problem.

The intuitive explanation for $\boldsymbol{W}_{\text{opt}}$ is as follows. The rank constraint ($\text{rank}(\boldsymbol{Z}) = 2$) can be rewritten as:

$$\text{trace}(\boldsymbol{Z}) - \lambda_0 - \lambda_1 = 0 \tag{14}$$

where λ_0 and λ_1 are the two largest eigenvalues of $\boldsymbol{Z}$, and can be calculated by:

$$\lambda_0 = \max_{||\boldsymbol{u}_0||=1} \boldsymbol{u}_0^T \boldsymbol{Z} \boldsymbol{u}_0 \quad \text{and} \quad \lambda_1 = \max_{||\boldsymbol{u}_1||=1, \boldsymbol{u}_1 \perp \boldsymbol{u}_0} \boldsymbol{u}_1^T \boldsymbol{Z} \boldsymbol{u}_1 \tag{15}$$

[3]Note that $D_{ij} = G_{ii} + G_{jj} - G_{ij} - G_{ji}$ is also an inner product between some constant matrix $\boldsymbol{C}$ and the Gram matrix $\boldsymbol{G}$.

Then, the rank constraint Equation (15) can be rewritten as:

$$\left\{ \boldsymbol{Z} : \min_{||\boldsymbol{u}_0||=1, ||\boldsymbol{u}_1||=1, \boldsymbol{u}_1 \perp \boldsymbol{u}_0} \langle \boldsymbol{I} - [\boldsymbol{u}_0, \boldsymbol{u}_1][\boldsymbol{u}_0, \boldsymbol{u}_1]^T, \boldsymbol{Z} \rangle = 0 \right\} \tag{16}$$

The above is equivalent to

$$\left\{ \boldsymbol{Z} : \min_{\boldsymbol{I} \succeq \boldsymbol{W} \succeq 0} \langle \boldsymbol{W}, \boldsymbol{Z} \rangle = 0 \right\} \tag{17}$$

Adding Equation (17) back to Equation (7) with a coefficient α, the rank constraint is re-written as a bilinear term in the objective function.

Consider an optimal solution $\boldsymbol{Z}_{\text{opt}}$ for the main problem. Selecting $\boldsymbol{W}_{\text{opt}} = \boldsymbol{U}_{\text{opt}} * \boldsymbol{U}_{\text{opt}}^T$ makes the convexified problem equivalent to the main problem, where columns of $\boldsymbol{U}_{\text{opt}}$ are the normalized eigenvectors of $\boldsymbol{Z}_{\text{opt}}$ corresponding to the n smallest eigenvalues. However, this is infeasible since $\boldsymbol{Z}_{\text{opt}}$ is unknown and needs to be determined. To obtain $\boldsymbol{W}_{\text{opt}}$ and corresponding optimal solution $\boldsymbol{Z}_{\text{opt}}$, we decompose it into two SDP sub-problems [14]:

Sub-probem 1:

$$\min_{\boldsymbol{Z} \in \mathbb{S}^{2+n}} \langle \boldsymbol{B}, \boldsymbol{G} \rangle + \alpha \langle \boldsymbol{W}^*, \boldsymbol{Z} \rangle \tag{18}$$

$$\text{s.t.} \quad D_{ij} \geq (r_i + r_j)^2 \quad \forall i, j \in \{0, ..., n-1\}$$

$$\boldsymbol{Z} \succeq 0$$

Sub-probem 2:

$$\min_{\boldsymbol{W} \in \mathbb{S}^{2+n}} \langle \boldsymbol{W}, \boldsymbol{Z}^* \rangle \tag{19}$$

$$\text{s.t.} \quad \boldsymbol{I} \succeq \boldsymbol{W} \succeq 0$$

$$\text{trace}(\boldsymbol{W}) = n$$

The two sub-problems are solved iteratively, that is, $\boldsymbol{W}^*$ is the optimal solution from Equation (19) and $\boldsymbol{Z}^*$ is the optimal solution from Equation (18) respectively. At the first iteration, $\boldsymbol{W}^*$ is initialized as $\boldsymbol{W}^0 = \boldsymbol{I}$, which reduces $\langle \boldsymbol{W}^*, \boldsymbol{Z} \rangle$ in Equation (18) to a trace heuristic for minimizing the rank using the 1-norm of the diagonal entries, whose 0-norm is the rank. The iteration is stopped when the objective values of Equation (18) and Equation (19) are not improved. When the iteration converges to a point such that $\langle \boldsymbol{W}^*, \boldsymbol{Z}^* \rangle = 0$, the rank constraint is satisfied, and we get the global optimal solution of the convexified problem.

B. Enhancement to the algorithm

In the real world, there are many other requirements that need to be considered, e.g., boundary I/O pads. In this section, we will develop techniques that enhance the flexibility, accuracy, and efficiency of the algorithm.

a) Euclidean distance square to Manhattan distance

In our objective function (Equation (6)), we use an Euclidean distance square matrix $\boldsymbol{D}$ to estimate the wirelength. The Manhattan distance may estimate the wirelength better. Sometimes, an optimal solution that minimizes Euclidean distance square does not guarantee optimal wirelength, and may even be a bad solution in terms of wirelength.

To address this issue, we propose to use an adaptive $\boldsymbol{B}$ matrix that changes during the iterations. Let $D_{ij}^{(t)}, M_{ij}^{(t)}$ be the Euclidean distance square and Manhattan distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ at iteration t, respectively. The basic idea is that, at each iteration t, rather than minimize the cost $c_{ij}^{(t)} = A_{ij} D_{ij}^{(t)}$ that is quantified by Euclidean distance square, we minimize an adaptive cost $c_{ij}^{\prime(t)}$ calculated by:

$$c_{ij}^{\prime(t)} = \frac{M_{ij}^{(t-1)}}{D_{ij}^{(t-1)}} c_{ij}^{(t)} = \frac{M_{ij}^{(t-1)}}{D_{ij}^{(t-1)}} A_{ij} D_{ij}^{(t)} \tag{20}$$

Since the Manhattan distance and Euclidean distance square between two modules do not change significantly for any single iteration, the adaptive cost $c_{ij}^{\prime(t)}$ is close to the cost quantified by real wirelength.

To do this, for $t > 0$, $A_{ij}^{(t)}$ is updated as: $A_{ij}^{(t)} = \frac{M_{ij}^{(t-1)}}{D_{ij}^{(t-1)}} A_{ij}$, and $\boldsymbol{B}$ at each iteration is also updated based on $\boldsymbol{A}^{(t)}$.

The technique can be extended to hyper-edges. Let $e = \{p_e^0, ..., p_e^k\}$ be the hyper-edge connecting k modules, then, at each iteration, e only influences A_{ij} if i and j are connected by e, and both of them are on the boundary of the e's bounding box at the last iteration. Reference [11] provides a detailed description of the hyper-edge case.

b) Boundary pin and fixed outline

Sometimes, modules are connected to boundary pins, e.g. I/O pads. These boundary pins can be inserted into our system without adding variables. Let us say that there are m boundary pins, and their center coordinates form a matrix $\overline{\boldsymbol{X}} \in \mathbb{R}^{2\times m}$, $\overline{\boldsymbol{A}} \in \mathbb{R}^{n\times m}$ is the matrix storing the connectivity information from modules to boundary pins, i.e., $\overline{A}_{ij}$ is the number of signal passed from module p_i to j_{th} boundary pin. The objective function between p_i to j_{th} boundary pin is estimated by their Euclidean distance square: $\overline{c_{ij}} = \overline{A}_{ij}\overline{D}_{ij}$, where $\overline{D}_{ij} = ||\boldsymbol{x}_i - \overline{\boldsymbol{x}}_j||^2 = G_{ii} - 2\boldsymbol{x}_i\overline{\boldsymbol{x}}_j + \overline{\boldsymbol{x}}_j^2$. Formally, minimizing $\overline{c_{ij}}$ is equivalent to:

$$\min_{\boldsymbol{x}_i} \langle \overline{\boldsymbol{B}}, \boldsymbol{Z} \rangle \tag{21}$$

where $\overline{\boldsymbol{B}}$ is all-zero except 1) $\overline{B_{2+i,2+i}} = \overline{A}_{ij}$; 2) $\overline{B}_{2+i,:2} = -2\overline{\boldsymbol{x}}_j^T$; 3) $\overline{B}_{:2,2+i} = -2\overline{\boldsymbol{x}}_j$. Integrating Objective (21) into the original objective, the wirelength between modules to boundary pins is also minimized. Another special case is a given outline of the chip, a simple incorporation is to add a lower bound and upper bound of $\boldsymbol{X}$, which is just a part of $\boldsymbol{Z}$.

c) PPM constraint

Sometimes, modules are pre-placed. This is called a Pre-Placed Module (PPM) constraint. If a module p_i is fixed, then $\boldsymbol{x}_i$ will have a given fixed value. Our formulation can be easily extended to handle PPM constraints by adding equation constraints in the main problem and the first sub-problem. After adding PPM constraints, the main problem becomes:

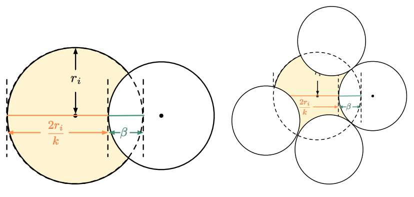
Main problem with PPM:

$$\begin{aligned}
&\min_{\boldsymbol{Z}} \langle \boldsymbol{B}, \boldsymbol{G} \rangle && (22)\\
\text{s.t.}\quad & D_{ij} \geq (r_i + r_j)^2 \quad \forall i, j \in \{0, ..., n-1\} \\
& G_{ij} = \boldsymbol{x}_i^T \boldsymbol{x}_j \quad \forall i, j \text{ where } p_i \text{ and } p_j \text{ are fixed} && (23)\\
& \boldsymbol{X}_{:,i} = \boldsymbol{x}_i \quad \forall i \text{ where } p_i \text{ is fixed} && (24)\\
& \boldsymbol{Z} \succeq 0 \\
& \text{rank}(\boldsymbol{Z}) = 2
\end{aligned}$$

The problem above is also decomposed into two sub-problems and solved iteratively, where the two new distance constraints (Equation (23) and Equation (24)) are added to the first sub-problem.

d) Non-square requirement

In our formulation, each module is modeled as a circle based on prior knowledge that a module is expected to be like a square. But, in some cases, a module can be a rectangle. For example, the soft module in fixed-outline floorplanning is a rectangle that has fixed area but a flexible aspect ratio. In these cases, a circle is not a good approximation, and strict distance constraints in Equation (11) may even harm the solution quality. To address this issue, we propose an adaptive distance constraint that is based on the maximum aspect ratio of the modules and the value of A_{ij}. Consider Fig. 3, the orange circle p_i havs radius r_i. Let k be the maximal aspect ratio of the module. We approximate the "forbidden zone" of p_i as $\frac{2r_i}{k}$ (the orange line in Fig. 3), which represents the length of the segment that cannot be occupied by other modules. Because $\frac{2r_i}{k} + \beta = 2r_i$ and the distance between two circle centers is $r_i + r_j - \beta$, the distance constraint can be written as:



(a) (b)

Fig. 3 The illustration of an adaptive distance constraint. (a) The orange part is an approximation of a rectangle with height $2r_i$ and width $\frac{2r_i}{k}$; (b) Multiple neighbors cause the circle to be squashed.

$$D_{ij} \geq \left(r_j - r_i + \frac{2r_i}{k}\right)^2 \tag{25}$$

When k is 1, i.e., the module is expected to be a square, Equation (25) is equivalent to Equation (11). When k is greater than 1, e.g., $k = 3$, the module has a forbidden segment with length at least $\frac{2r_i}{3}$, which mimics a rectangle with width $\frac{2r_i}{3}$ and height $2r_i$. To make $2r_i\frac{2r_i}{k} = s_i$, r_i is set to $\sqrt{\frac{ks_i}{4}}$. When p_i is connected with multiple modules in $\mathcal{G}$, Equation (25) should be adjusted: otherwise, the circle might be squashed by other modules (See Fig. 3(b)). To do this, k in Equation (25) is replaced by k_{ij}, which is defined by $k_{ij} = \frac{A_{ij}}{\sum_{l\in\mathcal{N}_i} A_{il}}(k-1)+1$, where $\mathcal{N}_i$ is the neighbors of i in $\mathcal{G}$. The intuition is, if two modules are closely connected, i.e., A_{ij} is large, then p_j is "allowed" to be closer to p_i, but is always upper bounded by $k_{ij} \leq k$. Furthermore, since $D_{ij} = D_{ji}$, the distance constraint can be formalized by:

$$D_{ij} \geq \max\left(\left(r_j - r_i + \frac{2r_i}{k_{ij}}\right)^2, \left(r_i - r_j + \frac{2r_j}{k_{ji}}\right)^2\right) \tag{26}$$

C. Overall Algorithm

Here, we present the overall algorithm combining the basic algorithm in Section IV-A and the enhancing techniques in Section IV-B. The algorithm is summarized in Algorithm 1. Ideally, we expect to use the smallest α that finds a feasible $\boldsymbol{Z}$ with rank 2. To achieve this, we start from a small α and increase it by a factor of 2 until the rank constraint is satisfied. For each α (Line 2 - 12), we iteratively solve the two sub-problems as described in Section IV-A. Some techniques described in Section IV-B are applied in this iterative process. For example, the constant matrix $\boldsymbol{B}$ is updated each iteration (Line 9) to estimate the Manhattan distance rather than a distance square (See Paragraph IV-B0a). When the solution of the two sub-problems converge or an iteration limit is reached (Line 10), we check the rank of $\boldsymbol{Z}$ (Line 12). If the rank is 2, we stop the algorithm and return the solution. Otherwise, we increase the rank constraint penalty, α, and repeat the process.

D. Comparison with other methods

In this section, we follow the discussion in Section III, and compare our method with others from the following perspectives.

Convexity and non-trivial global optimum. As stated before, both QP [13] and AR [1], [8] have convex problem formulations. However, their global optimal solution is trivial most of the time, placing all module centers at the same location is the global optimum. In contrast, the objective function of PP [2] is non-convex for the solution space, but the global optimum is not a trivial same-location solution. Compared with these methods, our SDP-based method divides the problem into two sub-problems, where both of them are convex and have a non-trivial global optimum.

Controllable area constraint. Another limitation of previous work is the lack of control over the area constraint, i.e, making sure that each module has enough space to be placed. QP does

Algorithm 1 `SDPGlobalFloorplanning`

```
Require: $\alpha \rightarrow$ the penalty coefficient for the rank constraint term in Objective 18;
Require: $\boldsymbol{A} \rightarrow$ the adjacent matrix of the connectivity graph $\mathcal{G}$;
Require: $\boldsymbol{s} \rightarrow$ the minimal area constraint of each module;
Require: $\epsilon \rightarrow$ A very small number;
Require: $max\_iter \rightarrow$ Maximal allowed iteration;
 1: repeat
 2:     $t \leftarrow 0$;                                        ▷ iteration counter
 3:     $\boldsymbol{W}^{(0)} \leftarrow \boldsymbol{I}$;        ▷ initialization using a trace heuristic
 4:     $\boldsymbol{B}^{(0)} \leftarrow$ Get $\boldsymbol{B}$ using Equation (8);
 5:     repeat
 6:         $t \leftarrow t+1$;
 7:         $\boldsymbol{Z}^{(t)} \leftarrow$ Solve sub-problem 1 using $\boldsymbol{W}^{(t-1)}$;
 8:         $\boldsymbol{W}^{(t)} \leftarrow$ Solve sub-problem 2 using $\boldsymbol{Z}^{(t)}$;
 9:         $\boldsymbol{B}^{(t)} \leftarrow$ Update $\boldsymbol{B}$ following Equation (20);
10:     until $||\boldsymbol{Z}^{(t)} - \boldsymbol{Z}^{(t-1)}|| + ||\boldsymbol{W}^{(t)} - \boldsymbol{W}^{(t-1)}|| < \epsilon$ or $t \geq max\_iter$
11:     $\alpha \leftarrow \alpha \times 2$;   ▷ increase the rank constraint penalty coefficient
12: until $\langle \boldsymbol{W}^{(t)}, \boldsymbol{Z}^{(t)} \rangle < \epsilon$   ▷ the rank constraint is satisfied
13: return $\boldsymbol{Z}^{(t)}[2:,:2]$;                     ▷ the final $\boldsymbol{X}$
```

	QP	AR [1], [8]	PP [2]	Ours
Convex	Yes	Yes	No	Yes
Non-trival optimal	Depends	No	Yes	Yes
Area constraint	No	Partly	Partly	Controllable
Quality	Poor	Good	Good	Near optimal
Efficiency	Very fast	Fast	Fast	Poor

TABLE I Comparison with other methods.

not consider the area constraint in the objective function, both AR and PP introduce a repeller force to push the modules away from each other, and the repeller force combats the attractor force that is affected by A_{ij}. These soft constraints lose the control of the distance between modules, which is important in some cases, e.g., the timing requirement for some paths. As a side effect, the final results either generate too much overlap, or too much distance between modules: when the attractor force is too weak, i.e., A_{ij} is small, the repeller force pushes two modules to a stable point that is far from each other. On the contrary, our method can directly control the distance, i.e., add $D_{ij} \geq ...$ or $D_{ij} \leq ...$ to the constraint. One example is the non-square requirement introduced in Paragraph IV-B0d.

Result quality. AR and PP are solved by a non-linear optimization package and easy to be trapped in a local optimum. The objective of QP does not consider the area constraint, which results in large overlapping. Therefore, an innegligible effort is still needed to satisfy the area constraint. Compared with these methods, our algorithm is shown to be global optimal if an appropriate $\boldsymbol{W}$ is selected [14]. Although the convex iteration only guarantees a local optimum, previous experiments on other fields show that the convergence happens at the global optimum most of the time [15], [16].

Efficiency. QP is the fastest method among these methods, since it can be solved by a single run of some off-the-shelf convex solver. AR and PP are also fast enough using some gradient-based optimization method. However, our SDP-based method is much slower than these methods: the SDP sub-problem in each iteration has an exponential time complexity, and the number of iterations for convergence also increases with the problem size according to our experiments. The exponential time complexity inhibits the extension of our method to large-scale problems.

V. Experimental Results

Our algorithm along with other state-of-the-art global floorplanning methods are implemented in Python. The convex optimization solver is MOSEK 10.0 [17]. For non-linear optimization, PyTorch Minimize [18] with BFGS algorithm is used. A legalization framework similar to previous works [2], [19] is implemented: with the global floorplan as the input, horizontal and vertical constraint graphs are constructed, then the shape optimization is modeled as a second order cone programming instance. Final HPWL is reported after the legalization. MCNC and GSNC are used for experiments. The aspect ratio constraint for modules is between 1/3 and 3. All experiments are run on a 64-core server with 2.2GHz CPU. We emphasize that global floorplanning is not limited to the automated fixed-outline floorplanning. A high-quality global floorplan with an accurate center coordinates approximation is helpful for manual floorplanning.

We first explore the property of our algorithm, and study how the proposed techniques influence the result. Effect of α on the result quality is given in Fig. 4. In general, larger benchmark requires a larger α. Fig. 4 also shows the effect of enhancement techniques. The non-square technique improves the result significantly except for the very small case (n10). Manhattan distance and hyper-edge approximation further improve the result on the basis of the non-square technique. α also influences the convergence rate and the final result quality. As shown in Fig. 5(a), larger α leads to faster convergence, but may get worse result. The convergence is also affected by the problem size. Larger benchmarks need a larger α to converge: $\alpha = 1024$ for n10 converges within 10 iterations, but the objective value is still decreasing for n50 and n100. The result in Fig. 5 shows the runtime per iteration increases exponentially with the problem size. From the two figures, our algorithm may not be scalable to large case, where well-studied placement techniques [10], [13] are recommended.

We also compare different global floorplanning methods on the fixed-outline floorplanning problem. In our algorithm, max_iter is 50, α starts from 0.5 except n100 and n200, whose max_iter is 100/20 and α starts from 1024. As shown in TABLE II, comparing with AR [1] and PP [9], ours reduces HPWL on average by 14.71% and 15.58% on aspect ratio 1:1 and 14.59% and 20.10% on aspect ratio 1:2, respectively. We also compare with other two methods, as shown in TABLE III. [7] is an analytical method based on non-convex density control. Parquet-4 [20] is a simulated annealing based method. Ours achieves better results than others on both ratios, which demonstrates the robustness of our algorithm. However, our advantage is less significant on large cases. n100 is a sweet spot for previous algorithm, where [7] outperforms ours. One reason might be the post-process adopted in [7]. Nevertheless,for larger problem sizes (n200), non-convex methods become trapped in local optima, while our algorithm gives better with only 20 iterations (about 2.5 h solve time).

VI. Conclusion

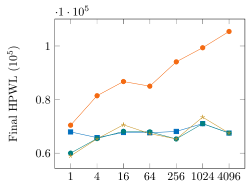
In this paper, we propose an SDP-based iterative algorithm that divides the problem into two convex sub-problems. We also design enhancement techniques to improve the flexibility, accuracy, and efficiency of the algorithm. The results show that our proposed method achieves better performance than previous algorithms. How to accelerate the algorithm, or design a hierarchical framework to enhance the scalability will be future work.

Acknowledgments

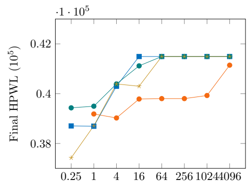
We would like to express our sincere gratitude to Ala Qumsieh, Dongjin Lee, and Zhuolun He for their invaluable assistance throughout this work.

References

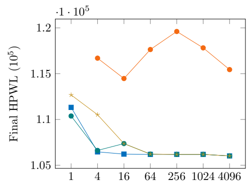
[1] C. Luo, M. F. Anjos, and A. Vannelli, "Large-scale fixed-outline floorplanning design using convex optimization techniques," in *2008 Asia and South Pacific Design Automation Conference*. IEEE, 2008, pp. 198–203.



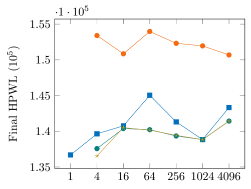
(a) ami33



(b) n10



(c) n30



(d) n50

Fig. 4 α-HPWL Plot. Orange is the basic algorithm described in Section IV-A; Blue uses non-square technique (Equation (26)); Green uses non-square technique and Manhattan distance approximation (Equation (20)); Yellow uses non-square technique, Manhattan distance approximation, and hyper-edge approximation [11]. Missing points mean the failure during legalization.

TABLE II HPWL comparison with AR [1] and PP [9].

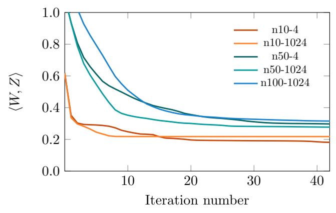
Statistics			1:1					1:2				
			Ours	AR [1]		PP [9]		Ours	AR [1]		PP [9]	
	block #	net #	Wire.	Wire.	Δ(%)	Wire.	Δ(%)	Wire.	Wire.	Δ(%)	Wire.	Δ(%)
n10	10	118	36277	38593*	3.84	45193	24.57	37185	38781*	4.29	43386*	16.67
n30	30	349	106013	117245*	10.59	122540	15.58	103857	118197*	13.80	129810*	24.99
n50	50	485	136547	149859*	9.78	161990	18.63	133341	158743*	19.05	181215*	35.90
n100	100	885	228040	285070	25.01	259440	13.76	238922	262741*	17.72	262700	10.00
n200	200	1585	407091	506070	24.31	428900	5.35	412469	451249*	18.10	465800	12.93
avg. Δ					14.71		15.58			14.59		20.10

Results of AR and PP are from [9], except those marked by *, which are implemented by us since they are not included in [9]. Implemented version shares the same legalization algorithm with ours. I/O pads are fixed on the boundary of the chips.

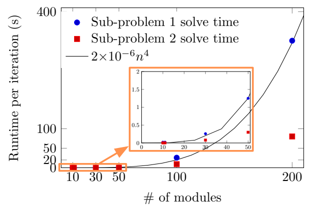
TABLE III HPWL comparison with Parquet-4 [20] and Analytical [7].

			ami33	ami49	n100	n200	avg. Δ
1:1	Ours	Wire.	**65485**	801402	304246	**553487**	
	Parquet-4 [20]	Wire.	82149	928597	342103	630014	
		Δ(%)	25.45	15.87	12.44	13.83	16.89
	Analytical [7]	Wire.	74072	**799239**	**291628**	572145	
		Δ(%)	13.11	-0.27	-4.14	3.37	3.02
1:2	Ours	Wire.	**66761**	**743685**	310178	**553157**	
	Parquet-4 [20]	Wire.	79131	942117	351542	645219	
		Δ(%)	18.53	26.68	13.33	14.57	18.23
	Analytical [7]	Wire.	75168	829888	**290158**	565927	
		Δ(%)	12.59	11.59	-6.45	0.49	4.56

Results of Parquet-4 and Analytical are from [7]. I/O pads are fixed on the locations given by the benchmark. The two algorithms are post-processed by *pl2sp()* function in Parquet-4, but ours is not.



(a) Convergence of subproblem 2 (Equation (19)). The values are normalized.



(b) Runtime per iteration.

Fig. 5 Convergence and runtime results. Line in (a) represents a benchmark-α setting. Line in (b) is a function proportional with n^4.

[2] J.-M. Lin and Z.-X. Hung, "Ufo: Unified convex optimization algorithms for fixed-outline floorplanning considering pre-placed modules," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 7, pp. 1034–1044, 2011.

[3] 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 Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 4, pp. 679–686, 2003.

[4] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "Vlsi module placement based on rectangle-packing by the sequence-pair," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 12, pp. 1518–1524, 1996.

[5] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu, "B*-trees: A new representation for non-slicing floorplans," in *Proceedings of the 37th Annual Design Automation Conference*, 2000, pp. 458–463.

[6] A. B. Kahng, "Classical floorplanning harmful?" in *Proceedings of the 2000 international symposium on Physical design*, 2000, pp. 207–213.

[7] Y. Zhan, Y. Feng, and S. S. Sapatnekar, "A fixed-die floorplanning algorithm using an analytical approach," in *Proceedings of the 2006 Asia and South Pacific Design Automation Conference*, 2006, pp. 771–776.

[8] M. F. Anjos and A. Vannelli, "An attractor-repeller approach to floorplanning," *Mathematical Methods of Operations Research*, vol. 56, no. 1, pp. 3–27, 2002.

[9] J.-M. Lin and H. Hung, "Ufo: Unified convex optimization algorithms for fixed-outline floorplanning," in *2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2010, pp. 555–560.

[10] C.-K. Cheng, A. B. Kahng, I. Kang, and L. Wang, "Replace: Advancing solution quality and routability validation in global placement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 9, pp. 1717–1730, 2018.

[11] P. Spindler, U. Schlichtmann, and F. M. Johannes, "Kraftwerk2—a fast force-directed quadratic placement approach using an accurate net model," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 8, pp. 1398–1411, 2008.

[12] 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 Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 7, pp. 1228–1240, 2008.

[13] X. Li, K. Peng, F. Huang, and W. Zhu, "Pef: Poisson's equation based large-scale fixed-outline floorplanning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2022.

[14] J. Dattorro, *Convex optimization & Euclidean distance geometry*. Lulu. com, 2010.

[15] W. Wang and N. Yu, "Chordal conversion based convex iteration algorithm for three-phase optimal power flow problems," *IEEE Transactions on Power Systems*, vol. 33, no. 2, pp. 1603–1613, 2017.

[16] L. T. Nguyen, J. Kim, and B. Shim, "Low-rank matrix completion: A contemporary survey," *IEEE Access*, vol. 7, pp. 94 215–94 237, 2019.

[17] M. ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 10.0.*, 2022. [Online]. Available: http://docs.mosek.com/10.0/toolbox/index.html

[18] R. Feinman, "Pytorch-minimize: a library for numerical optimization with autograd," 2021. [Online]. Available: https://github.com/rfeinman/pytorch-minimize

[19] S. Kai, C.-W. Pui, F. Wang, S. Jiang, B. Wang, Y. Huang, and J. Hao, "Tofu: A two-step floorplan refinement framework for whitespace reduction," *IEEE/ACM Design, Automation, and Test in Europe (DATE)*, pp. 1–6, 2023.

[20] S. N. Adya and I. L. Markov, "Fixed-outline floorplanning: Enabling hierarchical design," *IEEE Transactions on very large scale Integration (VLSI) systems*, vol. 11, no. 6, pp. 1120–1135, 2003.