3D-Flow: Flow-based Standard Cell Legalization for 3D ICs

Yuxuan Zhao, Peiyu Liao, Bei Yu
Department of Computer Science and Engineering
The Chinese University of Hong Kong

***Abstract*—The standard-cell placement legalization is a critical step in the physical design. The emerging 3D ICs have brought challenges to traditional legalizers on efficiency and effectiveness. In this work, we present a fast flow-based legalization algorithm, 3D-Flow, that minimizes cell displacement in a 3D solution space. Our legalizer resolves overflowed bins by finding the shortest augmenting path on a 3D grid graph, utilizing an effective branch-and-bound algorithm. Moreover, a post-optimization with a cycle-canceling algorithm is proposed to minimize the maximum displacement. Our approach leverages the global perspective inherent in network flow methods, considering multiple dies in 3D ICs to minimize cell displacement. Experimental results on ICCAD 2022 and 2023 contest benchmarks demonstrate our proposed algorithm achieves up to 13% and 43% less average and maximum cell displacement compared to state-of-the-art legalizers in a similar runtime.**

I. Introduction

Placement legalization is a critical step in the physical synthesis flow. It is typically invoked after global placement and timing optimization, such as gate sizing and buffer insertion, which may compromise the legality of a placement. An ideal legalization algorithm removes all overlap of instances while preserving the quality of the given placement. As technology scaling approaches its physical limits, 3D integrated circuits (3D ICs) [1] have emerged as a viable solution for extending Moore's Law. The vertical stacking of multiple dies in 3D ICs introduces new challenges and optimization opportunities for legalization algorithms.

Existing legalization algorithms can be broadly categorized into two types: *greedy search* and *formal optimization*. Most legalizers [2]–[5] fall into the former category, sequentially moving cells to the nearest free locations. Typically, the sequence of movement is determined by the cells' x-position, arranged in either ascending or descending order. A notable example of this approach is Tetris [2], which has been widely adopted in modern placement tools [6], [7]. Abacus [4] follows a similar procedure as Tetris, but moves the cells already legalized in a row by dynamic programming to minimize the total movement. While the greedy approaches are fast and flexible, they can result in excessive cell displacement, especially in designs with high utilization or blockages caused by macros.

In contrast, formal optimization approaches model placement legalization through mathematical formulations such as network flow [8]–[12] or diffusion [13]. Specifically, the diffusion model [13] employs force-directed methods to globally reposition cells. In flow-based methods, legalization is modeled as a minimum-cost flow problem, where cells are moved from source bins to sink bins, resolving overflow with minimal cell movement. If all cells are of uniform size, the problem can be efficiently solved in polynomial time [8]. However, real-world designs have cells with different sizes, necessitating a flow realization step in these approaches. Cho *et al.* [9] introduce a history-based legalization algorithm that incorporates history data into the cost formulation to avoid unfeasible flows. Inspired by the successive shortest path algorithm [14], BonnPlaceLegal [10] ensures that only flow augmentations, which can be realized exactly by cell movement, are considered. The algorithm iteratively computes the shortest paths from source bins to sink bins using Dijkstra's algorithm. While this approach achieves satisfactory cell displacement results, the iterative

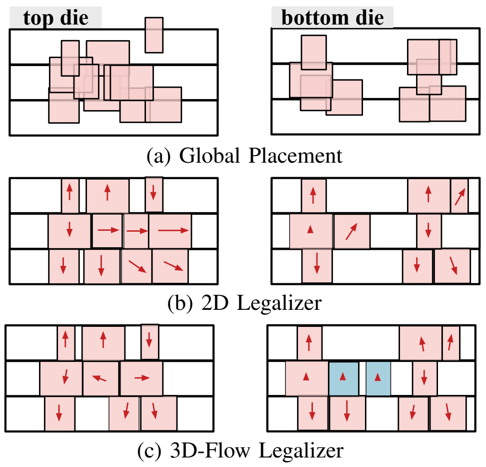


Fig. 1 Motivation for 3D-Flow legalizer. Global placement (a) is given for a 3D IC with two dies. (b) and (c) show legal placements, with red arrows indicating the cell displacement between global and legal placements. By moving two **blue**-colored cells (c) from the top to the bottom die, our 3D-Flow legalizer achieves reduced displacement.

calls to Dijkstra's algorithm make it unscalable for large designs. To improve efficiency, Darav *et al.* [11] propose utilizing the breadth-first search (BFS) algorithm to identify multiple candidate paths. Although this algorithm is robust and efficient, it may result in large total cell displacement as it does not consider the cost when selecting the augmenting paths. Innovations are needed to balance the trade-off between solution quality and efficiency in formal optimization approaches.

In the context of 3D placement, placers need to determine both cell positions and cell partitioning to optimize design objectives. Pseudo-3D placers [15], [16] separate the partitioning and placement phases. They utilize 2D placement tools to determine cell positions while employing either partitioning-first [15] or partitioning-last [16] flows to perform the die assignment. In contrast, true-3D placers [17]–[20] relax discrete die partitioning constraints and adopt analytical approaches to simultaneously optimize cell partitioning and positions in a 3D solution space. Recent studies demonstrate the superiority of true-3D placers, as they explore a larger solution space. However, legalization algorithms are overlooked in existing true-3D placers. The state-of-the-art (SOTA) true-3D placers [18], [19] greedily assign cells to the nearest die after global placement and then legalize the placement with fixed die assignments using traditional 2D legalizers [2], [4]. However, legalizing the placement in a 3D solution space will produce results with higher quality. Fig. 1 shows an example that by moving cells across the dies, the total displacement and maximum displacement can be significantly reduced at the legalization stage for 3D ICs.

In this work, we present a fast flow-based legalization algorithm for 3D ICs. Our approach leverages the global perspective inherent

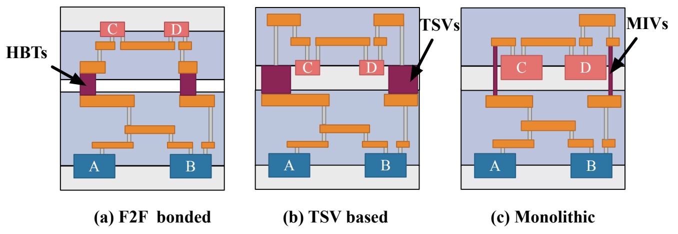


Fig. 2 The typical types of 3D ICs. (a) F2F bonded 3D ICs enable heterogeneous technology integration without space crowding on the silicon layer. (b) TSVs occupy large silicon area, resulting in low integration density. (c) Monolithic 3D ICs require sequential fabrication with low yield.

in network flow methods, considering multiple dies in a 3D solution space to minimize cell movement. Our contributions are summarized as follows.

- We propose the first 3D legalizer, 3D-Flow, to minimize cell displacement in a 3D solution space for 3D ICs using network flow methods.
- We resolve overflowed bins by finding the shortest augmenting path on a 3D grid graph, utilizing an effective branch-and-bound algorithm. A post-optimization with cycle-canceling is proposed to reduce maximum displacement.
- Experimental results on ICCAD 2022 and 2023 contest benchmarks demonstrate 3D-Flow achieves up to 13% and 43% less average and maximum displacement compared to SOTA 2D legalizers in a similar runtime.

II. Preliminaries

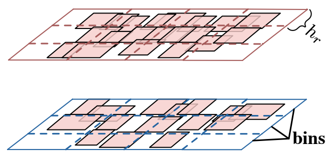
A. 3D ICs

3D integration enables higher transistor density on the silicon and replaces the long 2D interconnects with shorter inter-die connections. There are mainly three types of 3D ICs: through-silicon-via (TSV) based, monolithic, and face-to-face (F2F) bonding. Fig. 2 illustrates the 3D stacking for typical 3D ICs.

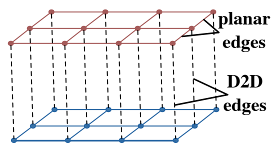
TSVs [21] occupy a significant amount of silicon area and introduce large parasitics, limiting TSV-based 3D ICs to a few inter-die connections. While monolithic inter-tier vias (MIVs) exhibit nanoscale sizes and enable fine-grain 3D interconnects, monolithic 3D ICs [22] require sequential fabrication of multiple dies, resulting in low yield and high cost. F2F bonded 3D ICs [23], [24] consist of two pre-fabricated dies, where the top die is flipped and stacked on top of the bottom die using hybrid bonding terminals on the topmost metal layers. The ease of fabrication and small size of bonding terminals enable a high integration density at a low cost. Additionally, pre-fabricated dies can adopt different technology nodes to achieve cost-effective performance, making F2F bonding a preferred approach. Recent ICCAD contests [25], [26] have addressed 3D placement for F2F bonded 3D ICs with heterogeneous technology integration. Therefore, we evaluate our 3D-Flow legalizer in F2F bonded settings, but our legalization algorithm is sufficiently general to apply to other types of 3D ICs with more than two dies.

B. Flow-based Legalization Formulation

We focus on standard cell legalization after global placement for true-3D placers, where cells are positioned in a continuous 3D space with their die assignments yet to be determined. The macros have been placed on their corresponding dies without any overlap. Specifically, we adopt the F2F bonded stacking specified in the ICCAD 2022 and 2023 contests [25], [26]. The cells must be placed on either the top or bottom die, aligned to placement rows and sites, and without any overlap.



(a) Initial Bin Assignment



(b) 3D Grid Graph

Fig. 3 Flow network construction. (a) Cells are assigned to their nearest bins based on global placement. (b) Each bin is modeled as a vertex. Adjacent bins on the same die are connected by planar edges. And the bins on different dies with planar overlap are connected by die-to-die (D2D) edges.

TABLE I The main notations used in this paper.

Notations	Descriptions
C	the set of standard cells (indexed by c)
(x'_c, y'_c)	initial low-left corner of cell c
(x_c, y_c)	current low-left corner of cell c
w_c	width of cell c (w_c^+ for top die, w_c^- for bottom die)
V	the set of bins (indexed by v)
(x_v, y_v)	low-left corner of bin v
w_v	width of bin v
$\Gamma(v)$	the set of (fractional) cells assigned to bin v
$D_c(v)$	displacement of cell c assigned to bin v

The main notations used in this paper are listed in TABLE I. The top and bottom dies are divided into horizontal placement rows with a row height denoted as h_r. Note that if heterogeneous technology integration is adopted, the row heights may differ for the top die (h_r^+) and the bottom die (h_r^-). For simplicity, we will omit the superscript when there is no ambiguity. The height of a standard cell is equal to the height of a row on its corresponding die. We treat the macros as blockages, hence each row may be segregated into several segments. And each segment is divided into uniform bins with a bin width denoted as w_v. Initially, all the movable cells C are assigned to their nearest dies and then to their nearest bins based on the global placement, as illustrated in Fig. 3(a). Cells are allowed to be fractionally assigned to two bins if the bins are on the same die and horizontally adjacent. Hence, the objects that we move are fractional cells $\gamma = (c_\gamma, \rho_\gamma)$, where $c_\gamma = c$ and $\rho_\gamma \in (0, 1]$, and we have $\sum_{\gamma:c_\gamma=c} \rho_\gamma = 1$ for any cell c. We denote the bin set as V. Adjacent bins on the same die are connected by planar edges, while bins on different dies with planar overlap are connected by die-to-die (D2D) edges. Thus, we construct a 3D grid graph $G(V, E)$, as illustrated in Fig. 3(b).

For a bin v, let $\text{cap}(v) = w_v$ represent the free capacity that can be used by standard cells. Based on the initial bin assignment, let $\Gamma(v)$ be the set of fractional cells assigned to bin v. We can compute the total width of fractional cells as $w_{\Gamma(v)} = \sum_{\gamma \in \Gamma(v)} w_{c_\gamma} \times \rho_\gamma$, where w_{c_γ} represents the width for cell c_γ. Then we can define the supply value for a bin v:

$$\text{sup}(v) = \max\{0, w_{\Gamma(v)} - w_v\}, \quad (1)$$

similarly, we define the demand value for a bin v:

$$\text{dem}(v) = \max\{0, w_v - w_{\Gamma(v)}\}. \quad (2)$$

If $\text{sup}(v) > 0$, the bin v is overflowed and is treated as a source bin. If $\text{dem}(v) > 0$, the bin v is treated as a sink bin. The goal is to find an assignment of cells to bins such that no bin is overflowed while minimizing the average and maximum cell displacement.

III. Proposed 3D-Flow Legalizer

The overall legalization flow of our 3D-Flow legalizer is shown at the end of this section (Algorithm 2). We first give the mathematical model for flow-based legalization in Section III-A. The core part of our legalizer involves finding the shortest augmenting path on the 3D grid graph using a branch-and-bound algorithm (Section III-B). Our legalizer iteratively augments flows along these paths (Section III-C) to resolve overflowed bins. Once we have determined an assignment of cells to bins, we can legalize all cells within their respective rows (Section III-D). Additionally, our legalizer adopts an incremental legalization step with a cycle-canceling algorithm (Section III-E) to reduce the maximum cell displacement.

A. Legalization as a Minimum-Cost Flow Problem

Based on the flow network $G(V, E)$, we can model the legalization as an assignment problem. The goal is to resolve all overflowed bins while minimizing the total cell displacement. Note that fractional cells are not applicable in this discrete assignment formulation. We should assign the complete cells to their nearest bins. Thus, we use $c \in \Gamma(v)$ to denote a cell c in a bin v instead of fractional cells. Let $S = \{v \mid v \in V, \text{sup}(v) > 0\}$ represent the source bins, and $T = \{v \mid v \in V, \text{dem}(v) > 0\}$ represent the sink bins. Let $R = V - S - T$ denote the remaining bins. We can have the following integer linear programming (ILP) formulation:

$$\begin{aligned}
\min \quad & \sum_{(u,v)\in E} \sum_{c\in\Gamma(u)} \text{cost}_{u,v,c} a_{u,v,c} \\
\text{s.t.} \quad & \sum_{u\in V} \sum_{c\in\Gamma(v)} w_c a_{v,u,c} - \sum_{u\in V} \sum_{c\in\Gamma(u)} w_c a_{u,v,c} \geq \text{sup}(v), && \forall v \in S \\
& \sum_{u\in V} \sum_{c\in\Gamma(u)} w_c a_{u,v,c} - \sum_{u\in V} \sum_{c\in\Gamma(v)} w_c a_{v,u,c} \leq \text{dem}(v), && \forall v \in T \\
& \sum_{u\in V} \sum_{c\in\Gamma(u)} w_c a_{u,v,c} - \sum_{u\in V} \sum_{c\in\Gamma(v)} w_c a_{v,u,c} = 0, && \forall v \in R \\
& a_{u,v,c} \in \{0, 1\}, \quad \forall (u, v) \in E, \forall c \in \Gamma(u),
\end{aligned} \tag{3}$$

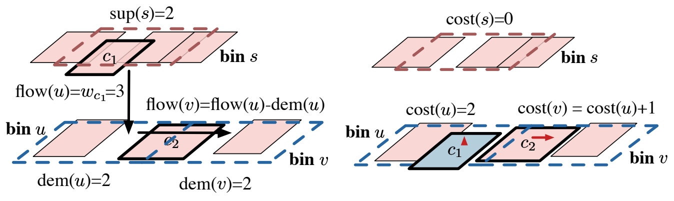
where $a_{u,v,c} = 1$ indicates the cell $c \in \Gamma(u)$ is moved from bin u to bin v; otherwise, the cell remains in bin u. The cost for such movement is calculated as:

$$D_c(v) = |x_c - x'_c| + |y_c - y'_c|, \tag{4}$$

$$\text{cost}_{u,v,c} = D_c(v) - D_c(u). \tag{5}$$

$D_c(v)$ is the estimated displacement when cell c is assigned to bin v. Since bins are aligned to placement rows, we have $y_c = y_v$, y-coordinate of bin v. And x_c is set as the nearest horizontal position within bin v based on the x-coordinate of global placement x'_c. For example, if $x'_c < x_v$, then we have $x_c = x_v$.

According to Equation (3), if w_c is constant for all cells, the optimization problem is a minimum-cost flow problem, which can be solved in $O(|V| \log |V| (|E| + |V| \log |V|))$. However, in practice, we have different cell sizes, resulting in an NP-hard problem. Given the sheer number of cells in modern designs, solving the ILP problem directly is impractical for legalization. One approach is to divide each cell into partial cells with unit width to enable a minimum-cost flow formulation. However, the resulting flow may be unattainable and even misleading [9]. The successive shortest path algorithm for solving minimum-cost flow [14] is appropriate for our application, as we can modify it to ensure that only augmentations that can be exactly realized by cell movements are considered, avoiding the flow realization issue.



(a) Candidate Path (b) Cell Movement

Fig. 4 Path augmentation. (a) Candidate path with corresponding flow. (b) Cell movement with corresponding cost. A candidate path $s - u - v$ for solving the overflow in source bin s is shown in (a). At each step, the cells with minimal cost are selected to move. Since s and u belong to different dies, cell c_1 is moved completely. And u and v are horizontally adjacent, cell c_2 is moved partially to minimize displacement cost.

Algorithm 1 Augmentation with Branch and Bound

Input: Flow network $G(V, E)$, a supply bin s.
Output: The shortest augmenting path p_{best} resolving overflow.

```
 1: flow($s$) = sup($s$); cost($s$) = 0; parent($s$) = $nullptr$;
 2: visited($s$) = true;
 3: $Q$.insert($s$)                    ▷ priority queue $Q$ sorted by cost(·);
 4: while $Q \neq \varnothing$ do
 5:     $u = Q$.pop()                        ▷ bin $u$ as source bin;
 6:     for each bin $v$ connected to bin $u$ do     ▷ bin $v$ as sink bin
 7:         if visited($v$) then                ▷ visit more than once
 8:             continue;
 9:         end if
10:         $outFlow, cost$ = computeFlow($u, v$);        ▷ Equation (6)
11:         flow($v$) = $outFlow$; cost($v$) = cost($u$) + $cost$;
12:         parent($v$) = $u$; visited($v$) = true;
13:         if cost($v$) < $(1 + \alpha)$cost($p_{\text{best}}$) then;     ▷ set the bound
14:             if flow($v$) ≤ dem($v$) then;          ▷ a candidate path
15:                 if cost($v$) < cost($p_{\text{best}}$) then
16:                     $p_{\text{best}} = v$;          ▷ update the shortest path
17:                 end if
18:             else
19:                 $Q$.insert($v$);              ▷ insert intermediate path
20:             end if
21:         end if
22:     end for
23: end while
24: return $p_{\text{best}}$;
```

B. Augmentation with Branch-and-Bound

The traditional successive shortest path algorithm [14] begins with a zero flow and iteratively employs Dijkstra's algorithm to identify shortest paths between supply node u and demand node v in the residual graph. However, this approach faces two significant limitations in our application. First, the repeated call of Dijkstra's algorithm becomes computationally expensive for large-scale designs. Second, during iterative optimization, the cost function eq. (5) can yield negative values, indicating that we can move cells toward their initial positions. But Dijkstra's algorithm requires non-negative edge costs. To address these limitations, we propose an enhanced algorithm that combines a modified breadth-first search (BFS) with branch-and-bound techniques to find the shortest augmenting path in the 3D grid graph, achieving high solution quality with better efficiency.

Algorithm 1 presents our path augmentation algorithm. Since our algorithm ensures that each bin is traversed only once (line 7), an

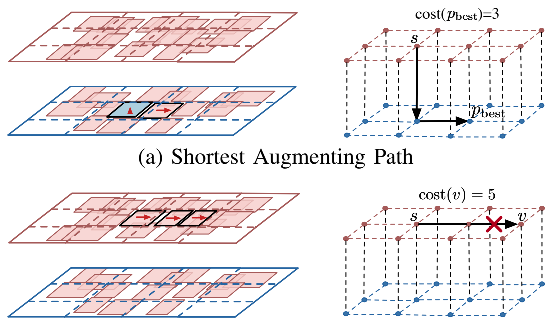


Fig. 5 Augmentation with branch and bound. (a) Augmenting path p_{best} with the smallest cost. (b) Eliminated branch with large cost. Moved cells are highlighted with red arrows indicating displacement. The **blue**-colored cell in (a) is from top die. The intermediate paths with large cost are stopped from being explored.

n-ary search tree rooted at the supply bin s is constructed during the search process. Each branch of the tree represents a candidate path for moving cells to resolve overflow. For each bin v in the search tree, we define parent(v) as the predecessor of v, flow(v) as the required flow that needs to be moved from bin parent(v) to bin v, and cost(v) as the total displacement cost for the path from root s to bin v. For the root bin s, the required flow is set as the supply value sup(s). Unlike the typical BFS algorithm, we prioritize bins with lower displacement costs when extending the search path (line 5).

A step that is critical for the algorithm is the selection of the set $C(u,v)$ of fractional cells to be moved from u to v in line 10, which should resolve overflow with minimal displacement cost. To ensure that no new overflowed bin is created, we compute the $outFlow$ as follows:

$$outFlow = w_{C(u,v)} \geq \text{flow}(u) - \text{dem}(u), \qquad (6)$$

where dem(u) represents the flow that can be sunk by bin u. Thus, $outFlow$ in Equation (6) is the flow that needs to be moved out when flow(u) is moved into bin u. If bins u and v are horizontally adjacent in the same die, we can choose the fractional cell $\gamma \in \Gamma(u)$ with $\rho_\gamma < 1$ to ensure that $w_{C(u,v)} = \text{flow}(u) - \text{dem}(u)$. Specifically, we sort the fractional cells in $\Gamma(u)$ by the cost in Equation (5), which reflects the displacement cost for such movement. The cheapest fractional cells are then selected to reduce their ρ_γ and the path cost is updated accordingly (line 11). On the other hand, if u and v belong to different rows or dies, we can only move cells completely because no cell can be distributed across different segments. Hence, moving γ from u to v means removing all parts of c_γ from their bins (which must be in the same segment as u) and assigning them to v. In these cases, we select the cells such that $w_{C(u,v)} \geq \text{flow}(u) - \text{dem}(u)$ with the lowest displacement cost. If the flow(v) can be completely sunk by bin v, a candidate path for resolving the overflow is found (line 14). An example of one candidate path is illustrated in Fig. 4.

We keep track of the path p_{best} with the smallest cost (line 16). During the search process, any branch with a cost greater than $(1+\alpha)\times\text{cost}(p_{\text{best}})$ will be eliminated (line 13), where $\alpha \geq 0$, controlling the solution space that our algorithm explores. When $\alpha = 0$, the algorithm becomes a greedy search, including only intermediate paths with costs smaller than the current best cost. Conversely, a large α will cause the algorithm to traverse the entire search tree. Although there are edges with negative costs, such edges are few. Thus, a small $\alpha = 0.1$ can help our algorithm find the shortest augmenting path with

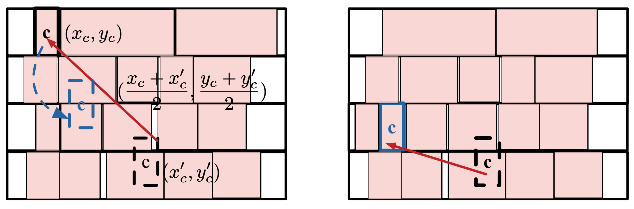


Fig. 6 Post-optimization with cycle-canceling. (a) Negative cycle. (b) Reduced maximum displacement. The black dashed box indicates the initial position. Our algorithm only selects cells with displacement larger than the predifined threshold to move toward their initial positions. Our incremental legalization will reduce the maximum displacement while making minimal perturbation to other cells.

great efficiency in practice. Fig. 5 shows an example that a branch with large cost is eliminated.

Note that BonnPlaceLegal [10] also adopts a flow-based methodology. However, it utilizes vanilla Dijkstra's algorithm to find the augmenting path on a 2D grid graph, which is not applicable for negative-cost movements. Based on the 3D flow network, our algorithm can move cells freely both intra-die and inter-die to minimize the total displacement with improved computational efficiency.

C. Moving Cells Along Augmenting Path

Once the shortest augmenting path is found by Algorithm 1, the set of fractional cells $C(u,v)$ are moved between the adjacent bins along the path to resolve the overflow. The cell movement is performed by backtracking the path from the leaf bin p_{best} to the root supply bin s. For each current bin v, we apply the same method as in Algorithm 1 to select the set of fractional cells $C(u,v)$ from the parent bin $u = \text{parent}(v)$ with the minimal cost, where $w_{C(u,v)} = \text{flow}(v)$. The procedure repeats until reaching the root supply bin s, thereby resolving the overflow in bin s.

D. Legalization Within a Row

After the flow-based legalization, cells are assigned to bins aligned with placement rows. However, the overlap between cells within a bin may still exist. Many well-known row-based placement algorithms [4], [27], [28] can be used to remove the overlap between cells within a row. We use the `PlaceRow` algorithm in Abacus [4], which efficiently minimizes quadratic cell movement in linear time.

E. Post-Optimization with Cycle-Canceling

At the final step, we propose a post-optimization technique inspired by the cycle-canceling algorithm [29] to minimize maximum displacement. The cycle-canceling algorithm improves the objective function by finding negative cost cycles in the residual graph and augmenting the flow on these cycles. In our application, we construct such negative cycles by moving cells with large displacement toward their initial positions. Specifically, we select the cells with displacement $D_c > threshold$, where $threshold = \max\{5h_r, \frac{D_{\max}}{2}\}$, h_r is the row height and $D_{\max}$ is the current maximum displacement. For each selected cell c, we reposition it at the midpoint between its current and initial positions $(\frac{x_c+x'_c}{2}, \frac{y_c+y'_c}{2})$. This movement may introduce a few overflowed bins, and we perform the flow-based legalization again and remove all the cell overlap. Fig. 6 shows an example of the proposed cycle-canceling algorithm. Our flow-based legalizer enables incremental legalization inherently, therefore, the post-optimization can significantly reduce maximum displacement with minimal perturbation to other cells.

Algorithm 2 3D-Flow Legalizer

```
1: Divide the layout into bins $V$;
2: Assign cells $C$ to their nearest bins;
3: Connect neighboring bins by edges in a 3D grid graph $G$;
4: $Q \leftarrow \{v \in V \mid \text{sup}(v) > 0\}$;        ▷ priority queue sorted by sup(·)
5: while $Q \neq \varnothing$ do
6:     $s = Q$.pop();                                  ▷ bin $s$ as source bin
7:     Call Augmentation path algorithm;               ▷ Algorithm 1
8:     Move cells;                                     ▷ Section III-C
9:     Update overflowed bins in $Q$;
10: end while
11: for each placement row $r$ do
12:     PlaceRow($r$);                                 ▷ Section III-D
13: end for
14: Post-optimization with cycle-canceling;            ▷ Section III-E
```

F. Overall Algorithm

Algorithm 2 provides an overview of the proposed 3D-Flow legalizer. We process all supply bins in descending order based on their supply values. For each supply bin s, we identify the augmenting path with minimal cost using Algorithm 1 and adjust cell positions accordingly to resolve overflow. This process iterates until all overflow is eliminated.

The choice of bin width w_v involves a trade-off between result quality and efficiency. For the flow-based legalization, we set the bin width $w_v = 10\bar{w}_c$ to enhance efficiency, where $\bar{w}_c$ is the average width of all cells. During post-optimization, w_v is reduced to $5\bar{w}_c$ for more precise cost estimation, given the limited number of overflowed bins. Our algorithm is flexible enough to handle various constraints. In heterogeneous technology integration, cell movement across die-to-die edges requires updating cell widths. We manage different widths w_c^+ on the top die and w_c^- on the bottom die, ensuring the utilization rate of each die is not exceeded.

$$\text{cost}_{u,v,c} = D_c(v) - D_c(u) + \text{sup}(v) - \text{dem}(v). \tag{7}$$

We set the die-to-die edge cost as shown in Equation (7) to encourage cells to move from the congested die while penalizing unnecessary cross-die movements.

IV. Experimental Results

A. Experimental Setup

To validate our proposed 3D legalizer, we conducted experiments on ICCAD 2022 [25] and 2023 [26] contest benchmarks, which address standard-cell placement and mixed-size placement for F2F bonded 3D ICs, respectively. The detailed benchmark statistics are shown in TABLE II. The global placements for ICCAD 2022 and 2023 contest benchmarks were generated by SOTA true-3D placers [18] and [19], respectively. We implemented the proposed 3D legalizer in C++. All the experiments were performed on a Linux machine with 20-core Intel Xeon Silver 4210R CPU (2.40GHz) and 24GB RAM.

B. Comparison with SOTA Legalizers

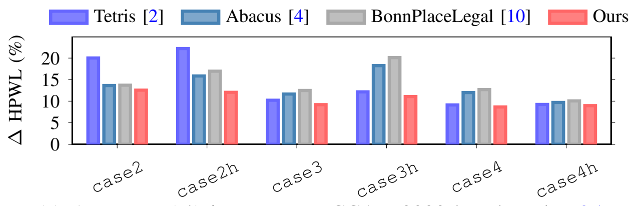
We compared 3D-Flow legalizer with existing SOTA 2D legalizers including Tetris [2], Abacus [4], and BonnPlaceLegal [10]. Tetris and Abacus are the most popular legalization algorithms based on greedy search, which are extensively used in modern global placers [6], [7], [18]–[20]. And BonnPlaceLegal is a competitive flow-based legalizer.

The experimental results are shown in TABLE III and TABLE IV. For ICCAD 2022 contest benchmarks, which contain only standard cells, legalizers have more flexibility to reposition cells without large macros as blockages. Our 3D-Flow legalizer consistently obtained the best results for all the cases, as shown in TABLE III. Compared to Tetris, it achieved 1.61× less average displacement and 2.31× less maximum displacement. Tetris places cells sequentially based on their x-position, resulting in larger displacement for cells placed later. And our legalizer reduced average displacement by 13% and maximum displacement by 54% compared to Abacus. Although Abacus allows movement of already placed cells, they remain on a fixed row, causing significant displacement for later placements. Moreover, 3D-Flow outperformed BonnPlaceLegal by 15% and 43% less average and maximum displacement. While BonnPlaceLegal uses a flow-based method, it limits edge costs to positive values, missing further optimization opportunities. In contrast, our 3D-Flow repositions cells in a 3D solution space, fully utilizing the vertical stacking of 3D ICs. Leveraging the global view in the network flow method, our legalizer allows simultaneous movement of multiple cells across rows and dies to minimize displacement. The post-optimization with cycle-canceling further reduces maximum displacement while maintaining quality.

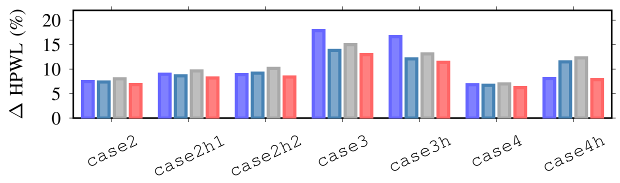
For ICCAD 2023 contest benchmarks, large macros divide placement rows into segments, posing challenges in finding available space near macro boundaries. Despite this, our 3D-Flow legalizer outperformed all SOTA legalizers, as demonstrated in TABLE IV. It achieved 1.46×, 1.08×, and 1.13× less average displacement, and 2.97×, 1.40×, and 1.34× less maximum displacement compared to

TABLE II The statistics of ICCAD 2022 [25] and ICCAD 2023 [26] contest benchmarks. h_r^+ and h_r^- represent row height values of the top and bottom die, respectively.

ICCAD 2022	#Cells	#Macros	#Nets	h_r^+	h_r^-
case2	2735	0	2644	176	252
case2h	2735	0	2644	252	252
case3	44764	0	44360	115	115
case3h	44764	0	44360	92	115
case4	220845	0	220071	92	115
case4h	220845	0	220071	103	115
ICCAD 2023	**#Cells**	**#Macros**	**#Nets**	h_r^+	h_r^-
case2	13901	6	19547	33	33
case2h1	13901	6	19547	33	48
case2h2	13901	6	19547	33	48
case3	124231	34	164429	33	48
case3h	124231	34	164429	36	48
case4	740211	32	758860	92	115
case4h	740211	32	758860	55	69



(a) Δ HPWL (%) increase on ICCAD 2022 benchmarks [25].



(b) Δ HPWL (%) on ICCAD 2023 benchmarks [26].

Fig. 7 HPWL increase (%) of global placement and legal placement for different methods. (a) ICCAD 2022 contest benchmarks. (b) ICCAD 2023 contest benchmarks.

TABLE III Comparison on the legalization results of our 3D-Flow legalizer with existing SOTA legalizers for ICCAD 2022 benchmarks. **RT** (s) stands for the total runtime including file IO. The average displacement and maximum displacement are normalized by the row height.

ICCAD 2022	Tetris [2]			Abacus [4]			BonnPlaceLegal [10]			Ours		
	Avg. Disp.	Max. Disp.	RT	Avg. Disp.	Max. Disp.	RT	Avg. Disp.	Max. Disp.	RT	Avg. Disp.	Max. Disp.	RT
`case2`	0.801	7.27	0.23	0.522	3.05	0.24	0.511	2.91	0.26	**0.503**	**2.77**	0.27
`case2h`	1.091	5.77	0.22	0.593	4.34	0.24	0.610	4.80	0.24	**0.511**	**2.97**	0.25
`case3`	0.929	13.40	1.15	0.705	5.81	1.35	0.727	6.13	2.70	**0.660**	**4.53**	1.54
`case3h`	1.162	8.30	1.19	0.938	9.47	1.44	0.986	7.95	6.28	**0.750**	**4.61**	1.73
`case4`	1.775	12.83	6.79	1.383	16.81	10.98	1.428	15.13	72.93	**1.174**	**11.25**	9.69
`case4h`	1.622	27.59	6.75	1.161	14.96	9.96	1.186	11.94	53.68	**1.100**	**8.14**	10.36
Average	1.613	2.31	**0.76**	1.125	1.54	0.95	1.153	1.43	3.34	**1.000**	**1.00**	1.00

TABLE IV Comparison on the legalization results of our 3D-Flow legalizer with existing SOTA legalizers for ICCAD 2023 benchmarks. **RT** (s) stands for the total runtime including file IO. The average displacement and maximum displacement are normalized by the row height.

ICCAD 2023	Tetris [2]			Abacus [4]			BonnPlaceLegal [10]			Ours		
	Avg. Disp.	Max. Disp.	RT	Avg. Disp.	Max. Disp.	RT	Avg. Disp.	Max. Disp.	RT	Avg. Disp.	Max. Disp.	RT
`case2`	3.029	13.27	0.50	2.199	11.33	0.61	2.308	11.50	2.49	**2.109**	**9.09**	0.64
`case2h1`	3.609	50.28	0.57	2.509	35.28	0.68	2.668	36.85	4.80	**2.433**	**27.03**	0.74
`case2h2`	3.709	46.29	0.55	2.673	39.26	0.71	2.828	34.11	5.19	**2.518**	**27.73**	0.74
`case3`	3.612	195.31	3.07	2.475	136.25	4.52	2.615	131.96	59.83	**2.442**	**70.82**	4.46
`case3h`	3.041	141.31	2.89	2.266	119.31	4.21	2.370	109.11	69.99	**2.204**	**99.05**	5.74
`case4`	1.349	127.69	22.86	1.000	17.32	37.79	1.023	15.32	264.23	**0.920**	**13.77**	39.13
`case4h`	1.709	128.19	23.75	1.432	78.55	47.02	1.492	77.71	315.26	**1.130**	**54.31**	37.73
Average	1.461	2.97	**0.69**	1.076	1.40	1.00	1.127	1.34	8.89	**1.000**	**1.00**	1.00

TABLE V Comparison on the legalization results for our 3D-Flow legalizer with and without die-to-die (D2D) cell movement on ICCAD 2023 contest benchmarks. #Move stands for the number of cells moved across dies.

ICCAD 2023	w/o. D2D		Ours		
	Avg. Disp.	Max. Disp.	Avg. Disp.	Max. Disp.	#Move
`case2`	2.197	11.33	**2.109**	**9.09**	35
`case2h1`	2.506	35.28	**2.433**	**27.03**	46
`case2h2`	2.711	32.10	**2.518**	**27.73**	54
`case3`	2.582	98.98	**2.442**	**70.82**	493
`case3h`	2.229	107.84	**2.204**	**99.05**	373
`case4`	0.931	15.32	**0.920**	**13.77**	2479
`case4h`	1.358	54.33	**1.130**	**54.31**	2081
Average	1.068	1.19	**1.000**	**1.00**	-

Tetris, Abacus, and BonnPlaceLegal, respectively.

The HPWL increase results for ICCAD 2022 and 2023 contest benchmarks are shown in Fig. 7. Although HPWL increase may not always align with displacement results, our 3D-Flow consistently achieved the best outcomes, with 1.29×, 1.30×, and 1.38× less increase for ICCAD 2022 contest benchmarks, and 1.17×, 1.13×, and 1.22× less increase for ICCAD 2023 contest benchmarks, compared to Tetris, Abacus, and BonnPlaceLegal, respectively.

Regarding runtime, Tetris is the fastest due to its simple procedure, being 1.32× and 1.45× faster than our method on ICCAD 2022 and 2023 contest benchmarks. And 3D-Flow achieved better results with similar runtime compared to Abacus. However, BonnPlaceLegal showed poor scalability in large cases. Leveraging our efficient branch-and-bound algorithm, 3D-Flow achieved 3.34× and 8.89× speedup over BonnPlaceLegal on ICCAD 2022 and 2023 contest benchmarks, respectively.

C. Ablation Study on Die-to-Die Cell Movement

To validate the effectiveness of die-to-die movement in our 3D legalizer, we performed experiments on ICCAD 2023 contest benchmarks for our 3D-Flow legalizer with and without die-to-die cell movement. The experimental results are shown TABLE V. In the last column, we list the number of cells moved across dies. By moving only a few cells vertically, our legalizer finds white space for other cells to minimize displacement in a global view. Our 3D-Flow legalizer outperformed its 2D counterpart by 6.8% and 19% less average and maximum displacement. The displacement of different settings for `case3` is visualized in Fig. 8.

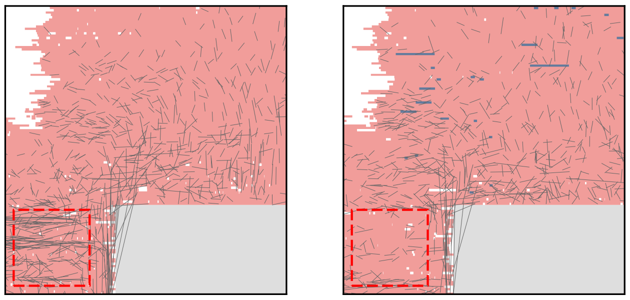

(a) w/o. D2D cell movement (b) Our 3D-Flow Legalizer

Fig. 8 Displacement visualization for the top die of `case3` in ICCAD 2023 contest benchmarks. (a) Without die-to-die (D2D) cell movement. (b)Using our 3D legalizer. The gray block represents the macro. The **blue**-colored cells in (b) are from the bottom die. Black lines indicate cell displacement.

V. Conclusion

This paper introduces 3D-Flow, the first 3D legalizer designed to minimize cell displacement for 3D ICs using network flow methods. Leveraging a novel branch-and-bound algorithm, 3D-Flow efficiently identifies the shortest augmenting path on a 3D grid graph to iteratively resolve overflowed bins. By fully utilizing the vertical stacking of 3D ICs, our legalizer finds legal positions for cells with minimal displacement in a global view. Additionally, a post-optimization with a cycle-canceling algorithm further reduces maximum displacement while preserving result quality. Experimental results on ICCAD 2022 and 2023 benchmarks show that our 3D legalizer achieves up to 13% and 43% less average and maximum displacement, respectively, compared to state-of-the-art legalizers, within a similar runtime.

Acknowledgements

The project is supported in part by Research Grants Council of Hong Kong SAR (No. RFS2425-4S02 and No. CUHK14211824), and the MIND project (MINDXZ202404).

References

[1] G. Murali and S. K. Lim, "Heterogeneous 3D ICs: Current status and future directions for physical design technologies," in *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*, 2021, pp. 146–151.

[2] D. Hill, "Method and system for high speed detailed placement of cells within an integrated circuit design," Apr. 9 2002, US Patent 6,370,673.

[3] N. Viswanathan, M. Pan, and C. Chu, "FastPlace 3.0: A fast multilevel quadratic placement algorithm with placement congestion control," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 2007, pp. 135–140.

[4] P. Spindler, U. Schlichtmann, and F. M. Johannes, "Abacus: fast legalization of standard cell circuits with minimal movement," in *ACM International Symposium on Physical Design (ISPD)*, 2008, pp. 47–53.

[5] W.-K. Chow, J. Kuang, X. He, W. Cai, and E. F. Y. Young, "Cell density-driven detailed placement with displacement constraint," in *ACM International Symposium on Physical Design (ISPD)*, 2014, pp. 3–10.

[6] M.-K. Hsu, Y.-F. Chen, C.-C. Huang, S. Chou, T.-H. Lin, T.-C. Chen, and Y.-W. Chang, "NTUplace4h: A novel routability-driven placement algorithm for hierarchical mixed-size circuit designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 33, no. 12, pp. 1914–1927, 2014.

[7] 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 *ACM/IEEE Design Automation Conference (DAC)*, 2019.

[8] U. Brenner and J. Vygen, "Legalizing a placement with minimum total movement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 23, no. 12, pp. 1597–1613, 2004.

[9] M. Cho, H. Ren, H. Xiang, and R. Puri, "History-based vlsi legalization using network flow," in *ACM/IEEE Design Automation Conference (DAC)*, 2010, pp. 286–291.

[10] U. Brenner, "Bonnplace legalization: Minimizing movement by iterative augmentation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 32, no. 8, pp. 1215–1227, 2013.

[11] N. K. Darav, I. S. Bustany, A. Kennings, and L. Behjat, "A fast, robust network flow-based standard-cell legalization method for minimizing maximum movement," in *ACM International Symposium on Physical Design (ISPD)*, 2017, pp. 141–148.

[12] J. Monteiro, M. Johann, and L. Behjat, "An optimized cost flow algorithm to spread cells in detailed placement," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 24, no. 3, pp. 1–16, 2019.

[13] H. Ren, D. Z. Pan, C. J. Alpert, and P. Villarrubia, "Diffusion-based placement migration," in *ACM/IEEE Design Automation Conference (DAC)*, 2005, pp. 515–520.

[14] A. V. Goldberg and R. E. Tarjan, "Finding minimum-cost circulations by successive approximation," *Mathematics of Operations Research*, vol. 15, no. 3, pp. 430–466, 1990.

[15] K. Chang, S. Sinha, B. Cline, R. Southerland, M. Doherty, G. Yeric, and S. K. Lim, "Cascade2D: A design-aware partitioning approach to monolithic 3D IC with 2D commercial tools," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2016, pp. 1–8.

[16] 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 (TCAD)*, vol. 36, no. 10, pp. 1716–1724, 2017.

[17] M.-K. Hsu, V. Balabanov, and Y.-W. Chang, "TSV-aware analytical placement for 3-D IC designs based on a novel weighted-average wirelength model," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 32, no. 4, pp. 497–509, 2013.

[18] P. Liao, Y. Zhao, D. Guo, Y. Lin, and B. Yu, "Analytical die-to-die 3d placement with bistratal wirelength model and gpu acceleration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2023.

[19] Y. Zhao, P. Liao, S. Liu, J. Jiang, Y. Lin, and B. Yu, "Analytical heterogeneous die-to-die 3d placement with macros," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2024.

[20] Y.-J. Chen, C.-H. Hsieh, P.-H. Su, S.-H. Chen, and Y.-W. Chang, "Mixed-size 3d analytical placement with heterogeneous technology nodes," in *ACM/IEEE Design Automation Conference (DAC)*, 2024, pp. 1–6.

[21] M. Motoyoshi, "Through-silicon via (TSV)," *Proceedings of the IEEE*, vol. 97, no. 1, pp. 43–48, 2009.

[22] S. S. K. Pentapati and S. K. Lim, "Heterogeneous monolithic 3d ics: Eda solutions, and power, performance, cost tradeoffs," in *ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2021, pp. 925–930.

[23] C. Premachandran, J. Lau, L. Xie, A. Khairyanto, K. Chen, M. E. P. Pa, M. Chew, and W. K. Choi, "A novel, wafer-level stacking method for low-chip yield and non-uniform, chip-size wafers for mems and 3d sip applications," in *IEEE Electronic Components and Technology Conference*. IEEE, 2008, pp. 314–318.

[24] S. Liu, J. Jiang, Z. He, Z. Wang, Y. Lin, B. Yu, and M. Wong, "Routing-aware legal hybrid bonding terminal assignment for 3d face-to-face stacked ics," in *ACM International Symposium on Physical Design (ISPD)*, 2024, pp. 75–82.

[25] 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 *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2022, pp. 1–5.

[26] K.-S. Hu, H.-Y. Chi, I.-J. Lin, Y.-H. Wu, W.-H. Chen, and Y.-T. Hsieh, "2023 ICCAD CAD Contest Problem B: 3D placement with macros," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2023.

[27] A. B. Kahng, P. Tucker, and A. Zelikovsky, "Optimization of linear placements for wirelength minimization with free sites," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*. IEEE, 1999, pp. 241–244.

[28] A. B. Kahng, I. L. Markov, and S. Reda, "On legalization of row-based placements," in *ACM Great Lakes Symposium on VLSI (GLSVLSI)*, 2004, pp. 214–219.

[29] A. V. Goldberg and R. E. Tarjan, "Finding minimum-cost circulations by canceling negative cycles," *Journal of the ACM (JACM)*, vol. 36, no. 4, pp. 873–886, 1989.