iPL-3D: A Novel Bilevel Programming Model for Die-to-Die Placement

Xueyan Zhao[1,3,2], Shijian Chen[2,1], Yihang Qiu[4,2], Jiangkao Li[5,2], Zhipeng Huang[2,*], Biwei Xie[1,2], Xingquan Li[5,2], and Yungang Bao[1,3,2]
[1]State Key Laboratory of Processors, Institute of Computing Technology, CAS, Beijing, China
[2]Peng Cheng Laboratory, Shenzhen, China
[3]University of Chinese Academy of Sciences, Beijing, China
[4]School of Integrated Circuits, Guangdong University of Technology, Guangzhou, China
[5]School of Mathematics and Statistics, Minnan Normal University, Zhangzhou, China
Email: zhaoxueyan21b@ict.ac.cn, *huangzhp@pcl.ac.cn, xiebiwei@ict.ac.cn, fzulxq@gmail.com, baoyg@ict.ac.cn

***Abstract*—Die-to-die (D2D) placement is a more challenging stage in achieving higher performance with complex constraints, critically impacting timing, power, yield, cost, etc. Existing placers often rely on indirect objectives (e.g., considering cut sizes in tier assignment), which can lead to a loss of the overall solution space utilization and may even deviate from the actual objective. To address this issue, this paper leverages the natural dominance relationship between decision variables to transform the original problem into a bilevel programming problem equivalently. Additionally, an alternating optimization framework is introduced to enhance the exploration of the overall solution space. On the one hand, we propose two tier optimization operators for simultaneous optimization of wirelength and #terminal in global and detailed perspectives; On the other hand, we present a near-optimal terminal legalization algorithm following an efficient multi-tier co-placement. Compared with the top three winners of the ICCAD'22 contest, our placer achieves 4.33%, 4.42%, and 5.88% smaller wirelength, 79.61%, 16.74%, and 15.76% fewer #terminal and competitive runtime. Moreover, our placer always uses the fewest #terminal and achieves amazing wirelength reduction when the terminal size changes.**

I. Introduction

The quality of 3D IC flow is significantly influenced by placement. In recent years, various 3D integration methods (e.g., hybrid wafer-to-wafer (W2W) bonding and monolithic 3D (M3D)) have been proposed to address the capacity bottleneck in vertical interconnection. Compared to traditional through-silicon vias (TSVs) technologies, emerging technologies introduce new features for better performance, yield, and cost [1]. However, three newly introduced features have an important impact on 3D placer:

- Heterogeneous Processes: Emerging technologies allow different tiers to use different processes.
- Higher Vertical Connection Capacity: Emerging technologies use smaller "VIAs" for vertical connections, which increase the capacity of vertical connections.
- New Net Model: Pins in the same net are connected together by "VIA" in each tier (Shown in Fig. 1).

The presence of heterogeneous processes poses a challenge for traditional analytical methods, such as those proposed in [2]–[4]. This is mainly due to the assumption that the density model is continuous in three dimensions,

*Corresponding author.

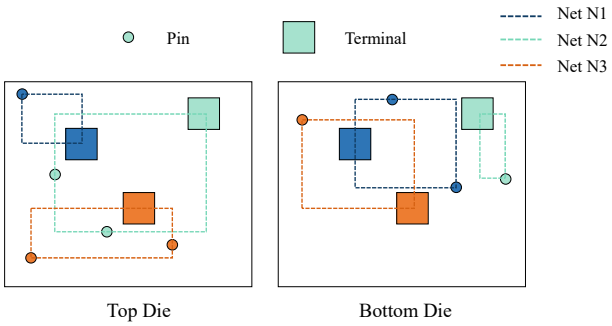

Fig. 1: The wirelength of a net is equal to the sum of the wirelength of the top and bottom parts.

which does not hold in the presence of heterogeneity. Meanwhile, higher vertical connection capacity and a new net model have made the bin-based partitioning method challenging to obtain high-quality solutions, as exemplified in works such as [5]–[7]. This method initially uses a 2D placer (e.g., [8]–[10])to obtain a projected solution. Next, the layout is divided into multiple regions, and the cells within each region are partitioned into two tiers using an area-balanced min-cut algorithm. However, this method does not have a comprehensive objective, and it cannot take full advantage of vertical interconnection. To address this limitation and create a more comprehensive objective, Lu et. al [11] utilize Graph Neural Networks (GNNs) for unsupervised graph representation learning. Upon completion of graph learning, area-balanced partitioning based on the learned representation is used. While this work is novel, it can be challenging to understand its workings from an optimization perspective.

Besides, most previous works treated the tier assignment and placement problem as two independent problems for flexibility. However, this approach limits the solution space, as the solutions obtained when considering each problem independently are merely subsets of the overall solution space. We believe that the interplay between the two phases can be used to observe the overall solution space better. Specifically, tier assignment can provide an initial solution for placement while placement can provide an estimate of the comprehensive objective for tier assignment. Therefore, in this paper, We propose our D2D placer, which can address new features while observing the overall solution space. Our main contributions are summarized as follows:

- We propose a novel modeling approach for the D2D placement problem utilizing bilevel programming. By leveraging the natural dominance relationship between decision variables, our approach provides a comprehensive observation of the overall solution space.
- We propose an efficient partitioning algorithm for the D2D placement problem, which leverages the locality of the placement problem while optimizing the wirelength and #terminal from both global and detailed perspectives.
- We propose an effective and efficient MIV legalization method, which is near optimal in wirelength.
- We propose an alternating optimization framework for bilevel programming and integrate all the above techniques to solve the D2D placement problem. Compared with the top three in ICCAD'22 contest [1], our placer achieves 4.33%, 4.42%, and 5.88% smaller wirelength, 79.61%, 16.74%, 15.76% less #Terminal, 1.84x, 2.32x, 0.68x speedup in runtime. Moreover, even without the alternating iteration, our algorithm achieves a 3% wirelength improvement and is 3.8x, 4.8x, and 1.4x faster than the top three, respectively.

The remainder of this paper is organized as follows. Section II describes the preliminaries; Section III models the D2D placement problem into a bilevel programming problem; Section IV explains the detail of our framework; Section V demonstrates the results; Section VI concludes the paper.

II. Preliminaries

A. Problem Statement

This paper considers the 3D placement problem, including multi-die netlist partitioning and placement with D2D vertical connections presented in the ICCAD'22 CAD Contest [1]. The objective is to assign cells to appropriate dies, creating "connector" (e.g., hybrid bonding terminals) that connects the top and bottom dies of the crossing net, and minimizing the total half-perimeter wirelength (HPWL) shown in Fig. 1. And the constraints that should be satisfied are listed below:

- Technology constraint: The top and bottom dies may use the same or different technologies with the same logical library. Different technologies are associated with different cell sizes, heights, and pin locations. In addition, all cells are single-row height.
- Maximum utilization constraint: Each die has the same area A but a different maximum utilization. The maximum utilization of the top (bottom) die is denoted by u_t (u_b).
- Terminal constraint: If cells in the same net are assigned to different dies, we should add one hybrid bonding terminal. All terminals have the same size and must satisfy the spacing constraint C. The maximum number of terminals is N_t.
- Legalization constraint: All cells need to be placed on the row and do not overlap.

B. Bilevel Programming

Bilevel programming problem consists of two levels of optimization tasks where one optimization task is nested within the other [12]. The upper and lower level problems are referred to as the leader and follower problems, respectively. Each level has its own objective, constraints, and decision variables. The lower-level optimization problem is a constraint to the upper-level optimization problem, so only those members considered feasible are lower-level optimal and satisfy the upper-level constraints. Above all, the general definition of bilevel programming is shown in Definition 1 and an example of the optimal value function is shown in Fig. 2.

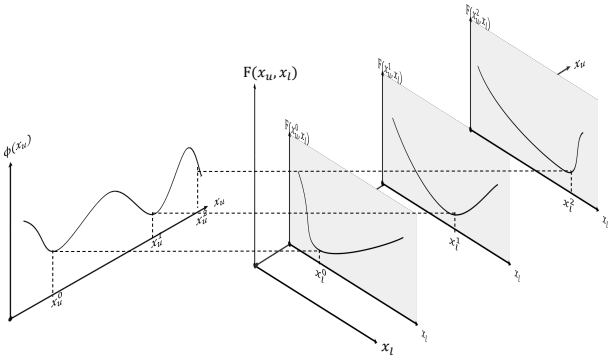

Fig. 2: Graphical representation for bilevel programming optimal value function $\phi(\cdot)$ when the lower subproblem was solved.

Definition 1 (Bilevel Programming)**.** *For the upper-level objective function $F : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ and lower-level objective function $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, the bilevel programming problem is given by*

$$\begin{aligned} \min_{x_u \in X_U, x_l \in X_L} \quad & F(x_u, x_l) \\ s.t. \quad & x_l \in \underset{x_l \in X_L}{\arg\min}\{f(x_u, x_l) | \\ & \qquad g_j(x_u, x_l) \le 0, j = 1, ..., J\} \\ & G_k(x_u, x_l) \le 0, k = 1, ..., K, \end{aligned}$$

where $G_k : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}, k = 1, ..., K$ denote the upper-level constraints, and $g_j : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ represent the lower-level constraints, respectively. Equality constraints may also exist that have been avoided for brevity. In the definition, the sets $X_U \subset \mathbb{R}^n$ and $X_L \subset \mathbb{R}^m$ may denote additional restrictions like integrality. It is common to assume these to be sets of reals unless mentioned otherwise.

III. Modeling for Die-to-Die Placement

A. Original Die-to-Die Placement

We introduce some notations to formulate the D2D placement problem more clearly. $H = (V, E)$ denotes the cells and nets among cells, where the cells and nets are denoted by a set of vertices $V = \{c_1, c_2, ..., c_n\}$ and a set of hyperedges $E = \{e_1, e_2, ..., e_m\}$, respectively. The coordinate of the cell c_i is denoted by (x_i, y_i, z_i), where $z_i \in \{0, 1\}$. And $e_j \in E$ is crossing net if and only if it has both top and bottom cells. We note the bottom part as e_j^- and the top as e_j^+. In addition, $T = \{t_1, t_2, ..., t_m\}$ represents the set of terminals used by crossing nets. (x_{t_j}, y_{t_j}) denotes the coordinate of terminal t_j. We use $\mathrm{WL}_t(e_j, \cdot)$ and $\mathrm{WL}(e_j, \cdot)$ to represent the wirelegnth of

net e_j in the 3D and 2D, respectively. Their relationship is defined as Eq. (1), and $\mathbf{x}_{e_j} = (\mathbf{x}, \mathbf{y}, x_{t_j}, y_{t_j})$.

$$\mathrm{WL}_t(e_j; \mathbf{x}, \mathbf{y}, \mathbf{z}, x_{t_j}, y_{t_j}) =$$
$$\begin{cases} \mathrm{WL}(e_j^- \cup \{t_j\}; \mathbf{x}_{e_j}) + \mathrm{WL}(e_j^+ \cup \{t_j\}; \mathbf{x}_{e_j}) & \varepsilon(e_j; \mathbf{z}) = 1; \\ \mathrm{WL}(e_j; \mathbf{x}, \mathbf{y}) & \varepsilon(e_j; \mathbf{z}) = 0, \end{cases} \tag{1}$$

where $\varepsilon(e; \mathbf{z})$ is defined as Eq. (2), and $\mathbb{I}(\cdot)$ is indicator function.

$$\varepsilon(e; \mathbf{z}) = \max_{c_i \in e}(z_i) - \min_{c_i \in e}(z_i). \tag{2}$$

Therefore, the original D2D placement problem can be formalized as the optimization problem shown in Eq. (3), where $D_b(\cdot)$ represents the density of bin b and M_b is the threshold density of bin b. S_b is the collection of all bins in the top, bottom, and terminal tiers. $A_1(c_i)$ and $A_0(c_i)$ represent the top and bottom area of cell c_i. Additionally, ρ is the cost of a terminal.

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{x}_t,\mathbf{y}_t} \quad & \sum_{e_j \in E} \mathrm{WL}_t(e_j; \mathbf{x}, \mathbf{y}, \mathbf{z}, x_{t_j}, y_{t_j}) + \rho\varepsilon(e; \mathbf{z}), \\ s.t. \quad & D_b(\mathbf{x}, \mathbf{y}, \mathbf{x}_t, \mathbf{y}_t, \mathbf{z}) \le M_b, \ \forall b \in S_b, \\ & \textstyle\sum_{i=1}^{n} A_1(c_i)\mathbb{I}(z_i) \le u_t\mathrm{A}, \\ & \textstyle\sum_{i=1}^{n} A_0(c_i)\mathbb{I}(1 - z_i) \le u_b\mathrm{A}, \\ & \textstyle\sum_{e_j \in E} \varepsilon(e_j; \mathbf{z}) \le N_t. \end{aligned} \tag{3}$$

B. Bilevel Programming for 3D-Placement

It is challenging to solve the original problem directly because of its nonlinear objective and integer variables. The existing partition-based algorithms are easy to solve but lose the overall view of the solution space. Therefore, a modeling way that is easy to solve and retains the overall view of the solution space is necessary. By analyzing the original optimization problem, $\mathrm{WL}_t(\cdot)$ is the key to why it is difficult to calculate analytically. Fortunately, once the vertical coordinates $\mathbf{z}$ are determined, the function becomes computable. Then the rest subproblem is similar to the traditional placement. Therefore, there is a natural dominance relationship between decision variables, and the relationship corresponds to the definition of bilevel programming. The lower level variable can be defined as $\mathbf{x}_l = (\mathbf{x}, \mathbf{y}, \mathbf{x}_t, \mathbf{y}_t)$, and the upper is $\mathbf{z}$. Then, the objective function can be rewritten as:

$$F(\mathbf{x}_l, \mathbf{z}) = \sum_{e_j \in E} \mathrm{WL}_t(e_j; \mathbf{x}_l, \mathbf{z}) + \rho\varepsilon(e; \mathbf{z}). \tag{4}$$

The lower level optimization problem can be defined as Eq. (5a).

$$\Psi(\mathbf{z}) = \arg\min_{\mathbf{x}_l}\{F(\mathbf{x}_l, \mathbf{z}) |\ D_b(\mathbf{x}_l, \mathbf{z}) \le M_b, \forall b \in S_b\} \tag{5a}$$

$$g(\mathbf{z}) = \min_{\mathbf{x}_l}\{F(\mathbf{x}_l, \mathbf{z}) |\ D_b(\mathbf{x}_l, \mathbf{z}) \le M_b, \forall b \in S_b\} \tag{5b}$$

Furthermore, there is a tautology that relates the two equations in Eq. (5b), that is $\forall \mathbf{x}_l \in \Psi(\mathbf{z}), F(\mathbf{x}_l, \mathbf{z}) = g(\mathbf{z})$. Using this proposition, we can transform the original form of bilevel programming into Eq. (6).

$$\begin{aligned} \min_{\mathbf{z},\mathbf{x}_l} \quad & g(\mathbf{z}) \\ s.t. \quad & \mathbf{x}_l \in \Psi(\mathbf{z}) \\ & \textstyle\sum_{i=1}^{n} A_1(c_i)\mathbb{I}(z_i) \le u_t\mathrm{A} \\ & \textstyle\sum_{i=1}^{n} A_0(c_i)\mathbb{I}(1 - z_i) \le u_b\mathrm{A} \\ & \textstyle\sum_{e_j \in E} \varepsilon(e_j; \mathbf{z}) \le N_t \end{aligned} \tag{6}$$

Evaluating $g(\mathbf{z})$ is extremely time-consuming. Therefore, we construct a surrogate function $\hat{g}(\mathbf{x}_l^k, \mathbf{z})$ for evaluating efficiently. The surrogate function takes two arguments. The first one is $\mathbf{x}_l$, which is a good enough solution for $g(\mathbf{z})$ in our assumption. The second is vertical coordinates $\mathbf{z}$. In addition, as shown in Eq. (6), the constraints and objective function are splittable because $\mathbf{x}_l$ has no effect on both the objective and other constraints. Therefore, we split the original problem into two Subproblems 1, 2 and replace the first subproblem's objective function with $\hat{g}(\mathbf{x}_l^k, \mathbf{z})$. Finally, we propose an alternate optimization Algorithm 1, which solves the bilevel programming by solving the two subproblems alternately.

Subproblem 1.

$$\begin{aligned} \min_{\mathbf{z}} \quad & \hat{g}(\mathbf{x}_l^k, \mathbf{z}) \\ s.t. \quad & \textstyle\sum_{i=1}^{n} A_1(c_i)\mathbb{I}(z_i) \le u_t A \\ & \textstyle\sum_{i=1}^{n} A_0(c_i)\mathbb{I}(1 - z_i) \le u_b A \\ & \textstyle\sum_{e_j \in E} \varepsilon(e_j; \mathbf{z}) \le N_t \end{aligned} \tag{7}$$

Subproblem 2.

$$\mathbf{x}_l^{k+1} = \underset{\Psi(\mathbf{z}^{k+1})}{Proj}\ (\mathbf{x}_l^k) \tag{8}$$

Algorithm 1 Alternate Optimization Algorithm

Input: Max iteration M;
Output: Result $\mathbf{x}_l$,$\mathbf{z}$;
1: $k \leftarrow 0$
2: $\mathbf{x}_l^0 \leftarrow$ Initial solution
3: **while** $k \le M$ **do**
4: $\quad \mathbf{z}^{k+1} \leftarrow$ solution of Eq. (7)
5: $\quad \mathbf{x}_l^{k+1} \leftarrow Proj_{\Psi(\mathbf{z}^{k+1})}(\mathbf{x}_l^k)$
6: $\quad k \leftarrow k + 1$
7: **end while**
8: **return** $\mathbf{x}_l^k, \mathbf{z}^k$

In Algorithm 1, the initial solution mentioned in line 2 corresponds to the flattened placement introduced by Section IV-A. Subproblem 1 is solved in line 4, and the key that affects the quality is the setting of the surrogate function $\hat{g}(\mathbf{x}_l, \mathbf{z})$. In this paper, we propose two kinds of surrogate functions. Using these surrogate functions, two tier optimization operators were proposed to optimize Subproblem 1 in global and detailed view, respectively. Subproblem 2 is defined as a projection operator, which corresponds to our planar solution correcting. The basic idea is to find a high-quality solution as close as possible to the existing planar solution.

IV. Our Framework

Fig. 3 shows the overall flow of our proposed placer, in which SP-1 and SP-2 correspond to the Subproblems 1

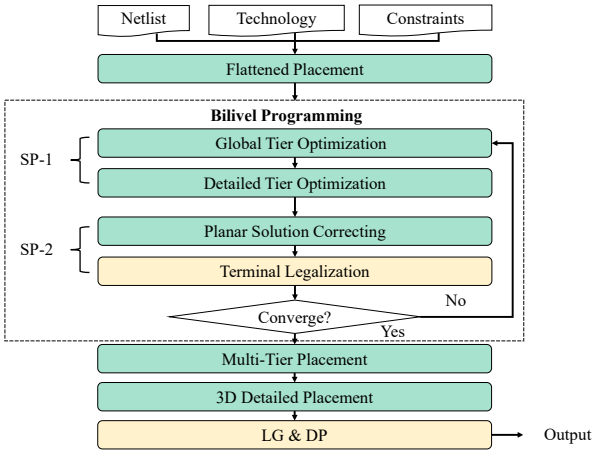

Fig. 3: Our bilevel programming placement framework.

and 2. In this section, we will explain the details of our framework.

A. Flattened Placement for Initial Solution

To obtain a surrogate function $\hat{g}(\mathbf{x}_l, \mathbf{z})$ that is as close as possible to the original function $g(\mathbf{z})$, the quality of $\mathbf{x}_l$ solution is critical. Obviously, there is a conclusion shown in Theorem 1. This theorem demonstrates the possibility that the quality of the solution obtained from the flattened global placement is good enough. Therefore, our framework uses the flattened global placement solution as the initial solution. Particularly, we perform flattened global placement according to the information of the given technology libraries.

Theorem 1. If technology libraries in the top and bottom die are the same, the global placement solution obtained by assigning all cells into one die and doubling the threshold of bin density is a lower bound of the die-to-die placement problem, and the optimal value is between this lower bound and the value of the final legal solution.

B. Global Tiers Optimization

1) Fast Terminal Legalization: In Subproblem 1, if the vertical coordinates of the standard cells are modified, not only the number of terminals will change, but also the extra wirelength introduced by terminals will be affected. Therefore, it is necessary to quickly evaluate the changes of extra wirelength while modifying the vertical coordinates of cells. For the determination of terminal coordinates, we first introduce the Definition 2. This definition inspired us to limit the maximum distance between the legal location of the terminal and the optimal region. In this way, the extra wirelength can be within a certain range, and there is an opportunity to do incremental terminal legalization (using BFS in our method).

Definition 2 (Terminal Optimal Region)**.** *The optimal region of terminal t_j is defined as the position such that the extra wirelength introduced by terminals is minimized. It can be formulated as Eq. (9).*

$$\min_{x_{t_j}, y_{t_j}} \mathit{HPWL}(e_j^- \cup \{t_j\}; \mathbf{x}_{e_j}) + \mathit{HPWL}(e_j^+ \cup \{t_j\}; \mathbf{x}_{e_j}) \tag{9}$$

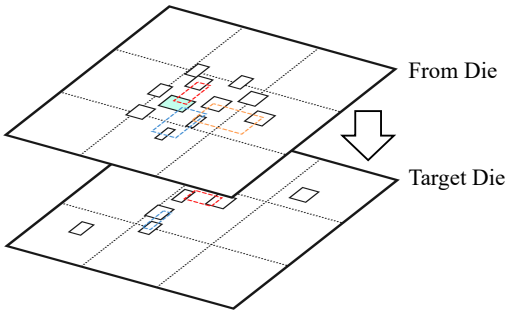

Fig. 4: Example for tier optimization.

This form is the same as the optimal region proposed by [13], that is, the bounding boxes' midpoint of the net e_j^- and net e_j^+ is the optimal solution.

2) Best Improvement for Tier Optimization: Solving Subproblem 1 is still difficult for analytical methods, while it is acceptable for classical iterative improvement methods because most parts can be evaluated incrementally (e.g., terminals' legal locations). For the iterative improvement algorithm, it is still necessary to restrict the state space for efficiency. Above all, we transform Subproblem 1 into another Subproblem 3.

Subproblem 3 (Tier Optimization)**.** *There are cell set S_f in the from die and S_t in the target die, where $f, t \in \{0, 1\}$ and $f \neq t$. For a cell set $S \subseteq S_f$, we say S is feasible if and only if $A_t(S_t \cup S) \leq B$, where B is a constant. The knapsack constraint can be written as $\mathcal{I} = \{S | A_t(S) \leq B - A_t(S_t)\}$. The goal is to find the ordered cell set $S = \{c_1, c_2, ..., c_j\} \subseteq \mathcal{I}$ whose vertical coordinates need to be modified to minimize surrogate function $\hat{g}$.*

Definition 3 (Neighborhood in Tier Optimization)**.** *We use $\mathcal{N}(S)$ to represent the cell set with a neighborhood relationship to S. If a cell set $S' \in \mathcal{N}(S)$, there is a cell $c \in S_f \backslash S$ and $c = S' \backslash S$.*

Definition 4 (Accept Function in Tier Optimization)**.** *We define the* accept function *as $ac(S, c) : \mathcal{I} \times S_f \rightarrow \{0, 1\}$, where returns 1 if and only if the terminals newly introduced have available locations after changing vertical coordinate of cell c and $S \cup \{c\} \subseteq \mathcal{I}$.*

As described in Subproblem 3, we relax the two linear constraints in the original Subproblem 1 into a knapsack constraint by restricting the direction of moving as shown in Fig. 4, where the size of a region is defined as τ times the bin size. Besides, the subproblem is also defined as a search problem. We use Definitions 3 and 4 to describe its neighborhood and accept function, respectively. To solve this subproblem, we use an algorithm shown in Algorithm 2.

The best improvement is an important idea used in the traditional partition algorithm [14], [15]. However, the critical heuristics in the algorithm are strongly related to the specific problem definition. Two ideas are essential in our algorithm. On the one hand, we maintain a priority queue Q shown in line 1. The priority function $p(S)$ is defined as a set function, which is used to evaluate the benefit when the cells contained in the set S are

Algorithm 2 Global Tier Optimization

Input: f, t, S_f, S_t, B;
Output: S;

Maintain a priority queue Q where elements in S_f are sorted in increasing order by their keys, and the key of c_i is initialized to $\frac{p(S\cup\{c_i\})-p(S)}{A_t(c_i)}$.
$S \leftarrow \emptyset$, $c_i, b_i \leftarrow Q.front()$
while $ac(S, c_i)$ **and** $b_i \geq 0$ **and** $A_t(S_t \cup S) \leq B$ **do**
    $S_f \leftarrow S_f \backslash \{c_i\}$, $S \leftarrow S \cup \{c_i\}$
    FastTerminalLegalization(c_i,f).
    Update the element's priority by region and nets.
    $c_i, b_i \leftarrow Q.front()$
end while
return S

moved from the bottom (top) die to the top (bottom) die. In detail, $p(S) = \{\hat{g}(\mathbf{x}_l^S, \mathbf{z}^S) | \forall c_i \in S_t \cup S, z_i = t \wedge \forall c_j \in S_f \backslash S, z_j = f \wedge (x_{t_k}, y_{t_k}) \in T(S)\}$, where $T(S)$ is the terminal coordinates set determined in the process of obtaining S. In particular, this priority represents the benefit per unit area, which is a vital technique for solving the maximization problem with the knapsack constraint. On the other hand, when the vertical coordinate of a cell is changed, only the cells in the same net or the same region need to be reevaluated. This is the key to reducing the times of evaluations and it can be parallelized. As for the runtime, the largest part is the updating of the queue Q, which is quite fast after parallelization.

$$\begin{aligned} p(S \cup \{c_i\}) - p(S) = \Delta\text{wirelenth} + \rho\Delta\text{\#Terminal} \\ + \alpha\big(d(S \cup \{c_i\}) - d(S)\big) \\ + \beta\big(o(S \cup \{c_i\}) - o(S)\big) \\ - \gamma d(S), \end{aligned} \tag{10a}$$

where

$$\begin{aligned} d(S) &= \sum_{\text{region } r} \max(\frac{A_r - M_r}{h_r}, 0), \\ o(S) &= \sum_{i=1}^{n} \sum_{c_j \in \{c_j | \forall c_j \in V, z_j = z_i\}} \frac{\text{Overlap}(c_j, c_i)}{h_{c_i}} \end{aligned} \tag{10b}$$

3) Hyperparameterized Surrogate Functions: The priority calculation shown in Algorithm 2 is the most critical part of the algorithm because it needs to balance the objective and constraints. This priority is defined as the difference of the objective function, which implies the idea of the discrete gradient. We use the formula shown as Eq. (10a) to give a general difference form of the priority function $p(S \cup \{c_i\}) - p(S)$, where α, β, and γ are hyperparameters, h_r, A_r, M_r represent region r's row height, the total area of region cells, maximum available area, respectively. h_{c_i} is the row height of the die that cell c_i belonging.

In this formula, we analyze several factors that may have an impact on the results, including (a) the total wirelength and #terminal. Particularly, the total wirelength includes the extra wirelength introduced by terminals. (b) the overflow of regions, and (c) the overlap between cells. Furthermore, this priority function reveals the principle that when γ is large enough, the region with the largest overflow is selected, then the region interior is sorted by the rest of the weight.

4) Other Techniques for Global Tier Optimization: To capture the global view as much as possible, some other techniques have been applied to global tier optimization:

- Additional Density Penalty: As mentioned in [16], the density weight has an essential impact on the solution. We use additional density penalties to cope with high-density scenarios.
- Variable Neighborhood Search: Variable neighborhood search (VNS) is a common technique in local search to avoid falling into local optimal prematurely. In our approach, we alternate neighborhoods several times in the hope of obtaining a better solution.

C. Detailed Tier Optimization

1) Dynamic Row-based Data Structure: Similar to the detailed placement in the traditional placement, further optimization is still necessary after global Tier optimization in our framework. However, the coarse-grained surrogate function (10a) is difficult to accurately represent the exact objective function, which may do harmful optimizations. Therefore, we propose a new data structure to help calculate the exact objective function (wirelength and #terminal). This data structure is inspired by the classical legalization algorithm Abacus [17]. However, the data structure in Abacus only allows cells to be inserted sequentially at the end of rows. We extended this mechanism to allow cells to be inserted anywhere in the row.

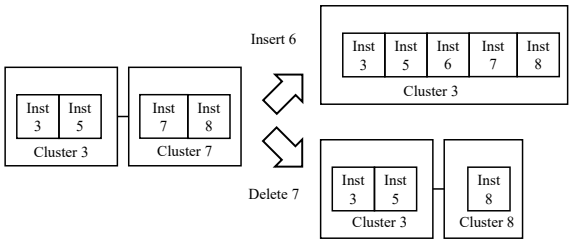

Fig. 5: An example for dynamic row-based datastructure.

Specifically, we maintain a partial order for all clusters in the same row according to their priority, which is defined as the horizontal center coordinate of the first standard cell in the cluster. Fig. 5 shows an example of inserting or deleting cells when using a dynamic row-based data structure.

2) First Improvement for Tier Optimization: Next, we will introduce our detailed tier optimization algorithm. It also restricts the direction of moving and is implemented as a first improvement algorithm. Specifically, we first sort all the cells according to the priority mentioned in Section IV-B2. Then, only top δ cells are evaluated and considered for changing vertical coordinates.

Therefore, evaluating the benefit of modifying cells' vertical coordinates is the most critical part of the detailed tier optimization algorithm. The specific operation is implemented as follows: a) Select an appropriate row in the target die for the cell c_i that needs to be modified vertical coordinate. Two modes serve different purposes:

FIX and MOVE. When the FIX mode is used, a row is selected according to the original horizontal coordinate of c_i. In contrast, when MOVE mode is used, the optimal region of cell c_i on the target die is computed first, and a row is selected using the center coordinate of the optimal region. b) Insert cell c_i into the selected row and quickly legalize it by using row-based data structures. c) Collect all cells affected by the collapse operation and calculate the benefits.

It is important to note that this process does not change the horizontal position of all cells but only maintains a new legal solution for calculating the fine-grained benefit. As for the runtime, it is necessary to limit the number of invoking.

D. Planar Solution Correcting

After tier optimization, the overflow in each region is mostly eliminated, and the approximate locations of the terminals are determined. The region size is still larger than the bin size in plane placement, which may lead to an inaccurate estimation of the surrogate function. Therefore, to make the surrogate function as accurate as possible, we need to correct the solution. For performance and stability reasons, we alternately fix the coordinates of cells on one die and place cells on the other die using a 2D placement algorithm [9].

E. Terminal Legalization

After placing all cells, the locations of terminals need to be determined. The terminal legalization problem can be formulated as an optimization problem shown in Eq. (11).

$$\begin{aligned} \min_{\mathbf{x}_t,\mathbf{y}_t} \quad & \sum_{j=1}^{m} \mathrm{WL}(e_j^- \cup \{t_j\}; \mathbf{x}_{e_j}) + \mathrm{WL}(e_j^+ \cup \{t_j\}; \mathbf{x}_{e_j}) \\ \text{s.t.} \quad & \min(|x_{t_i} - x_{t_j}|, |y_{t_i} - y_{t_j}|) \geq C, \\ & \forall i, j = 1, 2, ..., m. \end{aligned} \tag{11}$$

Especially, there is no mutual influence in the objective function between the different terminals. In other words, the objective function can be computed separately. Besides, all terminals are the same size. Thanks to both of these factors, if we restrict the coordinates of all terminals to integer multiples of the spacing constraint C, we can obtain an optimal solution using weighted bipartite graph matching. Specifically, we divide the layout into uniform square grids with length C. Then, we select k candidate grids around its terminal optimal region (TOR) for each terminal. The cost of a candidate grid is the Manhattan distance to its TOR. Therefore, one part of the bipartite graph is the candidate grids and the other part is defined as terminals. The example in Fig. 6 illustrates the bipartite graph we have described.

After obtaining a grid solution, we lift the restriction of the grid and perform a refinement. Specifically, we first sort the terminals by distance to their TOR. Then, each terminal moves in four directions and uses binary search to calculate the maximum distance it can move. We use the r-tree data structure to quickly determine the legality of the new location.

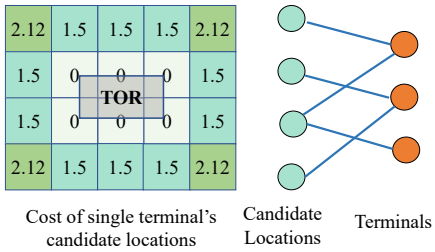

Fig. 6: Bipartite graph and cost example for terminal legalization.

To further analyze the quality of the algorithm we introduce Theorem 2. This theorem gives a lower bound for the solution obtained by restricting the coordinates. In practice, the wirelength difference between our solution and the optimal solution is usually less than 0.5%.

Theorem 2. The difference between the optimal value of Eq. (11) and the grid optimal value mentioned above is less than twice the product of #terminal and C.

Proof. We denote the optimal solution of Eq. (11) as $(\mathbf{x_t^*}, \mathbf{y_t^*})$. Then, we sort all the terminals in ascending abscissa order, giving $x_{t_i}^* + C \leq x_{t_j}^*$, iff $i < j \wedge |y_{t_i}^* - y_{t_j}^*| < C$. To obtain a feasible grid solution, we use $x'_{t_i} = \lfloor x_{t_i}/C \rfloor \times C$ to denote the new coordinates for each terminal. Obviously, $x'_{t_i} + C \leq x'_{t_j}$ still holds in the previous condition. Similarly, the same is true if $x'_{t_i} = \lceil x_{t_i}/C \rceil \times C$. Therefore, We can establish the relationship between feasible grid solution and optimal solution, as described in Eq. (12). Further, the objective of Eq. (11) is the same as the problem that finding the optimal region for two nets. The absolute value of the slope of this objective function is less than 2. Therefore, the difference in the x direction is less than mC and the total is less than $2mC$.

$$\begin{aligned} \sum_{j=1}^{m} |x'_{t_j} - x^*_{t_j}| = \min(&\sum_{j=1}^{m} x^*_{t_j} \bmod C, \\ &\sum_{j=1}^{m} (C - x^*_{t_j} \bmod C)) \leq \frac{mC}{2} \end{aligned} \tag{12}$$

□

Next, we try to analyze the complexity. The bipartite graph includes $(k+1)m$ vertices and km edges at most. To solve the weighted bipartite graph matching problem, we transform it into a minimum cost maximum flow problem. Compared with the traditional high complexity dinic algorithm, we use the network simplex (NS) algorithm as the solver. The NS algorithm is a graph theoretic specialization of the simplex algorithm. We use the implementation in Lemmon [18]. It typically expects $O(VE)$, but at worst it can be worse. In practice, it's ultra fast.

F. Multi-Tier Placement

After determining all cells' vertical coordinates, we performed refinement on the horizontal coordinates of

TABLE I
EXPERIMENTAL RESULTS ON ICCAD 2022 CONTEST BENCHMARKS.

Case	Flattened	3th			2nd			1st			Ours		
	GP	HPWL	#Terminal	CPU(s)	HPWL	#Terminal	CPU(s)	HPWL	#Terminal	CPU(s)	HPWL	#Terminal	CPU(s)†
case2	1758214	2097487	**163**	10	2080647	477	14	2072075	1131	45	**1992499**	461	45
case2_hidden	2111322	2644791	**151**	9	2735158	687	15	2555461	1083	40	**2530195**	658	53
case3	26474613	33063568	14788	145	30969011	11257	437	30580336	16820	635	**30234112**	**9612**	442
case3_hidden	24200040	28372567	11211	133	27756492	8953	482	27650329	16414	412	**26939286**	**8203**	479
case4	248129463	281378079	46468	925	274026687	51480	3284	281315669	84069	2580	**267381744**	**43140**	1078
case4_hidden	272085522	307399565	58860	983	308359159	59896	3283	301193374	84728	2239	**289541474**	**51641**	1144
N.Total	-7.63%	5.88%	15.76%	0.68	4.42%	16.74%	2.32	4.33%	79.61%	1.84	**0.00%**	**0.00%**	1.00

† The result of the top 3 winners is provided by the contest organizer with Intel(R) Xeon(R) CPU E7-4820@2.00 GHz, 128GB memory, and 8 threads. For fair comparisons, our runtime is scaled to reflect machine difference: CPU = runtime in our machine $\times \frac{\text{TOP-3's reported runtime}}{\text{TOP-3's runtime in our machine}}$.

components (including cells and terminals). Specifically, we extend the method in [19] to adapt to our problem. Specifically, we use the multi-field model to handle density constraints in different tiers separately.

G. Legalization and Detailed Placement

After all the cells' vertical and horizontal coordinates have been determined, we first call the existing 2D legalizer and detailed placer [10]. Then we use the row-based data structure mentioned in Section IV-B4 to complete the 3D detailed placement. This algorithm can realize the transformation from one legal solution to another legal solution. Therefore, it can be used at the end of the framework to further optimize the objective.

V. EXPERIMENTAL RESULTS

In this section, we describe a set of comprehensive experiments on the benchmarks provided by the ICCAD'22 CAD Contest [1]. Summary statistics of the benchmarks are presented in Table II, in which the problem sizes range from 2k to 220k. The number of cells, nets, bottom(top) maximum utilization, and the maximum number of terminals are denoted by "#Cells", "#Nets", "$u_b(u_t)$", and "#Max_terminals", respectively. In addition, "Diff_tech" indicates whether the top and bottom die used the same technology.

TABLE II
BENCHMARK STATISTICS OF ICCAD'22 CAD CONTEST.

Case	#Cells	#Nets	#Pins	#Max_terminals	u_t(%)	u_b(%)	Diff_tech
case2	2,735	2,644	8,118	2,000	70	75	Yes
case2_hidden	2,735	2,644	8,118	2,000	79	79	No
case3	44,764	44,360	142,246	36,481	78	78	No
case3_hidden	44,764	44,360	142,246	36,100	68	78	Yes
case4	220,845	220,071	773,551	183,612	66	70	Yes
case4_hidden	220,845	220,071	773,551	178,929	66	76	Yes

A. Experimental Setup

Our placer is implemented in C++ and all experiments were performed on the same platform with a 64-bit Linux machine using Intel Xeon Platinum 8380 CPU 2.30GHz and 960 GB memory. In addition, we used 8 threads for a fair comparison with the top 3 winners in ICCAD'22 CAD Contest [1]. In addition, some necessary parameters for our placer are also given in Table. III.

TABLE III
PARAMETERS OF OUR PLACER.

Parameter	**Description** (default setting)
ρ	Terminal cost. ($\rho = 500$)
α	Overflow in region cost. ($\alpha = 100$)
β	Overlap cost. ($\beta = 0.5$)
γ	Additional overflow penalty. ($\gamma = 0$ or 10000)
τ	Times of a region over a bin. ($\tau = 7$)
δ	Candidate locations for a terminal. ($\delta = 25$)
B	Maximum moving area. ($B = 1.1u_tA$)
M	Alternate optimization iteration. ($M = 4$)

B. Validation on Contest Benchmarks

The proposed method was applied to benchmarks from the ICCAD'22 CAD contest suite [1], and the cumulative HPWL post-detailed placement was measured. In the original contest, the evaluation score was calculated by summing the total HPWL of the top and bottom dies. However, the number of terminals is also an important consideration in the actual D2D placement problem. Therefore, we further use HPWL, the number of terminals, and runtime to evaluate the effectiveness of our method. Table I presents the comparison results between our algorithm and the top three winners. In the table, "HPWL" represents the total wirelength, "#Terminal" represents the number of terminals, "CPU" represents the runtime in seconds, and "N.Total" represents the normalized improvement of the corresponding metric. The best result for each case is highlighted in bold.

As shown in Table I, our algorithm has achieved the best results in all released benchmarks. Compared to the top 3 winners, our algorithm demonstrates 4.33%, 4.42%, and 5.88% wirelength improvements; 79.61%, 16.74%, and 15.76% less #Terminal; 1.84x, 2.32x, and 0.68x speedup. To further illustrate the effectiveness of our algorithm, we take the result of Theorem 1 as the lower bound and calculate the gap between our algorithm and the lower bound. Although the flattened global placement still has overlaps and does not consider the effect of terminals on the increasing of wirelength, our placer can obtain only 7.63% on total increasing wirelength than the flattened global placement. This indicates that our method is close to the optimal solution depending on the chosen global placement algorithm.

Further, We use the overall runtime fraction instead of the time complexity analysis to illustrate the efficiency. Fig. 7 illustrates the runtime breakdown of each module of our placer in case4. It can be found that the time

of each component is relatively balanced, indicating the effectiveness of the method.

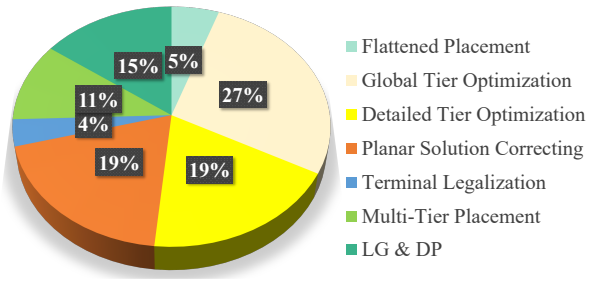

Fig. 7: Speed breakdown in Case4.

C. Ablation Studies

After conducting our experiments, we performed ablation studies to further evaluate the effectiveness of different components in our approach. We investigated the results with and without alternating optimization iteration as shown in Table IV. First, we analyzed the performance without alternating optimization to evaluate how effective our bilevel programming method can understand the overall solution space. The result without alternating optimization achieved a 3% improvement in wirelength over the top three competitors and outperformed all other results in runtime, using only a small number of additional terminals. When using alternating optimization, it intensified the exploration of the solution space. Therefore, The result with alternating optimization shows state-of-the-art in both wirelength and #terminal. Overall, these experimental results fully demonstrate the effectiveness and efficiency of our proposed algorithm.

TABLE IV
RESULTS WITH AND WITHOUT ALTERNATING OPTIMIZATION

Case	w/o. Alternating Optimization.			w/ Alternating Optimization		
	HPWL	#Terminal	CPU(s)	HPWL	#Terminal	CPU(s)
case2	2032655	555	20	1992499	461	45
case2_hidden	2562890	793	19	2530195	658	53
case3	30332531	10604	135	30234112	9612	442
case3_hidden	26935732	9288	128	26939286	8203	479
case4	270042122	54112	604	267381744	43140	1,078
case4_hidden	294923683	63283	637	289541474	51641	1,144
N.Total	1.33%	21.91%	0.48	0.00%	0.00%	1.00

D. Validation For Terminal legalization

We conducted an extra experiment to showcase the effectiveness of our terminal legalization algorithm. The results are presented in Table V. T.Width, #Terminal, and WL correspond to the spacing constraint C, the number of terminals, and the final HPWL, respectively. Additionally, TOR represents the wirelength obtained when placing all terminals in the terminal optimal region, which is not legal and hence can only serve as an upper bound for evaluation. The "Ratio" represents the difference between the final wirelength and this upper bound. As depicted in the table, we observed that this gap is quite small, which suggests that our results are near-optimal.

TABLE V
TERMINAL LEGALIZATION EXPERIMENTAL RESULTS.

Case	C	#Terminal	WL	CPU(s)	TOR	Ratio
case2	200	461	1992499	1	1981785	0.54%
case2_hidden	228	658	2530195	1	2512837	0.69%
case3	100	9612	30234112	7	30141038	0.31%
case3_hidden	92	8203	26939286	5	26875050	0.24%
case4	124	43140	267381744	15	266850007	0.20%
case4_hidden	132	51641	289541474	16	288659033	0.30%

E. Validation on Scalability

In 3D chip design, the size and spacing constraints of the terminals significantly impact overall quality. To further verify the scalability of our algorithm, we tested its performance by varying the terminal pitch. A smaller pitch (0.5x) means that more terminals can be placed in a given area, while a larger pitch (1.5x) allows for fewer terminals in the same area. Fig. 8 (a) shows that our algorithm achieves the best results in all situations. In particular, even for large terminals, the increase in wirelength is minimal, demonstrating the excellent scalability of our method. Fig. 8 (b) displays that our algorithm uses the fewest number of terminals in all cases. In practice, large terminals often have a greater impact on routability, power consumption, and other aspects. Therefore, we further reduce terminal usage when dealing with large terminals. This shows that our method is more practical for real-world problems.

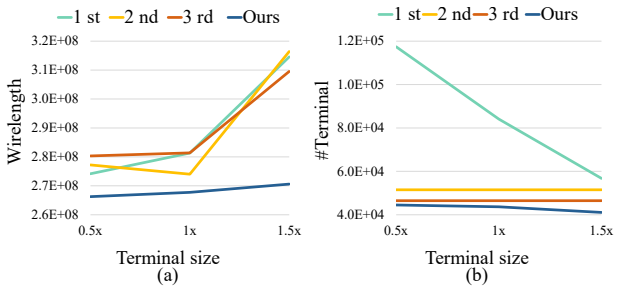

Fig. 8: Extra experiments of different terminal pitch (Case4).

VI. CONCLUSION

This paper proposes a new paradigm for solving the D2D placement problem by transforming the original problem into a bilevel programming problem. By leveraging the understanding of the overall solution space, our placer is able to achieve up to a 4% improvement in wirelength and a 79% reduction in the number of terminals compared to state-of-the-art methods. Our future work shall focus on designing strategies for hyperparameter updating in different alternate iterations and initial solutions for tier optimization.

ACKNOWLEDGMENT

This work is supported in part by the National Key R&D Program of China (No. 2022YFB4500403), the Major Key Project of PCL (No. PCL2023AS2-3), the Strategic Priority Research Program of Chinese Academy of Sciences (No. XDA0320300).

REFERENCES

[1] K.-S. Hu, I.-J. Lin, Y.-H. Huang, H.-Y. Chi, Y.-H. Wu, and C.-F. C. Shen, "2022 ICCAD CAD contest problem B: 3D placement with D2D vertical connections," in *Proceedings of ICCAD*. IEEE, 2022, pp. 1–3.

[2] J. Cong and G. Luo, "A multilevel analytical placement for 3D ICs," in *Proceedings of ASP-DAC*. IEEE, 2009, pp. 361–366.

[3] M.-K. Hsu, Y.-W. Chang, and V. Balabanov, "TSV-aware analytical placement for 3D IC designs," in *Proceedings of DAC*, 2011, pp. 664–669.

[4] J. Lu, H. Zhuang, I. Kang, P. Chen, and C.-K. Cheng, "ePlace-3D: Electrostatics based placement for 3D-ICs," in *Proceedings of ISPD*, 2016, pp. 11–18.

[5] S. Panth, K. Samadi, Y. Du, and S. K. Lim, "Placement-driven partitioning for congestion mitigation in monolithic 3D IC designs," in *Proceedings of ISPD*, 2014, pp. 47–54.

[6] B. W. Ku, K. Chang, and S. K. Lim, "Compact-2D: A physical design methodology to build two-tier gate-level 3-D ICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 6, pp. 1151–1164, 2019.

[7] S. Panth, K. Samadi, Y. Du, and S. K. Lim, "Shrunk-2-D: A physical design methodology to build commercial-quality monolithic 3-D ICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 10, pp. 1716–1724, 2017.

[8] Y. Lin, S. Dhar, W. Li, H. Ren, B. Khailany, and D. Z. Pan, "DREAMPlace: Deep learning toolkit-enabled GPU acceleration for modern VLSI placement," in *Proceedings of DAC*, 2019, pp. 1–6.

[9] J. Lu, P. Chen, C.-C. Chang, L. Sha, D. J.-H. Huang, C.-C. Teng, and C.-K. Cheng, "ePlace: Electrostatics-based placement using fast fourier transform and nesterov's method," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 20, no. 2, pp. 1–34, 2015.

[10] 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.

[11] Y.-C. Lu, S. S. K. Pentapati, L. Zhu, K. Samadi, and S. K. Lim, "TP-GNN: A graph neural network framework for tier partitioning in monolithic 3D ICs," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

[12] A. Sinha, P. Malo, and K. Deb, "A review on bilevel optimization: from classical to evolutionary approaches and applications," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 276–295, 2017.

[13] M. Pan, N. Viswanathan, and C. Chu, "An efficient and effective detailed placement algorithm," in *Proceedings of ICCAD*. IEEE, 2005, pp. 48–55.

[14] C. M. Fiduccia and R. M. Mattheyses, "A linear-time heuristic for improving network partitions," in *Proceedings of DAC*. ACM, 1982, pp. 1–6.

[15] A. E. Caldwell, A. B. Kahng, and I. L. Markov, "Improved algorithms for hypergraph bipartitioning," in *Proceedings of the 2000 Asia and South Pacific Design Automation Conference*, 2000, pp. 661–666.

[16] Z. Zhu, J. Chen, Z. Peng, W. Zhu, and Y.-W. Chang, "Generalized augmented lagrangian and its applications to VLSI global placement," in *Proceedings of DAC*. IEEE, 2018, pp. 1–6.

[17] P. Spindler, U. Schlichtmann, and F. M. Johannes, "Abacus: Fast legalization of standard cell circuits with minimal movement," in *Proceedings of ISPD*, 2008, pp. 47–53.

[18] B. Dezső, A. Jüttner, and P. Kovács, "Lemon–an open source C++ graph template library," *Electronic Notes in Theoretical Computer Science*, vol. 264, no. 5, pp. 23–45, 2011.

[19] J. Gu, Z. Jiang, Y. Lin, and D. Z. Pan, "DREAMPlace 3.0: Multi-electrostatics based robust VLSI placement with region constraints," in *Proceedings of the 39th International Conference on Computer-Aided Design*, 2020, pp. 1–9.