Pin-Accessible Legalization for Mixed-Cell-Height Circuits

Haocheng Li, Wing-Kai Chow, *Member, IEEE*, Gengjie Chen, Bei Yu, *Member, IEEE*, and Evangeline F.Y. Young, *Senior Member, IEEE*

***Abstract*—Placement is one of the most critical stages in the physical synthesis flow. Circuits with increasing numbers of cells of multirow height have brought challenges to traditional placers on efficiency and effectiveness. Besides providing an overlap-free solution close to the global placement (GP) solution, constraints on power and ground (P/G) alignments, fence region, and routability (e.g., edge spacing and pin short/inaccessible) should be considered. In this article, we propose a legalization method for mixed-cell-height circuits by a window-based cell insertion technique and two post-processing network flow-based optimizations. Compared with the champion of the ICCAD 2017 Contest, our algorithm achieves 35% and 13% less average and maximum displacement, respectively, as well as significantly fewer routability violations. Comparing our algorithm with the state-of-the-art algorithms on this problem, there is an 8% improvement in average displacement with comparable maximum displacement. The source code of our legalization is available at https://github.com/cuhk-eda/ripple.**

***Index Terms*—Bipartite matching, legalization, linear programming, network flow algorithm, overlap removal, placement.**

I. Introduction

STANDARD cells are designed with the same height and aligned on placement sites for the simplicity of physical synthesis. With the globalized design and fabrication of integrated circuits, the semiconductor feature size continuously shrinks down to three nanometers [2]. However, the number of in-cell tracks diminishes significantly and the internal routability within a single-site-high cell becomes inadequate. In subfive-nanometer technology nodes, there are placement sites of just three-track-high [3], which results in design difficulties of complex standard cells with high driving strength, such as multiplexers [4] and multibit flip-flops [5]. On the other hand, cell area is wasted in the region for N-type transistors when large cell width is required by P-type transistors to achieve similar rise and fall transition time [6]. For ascending performance and efficiency, complex cells are now designed with multirow height and two-layer metal routing while simple cells remain single-row height and one-layer metal routing [7]. Consequently, the challenges in physical synthesis migrate from cell design to cell placement.

As the name suggested, the placement stage of physical synthesis allocates standard cells on sites of the core area. Traditionally, placement consists of three steps: 1) global placement (GP); 2) legalization; and 3) detailed placement. GP imports cells from the netlist and spreads the cells optimizing wirelength with analytical models [8]. Multiple objectives are considered, including congestion, timing, and power, but physical constraints, such as overlapping, site alignment, and special net connection, are neglected. The physical constraints are instead resolved in the later step, legalization, where the quality of GP solution is preserved as much as possible, i.e., the average displacement of cells is minimized [9]. Finally, further refinement is conducted in detailed placement where the physical constraints are satisfied and wirelength is further reduced [10]. Therefore, the introduction of mix-cell-height circuits migrates most of the problems to the legalization step.

The previous works on legalization algorithms for mixed-cell-height circuits can be categorized into four types. The earlier literature converts the mixed-cell-height circuit legalization problem into a single-cell-height problem. Wu and Chu [11] constructed artificial double-row cells from pairs of single-row cells by graph matching and conducted a double-row detailed placement on cells with the same double-row height. They consider width difference, cell connectivity, and displacement in the weight of graph edges. This method is under the assumption that all double-row-high cells have the same power and ground (P/G) rail configuration. While maintaining the cell area, Dobre *et al.* [12] modified the mixed-cell-height library to have the same height with the shortest cell and partitioned the floorplan based on an initial placement solution conducted by the modified same-cell-height library and conventional detailed placement algorithms. A particular cell height is then specified to each partition and conventional detailed placement algorithms are performed again for each partition. Its limitation is that all cell functions should have implementations in every cell height. The empirical results show that mixed-cell-height designs achieve area and power reduction compared with same-cell-height designs but cell height swapping is required in this procedure and, as a strong and unnecessary constraint, selecting the same cell height for each floorplan partition affects the estimation of timing

Manuscript received May 21, 2020; revised August 20, 2020 and November 14, 2020; accepted December 28, 2020. Date of publication January 21, 2021; date of current version December 23, 2021. This work was supported in part by the Research Grants Council of Hong Kong Special Administrative Region, China, under Project CUHK 14202218. This work is an extension of [1]. This article was recommended by Associate Editor I. H.-R. Jiang. *(Corresponding author: Haocheng Li.)*

Haocheng Li, Gengjie Chen, Bei Yu, and Evangeline F.Y. Young are with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong (e-mail: hcli@cse.cuhk.edu.hk; gjchen@cse.cuhk.edu.hk; byu@cse.cuhk.edu.hk; fyyoung@cse.cuhk.edu.hk).

Wing-Kai Chow is with Cadence Design Systems, Austin, TX 78759 USA (e-mail: wkchow@cadence.com).

Digital Object Identifier 10.1109/TCAD.2021.3053223

and power in logic synthesis and highly restricts the solution space.

Similar to the single-cell-height circuit legalizer Tetris [13] and dynamic-programming-based Abacus [14], the second type of mixed-cell-height circuit legalization algorithms honors the horizontal cell order of GP. For example, Wang *et al.* [15] extended Abacus and optimized a quadratic programming (QP) of multirow cells. A number of cell pins are regarded as the weight of the quadratic cell displacement to restrict the wirelength overhead. They legalize cells from left to right and evaluate the legalization cost of each cell in every neighboring row. The cost of dead space is discussed in this method with six cases, which is time consuming compared with other methods. Chen *et al.* [16] proposed a much faster method by relaxing the constraint of right boundary and dividing multirow cells into multiple single-row cells whose adjacency is guaranteed by Lagrangian relaxation. By applying the Karush–Kuhn–Tucker (KKT) conditions, the quadratic legalization problem is then transformed into a linear complementary problem (LCP) and solved by the modulus-based matrix splitting iteration method (MMSIM) whose convergence requires the just-mentioned separated formulation of multirow cells. The cells placed outside of right boundary are finally resolved by a Tetris-like method, which results in large maximum displacement for dense designs with irregular fence shapes. To resolve the effect of locally dense area, Zhu *et al.* [17] proposed a movement-aware cell reassignment method. After initially solving the quadratic legalization problem by MMSIM, disruptive cells are identified and reassigned toward sparse areas. MMSIM is then repeatedly conducted until the objective function has no further improvement. However, maintaining the cell order of GP loses solution space and may cause poor results especially for dense designs.

The third type is free from the artificial restriction on cell order. Chow *et al.* [18] proposed a multirow local legalization (MLL) algorithm where cells are legalized sequentially. When legalizing a target cell, the row assignments and relative order of previously legalized cells are maintained. In a window around the GP location of the target cell, different row assignments and insertion points of the target cell are explored and the total displacement is minimized by shifting cells in the local region horizontally. Its major limitation is that the later a cell is placed, the higher potential exists for the cell to be placed with large displacement. In addition, the minimized displacement is w.r.t. the current locations of local cells, which can be accumulated to large displacements w.r.t. GP locations after many iterations. MrDP [19] proposes a wirelength-driven legalization based on a chain move scheme and extends a dual min-cost flow (MCF) method [20] from single-row to multirow cells for postrefinement. A potential problem is that using half-perimeter wire-length (HPWL) instead of displacement as the objective function in legalization may disturb other objectives optimized in GP.

There are some recent works that legalize mixed-cell-height circuits with additional constraints, such as IR drop mitigation [21] and half-row fragmentation [22]. However, none of the previous work solve the problem comprehensively. Some of them target at minimizing HPWL while some focus on the total displacement, but none of them simultaneously handle other important measures and constraints, such as the existence of fence region and routability issues, which include pin inaccessible, pin short, and edge spacing.

In this article, we present a fast and high-quality legalization framework for standard cells with mixed cell heights, which outperforms the state of the art under the displacement objective. Our legalizer optimizes both the maximum and average displacement. It also considers fences and routability constraints, minimizing the number of violations. Our major contributions are as follows.

1) We develop a mixed-cell-height circuit legalizer optimizing the maximum and average displacement with constraints on fences, edge spacing, and pin accessibility.
2) We devise a thread-safe method called multirow global legalization (MGL) that inserts a cell optimally into a window, minimizing the average displacement of all the cells in the region from their GP instead of their current positions.
3) An iterative bipartite graph matching is devised to minimize the maximum displacement among a group of cells that can exchange their positions without creating additional violations.
4) We extend the MCF formulation of the fixed-row-and-order problem to the one that optimizes a weighted sum of the maximum and average displacement, with range constraints on the cell movements to avoid pin short and pin access violations.

The remainder of this article is organized as follows. Section II illustrates the problem formulation and constraints. Section III provides a detailed explanation of our proposed techniques. Section IV verifies the effectiveness of our approach, followed by a conclusion in Section V.

II. Problem Formulation

Given a set of m multirow height cells $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$. The cell height and displacement are measured in terms of the multiple of single row height. Let $\mathcal{H}$ be the set of cell height, and $\mathcal{C}_h \subseteq \mathcal{C}$ be the set of cells whose height is h. The problem is to place each cell c_i from GP (x_i', y_i') into (x_i, y_i) with a corresponding displacement:

$$\delta_i = \delta_{xi} + \delta_{yi} = |x_i - x_i'| + |y_i - y_i'| \tag{1}$$

such that the maximum and average displacement is minimized:

$$S_{\text{am}} = \frac{1}{|\mathcal{H}|} \sum_{h \in \mathcal{H}} \frac{1}{|\mathcal{C}_h|} \sum_{c_i \in \mathcal{C}_h} \delta_i \tag{2}$$

which is the metric used in the ICCAD 2017 Contest [9]. Multirow height cells provide high driving strength so that they potentially supply more fanouts and are more critical in timing. Being minorities, the displacement of multirow cells is better honored. Besides, this objective can better verify the effectiveness of legalizers on cells of any height.

Besides, cells should be overlap free and aligned to placement sites of the chip. The P/G alignment and the fence region constraint are treated as hard constraints:

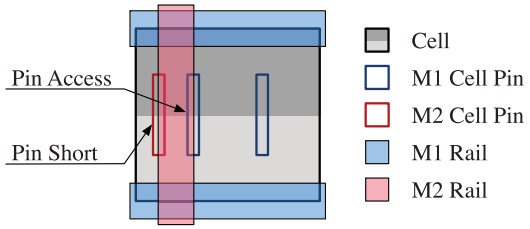


Fig. 1. Pin access and pin short.

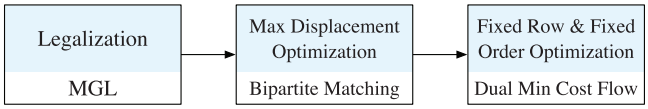


Fig. 2. Proposed legalization flow.

1) Cells with even cell heights must be placed in alternate rows with aligned P/G rails [18];
2) Cells assigned to a fence region must be placed inside the fence boundary [23].

Note that there is no restriction on the row assignments for cells of odd cell height because they can be flipped vertically to correctly align with the P/G rails.

We divide the routability violations of cell pins in two categories and consider them as soft constraints [24]. Intercell pin short or spacing violations are modeled as edge spacing. Violations with P/G grids and input/output (I/O) pins are modeled as pin short or pin access violations.

1) An edge-type rule is specific to neighboring vertical-edge pairs of cell instances. *Cell edges* particularly refers to vertical edges of cells and are categorized into *cell edge types*. A minimum spacing is required between any two cell edge types.
2) Signal pins of cells should be able to connect with vias or regular wires without creating violations. If a signal pin is short or inaccessible due to the P/G grids and I/O pins, there may not be available access point for the pin in detailed routing.

In the modern chip design, the P/G rails are usually regular grids running horizontally or vertically in alternate metal layers. Note that a signal pin on metal layer k is short if it overlaps with a P/G rail or an I/O pin on metal layer k. A signal pin is inaccessible if it cannot be accessed by a regular wire without creating violations on metal layer k or it cannot be accessed by a via without creating violations on metal layer $k+1$. As shown in Fig. 1, the left pin on metal layer one (M1) has pin access problem with the rail on metal layer two (M2), and the M2 pin is short with the M2 rail.

III. Algorithms

The overall algorithmic flow is illustrated in Fig. 2, which consists of three stages.

1) Given a GP solution with multicell-height circuits, we first legalize it by MGL, which inserts the cells into the placement region. Note that a cell may belong to a specific fence region. Cells that do not belong to any fence regions are placed in the *default fence* region, which is outside all other given fence regions.
2) Next, the maximum displacement is optimized by swapping cells of the same type in the same fence region. The displacement cost function is linear at the beginning and exponential afterward to maintain the average displacement at the same time.
3) Finally, keeping the rows and cell order unchanged, the average and maximum displacement is further optimized by linear programming.

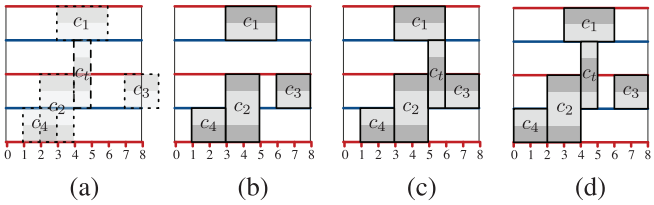


Fig. 3. Comparison between MLL and our purposed MGL. (a) GP. (b) Four cells legalized; Final results of minimizing the total displacement w.r.t. (c) current locations (MLL) and (d) GP locations (MGL).

Details of these three major steps are explained in the following sections.

A. Legalization

In this section, we will introduce the MGL method, which legalizes cells sequentially to minimize the average and maximum displacement from the given GP positions.

Inspired by MLL [18], MGL legalizes cells sequentially. Different from MLL that calculates displacement based on the current cell locations and can eventually accumulate a large displacement w.r.t. GP locations, MGL minimizes the displacement from GP locations directly. Fig. 3 illustrates an example, where the given GP positions are shown in Fig. 3(a). Suppose cells $c_1 - c_4$ are legalized before inserting target cell c_t as in Fig. 3(b), which already has a total displacement of two. In Fig. 3(c), MLL optimizes the total displacement w.r.t. current locations and achieves a value of one. However, the total displacement from GP position is actually three. Fig. 3(d) shows the result with minimized total displacement from GP positions (i.e., two) produced by MGL.

Algorithm 1 shows the flow of MGL. When legalizing a *target cell* c_t, a *window* r_t around its GP position (x'_t, y'_t) is considered. Meanwhile, legalized cells that lie completely within r_t are referred to as *local cells*, which can be shifted for legalizing c_t. In MGL, the row and order assignment of local cells are fixed, but those of c_t are enumerated and evaluated. With row and relative order of local cells fixed, inserting c_t with height h_t implies that we need to place c_t in some gaps between the legalized cells in h_t consecutive rows. A combination of those gaps for inserting c_t is an *insertion point*. For a given c_t and r_t, MGL first obtains all the legal insertion points by using the enumerating method in [18] (line 1). It then calculates the optimal displacement cost of all the insertion points (lines 2–10). The displacement curve, which represents the cost of each insertion point with varied x-coordinate of c_t, can be constructed by adding up all displacement curves of the

Algorithm 1 MGL

Require: Window r_t, GP position (x'_t, y'_t) of target cell c_t.
Ensure: Legal positions of c_t and local cells.
1: Find candidate insertion points $\{p_i\}$ in r_t;
2: **for all** $p_i \in \{p_i\}$ **do**
3: **for all** breakpoint b **do**
4: Store (x_b, left slope k_b^l, right slope k_b^r) in *points*;
5: **end for**
6: AcCurve(*points*); ▷ Accumulate slopes.
7: $d_i \leftarrow$ optimal displacement;
8: $x_t^i \leftarrow$ optimal x-coordinate;
9: $y_t^i \leftarrow$ y-coordinate of p_i;
10: **end for**
11: $j \leftarrow \arg\min_i d_i$;
12: Place c_t at (x_t^j, y_t^j) and spread local cells;

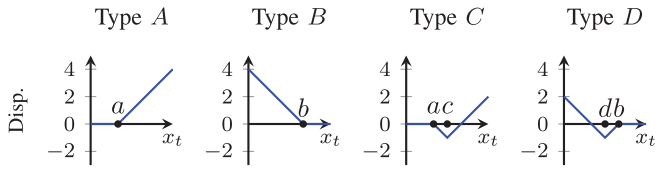


Fig. 4. Four types of displacement curves.

local cells and c_t. The construction of the curves is explained with more details later. The best position to insert c_t with an insertion point is the position with the lowest cost on the displacement curve. After inserting c_t at the position with the lowest cost, local cells are shifted to the left or right when needed to legalize the placement (lines 11 and 12). If there is no valid insertion point, the size of r_t is increased and the legalization for c_t is conducted again. In practice, the window size is initialized as the size of c_t, i.e., to check if c_t can be directly placed at its GP position without creating violations. After that, r_t expands horizontally by six rows and vertically by four rows.

In MLL, when a valid insertion point p is considered, there are only two types of displacement curves for any local cell as illustrated by types A and B in Fig. 4 in which the horizontal axis is the x-position of the target cell and the vertical axis is the displacement contributed by the local cell. The curves are of these shapes since we are measuring the distance from the original positions of the local cells before the target cell is inserted. Cells on the right of p in the window have displacement curves of type A because they may be pushed to the right of their original positions due to the insertion of the target cell at different horizontal positions. Similarly, cells on the left of p in the window have displacement curves of type B because they may be pushed to the left due to the insertion. The turning points of these curves are called *critical positions* as they start contributing to the cost in displacement if the target cell moves beyond the corresponding critical coordinate [18]. Since there are only these two types of curves in MLL, the optimal position to place the target cell can be obtained efficiently by finding the median of all these critical positions.

In MGL, the scenario is more complicated since the displacement is counted w.r.t. the given GP position. There are two more types of displacement curves as illustrated by types $A - D$ in Fig. 4. Without loss of generality, we first discuss the local cells on the right of a valid insertion point p, where there are two possible types of curves A and C. Cells with their GP positions at or on the left of their current positions have displacement curves type A because the target cell will only push them further to the right from their GP positions. For cells with their GP positions on the right of their current positions will have displacement curve type C. The turning points on these curves are either critical positions as in MLL (labeled by a) or positions computable from the GP positions of the local cells (labeled by c). We call all these turning points *breakpoints*. To prove the property of displacement curves, we introduce the definition of clusters as follows.

Definition 1: A *right cluster* is a set of cells that can move to the right together without creating overlaps with other cells. Specifically, a cell without adjacent cell on the right is a right cluster by itself.

Property 1: Every right cluster has a cell without adjacent cell on its right so that this cell is a right cluster itself.

Property 2: Every right cluster has a cell without adjacent cell in the cluster on its left so that the cluster without this cell is still a right cluster.

Lemma 1: If all cells in a window are placed at their optimal positions (i.e., total displacement is the smallest under the fixed row and fixed-order constraint) w.r.t. their GP positions, the displacement curve for moving a right cluster to the right is piecewise linear, nondecreasing, and convex.

Proof of Lemma 1: Based on the two properties, we prove it by mathematical induction. In the base step, if a single cell is a right cluster, the GP position of this cell will not be on the right of its current position because, otherwise, we will have moved it further to the right. Therefore, its displacement curve is like type A, which is piecewise linear, nondecreasing, and convex. In the induction step, we assume that the lemma is correct for all right clusters of r-1 cells. Then, for any right cluster $\mathcal{R}$ of r cells, we split it into a single cell c_l without adjacent cell in $\mathcal{R}$ on its left and another right cluster $\mathcal{R}\backslash\{c_l\}$ with r-1 cells. If the displacement curve d_l of c_l is of type A, the lemma is clearly correct for $\mathcal{R}$ because adding two piecewise linear, convex, and nondecreasing curves will result in the same properties. If d_l is of type C, then its slops are

$$d'_l = \begin{cases} 0, & x_t < a \\ -m_l, & a < x_t < c \\ m_l, & c < x_t \end{cases} \tag{3}$$

where m_l is the weight on the displacement of c_l. Since all cells in $\mathcal{R}$ are originally placed at optimal positions, the slope of the displacement curve $d_{\mathcal{R}}$ of the cluster $\mathcal{R}\backslash\{c_l\}$ satisfies

$$\begin{cases} d'_{\mathcal{R}} = 0, & x_t < a \\ d'_{\mathcal{R}} \geq m_l, & a < x_t \end{cases} \tag{4}$$

because, otherwise, the whole right cluster $\mathcal{R}$ should have been moved further to the right. Hence, adding up d_l and $d_{\mathcal{R}}$ gives a piecewise linear, nondecreasing, and convex shape curve.

Algorithm 2 AcCurve

Require: $points = \{(x_b, k_b^l, k_b^r) \mid b \text{ is a breakpoint}\}$.
Ensure: Total displacement curve.
1: Sort *points* by x_b;
2: **for all** $x_{b_1} = x_{b_2}$ **do**
3: $k_{b_1}^l \leftarrow k_{b_1}^l + k_{b_2}^l$;
4: $k_{b_1}^r \leftarrow k_{b_1}^r + k_{b_2}^r$;
5: Remove b_2 from *points*.
6: **end for**
7: **for all** breakpoint b **do**
8: $\bar{k}_b^l \leftarrow \sum_{x_i<x_b} k_i^r + \sum_{x_i \geq x_b} k_i^l$;
9: $\bar{k}_b^r \leftarrow \sum_{x_i \leq x_b} k_i^r + \sum_{x_i > x_b} k_i^l$;
10: **end for**

Therefore, the lemma is correct for any right cluster of r cells, which accomplishes the proof. ∎

Theorem 1 states that the final displacement curve is convex if the local cells are originally at their optimal positions w.r.t. their GP positions, before the target cell is inserted.

Theorem 1: Consider a window W containing a target position (x_t', y_t'), a set $\mathcal{S}$ of local cells lying completely inside W, and a target cell c_t to be inserted into W. If all the cells in $\mathcal{S}$ are originally placed at their optimal positions (i.e., total displacement is the smallest under the fixed row and fixed-order constraint) w.r.t. their GP positions, the displacement curve, where the x-axis is the position of the target cell x_t, obtained by adding up the displacement curves of all the cells in $\mathcal{S}$ is piecewise linear and convex.

Proof of Theorem 1: The total displacement curve is clearly piecewise linear because the displacement curve for each cell is piecewise linear and $|\mathcal{S}|$ is finite. The right (left) part of the displacement curve is accumulated by a set of right clusters so it is nondecreasing (nonincreasing) and convex. Therefore, the total displacement curve is piecewise linear and convex. ∎

The precondition of having the local cells at optimal positions w.r.t. their GP positions would require running an MCF (as described in Section III-C) before invoking MGL that will lengthen the running time. Therefore, in our implementation, we compute the cost at each breakpoint to find the optimal position. Since the number of breakpoints is linear with the number of local cells, the optimal positions can be found in linear time, as shown in Algorithm 2. Given a set of breakpoints sorted by x-coordinate (line 1), we first merge breakpoints sharing the same x-coordinate by adding the slopes together (lines 2–6). For the total displacement curve, its left (right) slope at each breakpoint is the right slope sum of left (nonright) breakpoints adding the left slope sum of nonleft (right) breakpoints. Then, the optimal x-coordinate x_{b_0} is at the boundary of its feasible region or satisfies $\bar{k}_{b_0}^l \bar{k}_{b_0}^r \leq 0$.

B. Maximum Displacement Optimization

In this section, we will present a maximum displacement optimization method in a legal placement. Recall that in MGL, each cell is processed sequentially and it will then be fixed to a row once placed. The maximum displacement can be further reduced if the row assignments can be changed, especially for the cells being placed near the end of MGL. It is unavoidable to place them with large displacements if the regions around their GP locations are dense. Fig. 6(a) shows the displacement of a cell type in a fence region. Each rectangle represents a cell. Red cells are of the same type and gray cells are of other types. The long gray lines connect cells to their corresponding GP positions. Some cells are placed to even tens of rows away from their GP positions.

Algorithm 3 Maximum Displacement Optimization

Require: Cell list $\mathcal{C}$.
Ensure: Legal positions of $\mathcal{C}$.
1: **while** *true* **do**
2: $c_t \leftarrow$ cell with max displacement;
3: $\mathcal{C}_T \leftarrow$ cell list with the same type and fence;
4: **if** $|\mathcal{C}_T| > s_0$ **then**
5: Sort $\mathcal{C}_T$ by $\left|x_i - \frac{x_t+x_t'}{2}\right| + \left|y_i - \frac{y_t+y_t'}{2}\right|$;
6: $\mathcal{C}_T \leftarrow$ top-s of $\mathcal{C}_T$;
7: **if** $c_t \notin \mathcal{C}_T$ **then**
8: $\mathcal{C}_T \leftarrow \mathcal{C}_T \cup \{c_t\}$;
9: **end if**
10: **end if**
11: Optimize $\mathcal{C}_T$ by bipartite matching;
12: **if** max displacement of $\mathcal{C}_T$ is not changed **then**
13: **return** ;
14: **end if**
15: **end while**

To reduce the maximum displacement without introducing violations to the legal placement, we perform an iterative min-cost bipartite matching to optimize the maximum displacement. For each iteration, we regard the cell c_t with the largest displacement as the target cell and reduce its displacement with the help of cells close to the current and GP positions of the target cell. Algorithm 3 shows the flow of the optimization. Cell list $\mathcal{C}_T \subseteq \mathcal{C}$ is initialized by cells with the same type and fence region as c_t (line 3). If $|\mathcal{C}_T|$ is larger than a given number s_0, only c_t and the cells close to $([(x_t + x_t')/2], [(y_t + y_t')/2])$, the middle point of the current and GP positions of c_t, are selected for fast running time (lines 4–10). Given a bipartite graph $G = (\mathcal{C}_T, \mathcal{P}_T, \mathcal{C}_T \times \mathcal{P}_T)$ on the current positions $\mathcal{P}_T \subseteq \mathcal{P}$ of the cells in $\mathcal{P}_T$, any cell $c_i \in \mathcal{C}_T$ can take up the positions $p_j = (x_j, y_j)$ of another cell c_j to minimize the maximum displacement without creating any violations. The problem is to find a perfect matching $\mathcal{S} \subseteq \mathcal{C}_T \times \mathcal{P}_T$ between cells and positions with the minimum total cost $\sum_{(c_i,p_j)\in\mathcal{S}} D_{i,j}$, where $D_{i,j} = \phi(|x_j - x_i'| + |y_j - y_i'|)$ and $\phi(\delta)$ is defined as a strictly increasing function such that it is linear when δ is small to preserve the average displacement. After a certain threshold of δ, ϕ will increase rapidly in order to discourage large displacement. Here, we have

$$\phi(\delta) = \begin{cases} \delta, & \delta \leq \delta_0 \\ \frac{\delta^5}{\delta_0^4}, & \text{otherwise} \end{cases} \tag{5}$$

where δ_0 is the tolerable maximum displacement threshold. This min-cost perfect matching problem

can be optimally solved by formulating as an MCF problem [25].

There are previous works that use bipartite matching in detailed placement to minimize HPWL but only those cells on "independent" nets can be optimized simultaneously [26], [27]. Here, the cost function ϕ is defined in such a way that both the average and maximum displacement are handled and all selected cells can be optimized simultaneously.

C. Fixed Row and Fixed-Order Optimization

After the matching-based maximum displacement optimization, we perform a final postprocessing refinement to further reduce the maximum and average displacement by shifting the cells locally without changing the cell order and row assignments. Taking the objective of the total displacement as an example, given a set of m multirow height cells $\mathcal{C} = \{c_i\}$, the problem can be formulated as follows:

$$\min_{x_i} \quad \sum_i m_i \delta_{xi} \tag{6}$$

$$\text{s.t.} \quad x_i + w_i \le x_j \quad \forall (i,j) \in \mathcal{E} \tag{6a}$$

$$l_i \le x_i \le r_i \quad \forall c_i \in \mathcal{C} \tag{6b}$$

where m_i is the weight on the x-displacement δ_{xi} of cell c_i, w_i is the width of c_i, l_i and r_i are the left and right boundary of the row segment that c_i can horizontally shift inside, and $\mathcal{E}$ is the set of neighboring pairs where $(i,j) \in \mathcal{E}$ if and only if c_i is the left neighbor of c_j on some rows. The left and right boundries of segments can be determined by core boundries, fence regions, or placement blockages.

Equation (6) can be converted to a dual MCF problem and effectively solved [19], [20]. Compared to the formulation in [19], our transformation to MCF has three strengths.

1) There are significantly fewer vertices in the flow network, which is more efficient.
2) The maximum and average displacement are optimized simultaneously.
3) Weight m_i is set according to (2) but it is ignored in [19].

We first split the x-displacement δ_{xi} in (6) to a pair of variables x_i^- and x_i^+

$$\max_{x_i, x_i^-, x_i^+} \sum_i m_i \left(x_i^- - x_i^+\right) \tag{7}$$

$$\text{s.t.} \quad x_i^- \le x_i - x_i' \le x_i^+ \quad \forall c_i \in \mathcal{C} \tag{7a}$$

$$x_i^- \le 0 \le x_i^+ \quad \forall c_i \in \mathcal{C} \tag{7b}$$

$$x_i - x_j \le -w_i \quad \forall (i,j) \in \mathcal{E} \tag{7c}$$

$$x_i \ge l_i \quad \forall c_i \in \mathcal{C}_L \tag{7d}$$

$$x_i \le r_i \quad \forall c_i \in \mathcal{C}_R \tag{7e}$$

where $\mathcal{C}_L$ ($\mathcal{C}_R$) is the set of left-most (right-most) cells in at least one of the segments.

Note that (7) itself is not a dual of an MCF problem. As the dual LP of an MCF problem, each variable in the LP represents a vertex in the MCF and each constraint in the LP represents an edge in the MCF, where the vertices of variables with positive signs are sources of edges, the vertices of variables with negative signs are sinks of edges, and the constant is the cost of edges. To preserve the nature of an edge that consists of a source and a sink, we introduce an auxiliary variable $\tilde{x}_0$ representing the coordinate of the core origin in an auxiliary coordinate system. By substituting the coordinates in the auxiliary coordinate system $\{\tilde{x}_i, \tilde{x}_i^-, \tilde{x}_i^+\}$ for those in the core coordinate system $\{x_i, x_i^-, x_i^+\}$, we have

$$\max_{\tilde{x}_0, \tilde{x}_i, \tilde{x}_i^-, \tilde{x}_i^+} \sum_i m_i \left(\tilde{x}_i^- - \tilde{x}_i^+\right) \tag{8}$$

$$\text{s.t.} \quad \tilde{x}_i^- \le \tilde{x}_i - x_i' \le \tilde{x}_i^+ \quad \forall c_i \in \mathcal{C} \tag{8a}$$

$$\tilde{x}_i^- \le \tilde{x}_0 \le \tilde{x}_i^+ \quad \forall c_i \in \mathcal{C} \tag{8b}$$

$$\tilde{x}_i - \tilde{x}_j \le -w_i \quad \forall (i,j) \in \mathcal{E} \tag{8c}$$

$$\tilde{x}_i - \tilde{x}_0 \ge l_i \quad \forall c_i \in \mathcal{C}_L \tag{8d}$$

$$\tilde{x}_i - \tilde{x}_0 \le r_i \quad \forall c_i \in \mathcal{C}_R \tag{8e}$$

whose constraints have exactly one variable with positive sign and one variable with negative sign. Thus, its dual linear programming is an MCF problem

$$\min_f Q = \sum_i \left(x_i' \left(f_i^+ - f_i^-\right) - l_i f_i^l + r_i f_i^r\right) - \sum_{(i,j) \in \mathcal{E}} w_i f_{ij} \tag{9}$$

$$\text{s.t.} \quad F_i = f_i^+ - f_i^- + f_i^r - f_i^l + \sum_{j:(i,j)\in\mathcal{E}} f_{ij} - \sum_{k:(k,i)\in\mathcal{E}} f_{ki} = 0 \quad \forall c_i \in \mathcal{C} \tag{9a}$$

$$F_0 = -\sum_i F_i = 0 \tag{9b}$$

$$0 \le f_i^+, f_i^- \le m_i \quad \forall c_i \in \mathcal{C} \tag{9c}$$

$$f_i^l, f_i^r \ge 0 \quad \forall c_i \in \mathcal{C} \tag{9d}$$

$$f_i^l = 0 \quad \forall c_i \in \mathcal{C} - \mathcal{C}_L \tag{9e}$$

$$f_i^r = 0 \quad \forall c_i \in \mathcal{C} - \mathcal{C}_R \tag{9f}$$

$$f_{ij} \ge 0 \quad \forall (i,j) \in \mathcal{E} \tag{9g}$$

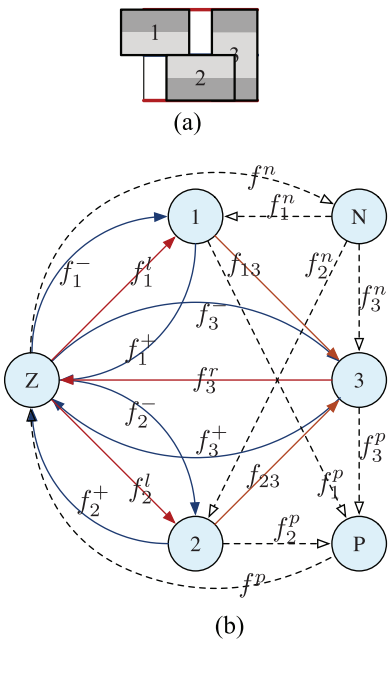
where $\{f\} = \{f_i^-\} \cup \{f_i^+\} \cup \{f_{ij}\} \cup \{f_i^l\} \cup \{f_i^r\}$. After solving the MCF, the coordinates in the auxiliary coordinate system are the potential of vertices and the coordinates in the core coordinate system can be calculated by $x_i = \tilde{x}_i - \tilde{x}_0$, $x_i^- = \tilde{x}_i^- - \tilde{x}_0$, $x_i^+ = \tilde{x}_i^+ - \tilde{x}_0$. Note that each of the auxiliary vertices $\{v_i^-\}$ and $\{v_i^+\}$ connects only two edges, which can be combined to form one edge. Hence, they can be eliminated. Overall, this is an MCF problem with $m+1$ vertices and $2m+|\mathcal{C}_L|+|\mathcal{C}_R|+|\mathcal{E}|$ edges where m is the number of cells, while the MCF in [19] has $3m+2$ vertices and $6m+|\mathcal{E}|$ edges. Our formulation is simpler and thus can be solved more efficiently.

1) Extension Considering Maximum Displacement: The formulation above optimizes the total displacement. To consider the maximum displacement, we further introduce a pair of auxiliary variables δ^- and δ^+ whose absolute values represent the largest displacement of the cells to the left and to the right of the corresponding GP position. Thus, we extend 8 to consider a weighted sum as follows:

$$\max_{\delta^-, \delta^+, \tilde{x}_0, \tilde{x}_i, \tilde{x}_i^-, \tilde{x}_i^+} m_0\left(\delta^- - \delta^+\right) + \sum_i m_i \left(\tilde{x}_i^- - \tilde{x}_i^+\right) \tag{10}$$

$$\text{s.t.} \quad \delta^- \le \tilde{x}_i - x_i' - \delta_{y_i} \quad \forall c_i \in \mathcal{C} \tag{10a}$$

$$\delta^- \le \tilde{x}_0 - \delta_{y_i} \quad \forall c_i \in \mathcal{C} \tag{10b}$$



Flow	Cap	Cost
f_1^-	1/4	1
f_1^+	1/4	-1
f_2^-	1/4	-1
f_2^+	1/4	1
f_3^-	1/2	-3
f_3^+	1/2	3
f^n	1/50	0
f^p	1/50	0
f_1^n	1/50	1
f_1^p	1/50	-1
f_2^n	1/50	-1
f_2^p	1/50	1
f_3^n	1/50	-3
f_3^p	1/50	3
f_{13}	∞	-3
f_{23}	∞	-3
f_1^l	∞	0
f_2^l	∞	0
f_3^r	∞	5

(c)

Fig. 5. Example of fixed row and fixed-order optimization.

$$\delta^+ \geq \tilde{x}_i - x'_i + \delta_{y_i} \quad \forall c_i \in \mathcal{C} \tag{10c}$$

$$\delta^+ \geq \tilde{x}_0 + \delta_{y_i} \quad \forall c_i \in \mathcal{C} \tag{10d}$$

(8a)–(8e)

where m_0 balances the maximum and the average displacement in the objective function and δ_{y_i} is the y-displacement of cell c_i, which are constants since the row assignments will be preserved in this step. The dual LP is as follows:

$$\min_f Q + \left(f^p + f^n\right) \max_i \delta_{y_i} + \sum_i \left(x'_i\left(f_i^p - f_i^n\right) - \delta_{y_i}\left(f_i^p + f_i^n\right)\right) \tag{11}$$

$$\text{s.t.} \quad F_i + f_i^p - f_i^n = 0, f_i^p, f_i^n \geq 0 \quad \forall c_i \in \mathcal{C} \tag{11a}$$

$$F_0 + f^n - f^p = 0, 0 \leq f^p, f^n \leq m_0 \tag{11b}$$

$$f^p - \sum_i f_i^p = f^n - \sum_i f_i^n = 0 \tag{11c}$$

(9c)–(9g)

where f^p, f^n, $\{f_i^p\}$, and $\{f_i^n\}$ are auxiliary variables for handling the maximum displacement.

Fig. 5 shows an example. Cells c_1 and c_2 are single row while cell c_3 is double row. The corresponding flow graph is shown in Fig. 5(b). Vertices v_z, v_n, and v_p are auxiliary nodes while each of the other nodes represents a cell in Fig. 5(a). The solid straight edges from v_z (e.g., f_1^l) represent the flows for the constraints of the left boundary. The solid straight edge to v_z (f_3^r) represents the flow for the constraints of the right boundary. The other solid straight edges (e.g., f_{13}) illustrate the flows for the constraints between neighbouring cells. The solid curly edges (e.g., f_i^-) represent the flows formulating the absolute value. The dotted edges (e.g., f^n) represent the flows for the formulation of the maximum displacement.

To solve the MCF problem, we deploy a network simplex algorithm that performs $O(mnNU)$ pivots and $O(n)$ time per pivot in the worst-case scenario, where m and n denote the number of nodes and arcs in the flow network, respectively, U denotes the maximum arc capacities, and N denotes the largest arc cost [28]. However, this bound does not reflect the typical performance of the algorithm in practice [29]. With the first eligible arc pivot rule, the algorithm can be much faster than this bound.

D. Routability-Driven Refinement

Edge spacing rules define the minimum distances between different types of cells. The method in Section III-B does not create violations to any edge spacing rules because only cells of the same type replace each other in the bipartite matching and all pairs of consecutive cell edges remain unchanged. For the MGL and the fixed row and fixed-order optimization, there are two kinds of cell moves. One is to horizontally shift cells to the left or right with row assignment and cell order fixed, where the edge spacing is reserved between each pair of consecutive cells. The other is to insert a target cell, where the edge spacing is reserved when constructing insertion points.

Pin access and pin short violations are caused by signal pins of cells sheltered by P/G rails or I/O pins so that no same layer wire or upper layer via can access the pins without causing design rule violations (DRVs). Due to limited numbers of lower layer I/O pins and resource reservation for their routing, we treat the regions below I/O pins in M1 or M2 as placement blockages. So the source of violations can be divided into two types: 1) overlaps with horizontal rails or 2) with vertical rails.

In MGL (Section III-A), if an insertion point has a violation with a horizontal rail, it will not be considered as a valid insertion point. On the contrary, violations with vertical rails are handled in a more detailed manner. When enumerating the insertion points in a window, local cells are moved to the left-most and right-most positions in the local region to validate the insertion points. After a local cell being moved to its left-most position, it moves back to the right until it has no violation with a vertical rail; the opposite happens for right-most positions. If the window accommodates no violation-free insertion points for both target cell and local cells, it is discarded and a larger window will be considered in the next iteration until the window covers the whole fence region. While evaluating an insertion point, the optimal position is chosen accordingly to the displacement curve. If there is a violation with vertical rails, a least-displaced-position without violation is selected.

Maximum displacement optimization (Section III-B) does not create pin access or pin short violations. To avoid more pin access and pin short violations in the fixed row and fixed-order optimization (Section III-C), the cells will be restricted to a feasible range defined by the intersection of the row segment and the P/G rails. Thus, every cell has its left and right boundary constraints in the MCF, i.e., $\mathcal{C}_L = \mathcal{C}_R = \mathcal{C}$. l_i

and r_i in the formulation are the left and right boundaries of cell c_i's feasible range, respectively.

E. Multithread Implementation

Since local windows that do not overlap with each other can be processed simultaneously, MGL is implemented with multithreading to speed up processing. A scheduling step decides which local windows can be processed at the same time. The *scheduler* maintains a list $\mathcal{L}_p$ containing 2-tuples of the target cell and its corresponding local window under processing. In each iteration, the scheduler selects a fixed number of these 2-tuples that do not have their local windows overlapping with each other and pushes them into $\mathcal{L}_p$.

Legalizers in the child threads process MGL whenever there is unprocessed 2-tuple in $\mathcal{L}_p$ and apply the legalization step. The scheduler switches to perform MGL as well when it finished scheduling. If MGL failed to insert the target cell, the local window is expanded and the new 2-tuple of the target cell and the expanded local window will then be pushed into a waiting list $\mathcal{L}_w$ from which the scheduler will select the 2-tuples for the next iteration. Since the scheduler synchronizes all threads and the cell order in $\mathcal{L}_w$ is kept the same as the original cell order, the multithread implementation is deterministic once the capacity of list $\mathcal{L}_p$ is determined.

IV. Experimental Results

We implemented the proposed legalization algorithm for mixed-cell-height circuits in C++ programming language. LEMON [25] is used as the MCF solver. We run all experiments on a 64-bit Linux machine with eight cores of Intel Xeon 2.1-GHz CPUs and 64-GB RAM.

A. ICCAD 2017 Benchmark Results

In the first experiment, we compare with a binary from the first place of the ICCAD 2017 Contest [9] and a binary from the state of the art [17]. Eight threads are used for the proposed algorithm and both comparison algorithms to simulate the Contest environment. We do not compare with [30] because the average displacement in its objective function does not consider the weights on cell numbers as (2) and thus, the results are biased toward single-row cells since, shown in Table I, at least 85% cells are single-row cells. Some other benchmark statistics are also shown in Table I, including design density and GP HPWL. Here, *density* is measured by the total cell area over the total free area.

We evaluate the pin access violations with Innovus 18.12 taking both *Preroute DRC Violations* and *Pin Access Violations* into count. We adopt the score function in the contest as much as possible to have a more comprehensive comparison:

$$S = \left(1 + S_{\text{hpwl}} + \frac{N_p + N_e}{m}\right)\left(1 + \frac{\max_i\{\delta_i\}}{\Delta}\right)S_{\text{am}} \quad (12)$$

TABLE I
Statistics of ICCAD 2017 Benchmarks

Benchmark	# Fixed Macros	# Cells of Different Heights 1	2	3	4	Density	HPWL (+e9)
des_perf_1	0	112644	0	0	0	90.6%	1.22
des_perf_a_md1	4	103589	4699	0	0	55.1%	2.16
des_perf_a_md2	4	105030	1086	1086	1086	55.9%	2.18
des_perf_b_md1	0	106782	5862	0	0	55.0%	2.11
des_perf_b_md2	0	101908	6781	2260	1695	64.7%	2.14
edit_dist_1_md1	0	118005	7994	2664	1998	67.4%	4.01
edit_dist_a_md2	6	115066	7799	2599	1949	59.4%	5.10
edit_dist_a_md3	6	119616	2599	2599	2599	57.2%	5.33
fft_2_md2	0	28930	2117	705	529	82.7%	0.45
fft_a_md2	6	27431	2018	672	504	32.3%	1.09
fft_a_md3	6	28609	672	672	672	31.2%	0.95
pci_bridge32_a_md1	4	26680	1792	597	448	49.5%	0.45
pci_bridge32_a_md2	4	25239	2090	1194	994	57.7%	0.57
pci_bridge32_b_md1	6	26134	1756	585	439	26.6%	0.66
pci_bridge32_b_md2	6	28038	292	292	292	18.3%	0.58
pci_bridge32_b_md3	6	27452	292	585	585	22.2%	0.58

where S_{hpwl} is the ratio of HPWL increase, N_p and N_e are the numbers of violations on pin access/short and edge spacing, m is the number of cells, displacements δ_i and S_{am} are calculated by (1) and (2), and Δ is 100 [9]. Note that the runtime scores are not included because they are measured w.r.t other teams. The penalties of maximum displacement and target utilization are not included because their exact definitions are not clear and a constant factor is not revealed. The results are listed in Table II where our proposed algorithm achieves 35% smaller average displacement and 13% shorter maximum displacement compared with the first place. Our proposed algorithm also achieves 8% smaller average displacement and 4% longer maximum displacement compared with the state-of-the-art work. For routability-driven constraints, we have no edge spacing violations while the first place produces nearly six thousand in one case. We also have significantly fewer pin access violations. Without counting the designs that we are violation free, the numbers of violations are reduced by 398 times and 446 times on average, compared with the first place and the state of the art, respectively. In terms of S, our purposed method has 47% and 9% improvement on average.

B. Modified ISPD 2015 Benchmark Results

In the second experiment, we compare with the state-of-the-art placers [15], [16], [18]. The benchmarks are modified from the ISPD 2015 Contest [23] and provided by Chow *et al.* [18]. 10% of the cells were selected and converted to double height and half width. To be consistent with other works listed in Table III, we adapted our program to use total displacement as the objective function and ignored fences as well as routability-driven constraints. Note that the results of [15], [18] are improved ones reported in [16]. We can see that we have improved over the previous published works by 19%, 16%, and 8%, respectively, in total displacement.

C. Tradeoff on Window Size

In the third experiment, we verify the tradeoff between running time and solution quality on the expansion step of window size. As shown in Table IV, a larger window size is beneficial on average displacement while a smaller window

TABLE II
COMPARISON BETWEEN OUR ALGORITHM AND THE CHAMPION IN ICCAD 2017

Benchmark	Avg. Disp.			Max. Disp.			HPWL (+e9)			Pin Access			Edge Space			Score S			Runtime (s)		
	1st	[17]	Ours	1st	[17]	Ours	1st	[17]	Ours	1st	[17]	Ours	1st	[17]	Ours	1st	[17]	Ours	1st	[17]	Ours
des_perf_1	0.710	0.807	0.870	7.7	6.6	10.1	1.30	1.33	1.34	7313	3636	216	0	0	0	0.87	0.96	1.05	9.55	22.64	28.91
des_perf_a_md1	1.818	0.994	0.913	62.6	60.7	60.7	2.26	2.23	2.23	2566	3080	2	0	0	0	3.16	1.69	1.51	5.46	3.80	3.85
des_perf_a_md2	3.476	1.328	1.143	68.0	40.4	48.1	2.28	2.26	2.25	2604	2986	2	0	0	0	6.26	1.98	1.75	5.50	6.27	3.84
des_perf_b_md1	0.698	0.701	0.659	9.0	9.0	11.4	2.16	2.16	2.16	2442	2392	17	0	0	0	0.80	0.80	0.75	4.47	3.28	4.34
des_perf_b_md2	0.655	0.655	0.617	20.0	19.4	24.1	2.19	2.20	2.19	2125	2120	0	0	0	0	0.82	0.82	0.78	4.16	2.40	3.80
edit_dist_1_md1	0.798	0.765	0.660	7.9	7.8	5.8	4.09	4.10	4.09	3435	3414	0	0	0	0	0.90	0.87	0.71	5.17	3.10	3.89
edit_dist_a_md2	0.646	0.639	0.612	16.4	16.4	16.4	5.18	5.18	5.17	3230	3235	0	0	0	0	0.78	0.77	0.72	4.31	3.71	4.29
edit_dist_a_md3	0.901	0.838	0.760	28.0	23.3	23.3	5.46	5.48	5.46	3479	3361	194	0	0	0	1.21	1.09	0.96	44.34	25.66	16.73
fft_2_md2	0.675	0.837	0.716	6.6	7.2	6.3	0.49	0.52	0.51	1942	932	75	5980	0	0	0.96	1.07	0.86	1.18	0.90	1.40
fft_a_md2	0.566	0.568	0.562	34.3	34.3	34.3	1.11	1.11	1.11	807	814	11	0	0	0	0.79	0.79	0.77	0.99	0.43	0.87
fft_a_md3	0.536	0.538	0.529	11.0	11.0	11.0	0.97	0.97	0.97	764	765	5	0	0	0	0.62	0.62	0.60	1.02	0.46	0.80
pci_bridge32_a_md1	0.696	0.664	0.650	42.6	42.6	45.7	0.47	0.47	0.48	588	564	1	0	0	0	1.05	1.00	0.99	1.11	0.89	1.07
pci_bridge32_a_md2	0.898	0.894	0.832	27.2	18.1	18.1	0.59	0.60	0.60	1845	813	18	0	0	0	1.27	1.15	1.05	1.89	3.02	1.30
pci_bridge32_b_md1	1.064	0.824	0.776	87.7	51.4	51.4	0.69	0.68	0.68	587	753	0	0	0	0	2.11	1.32	1.21	1.26	0.63	1.12
pci_bridge32_b_md2	1.084	0.791	0.698	72.3	54.6	61.7	0.60	0.59	0.59	646	862	0	0	0	0	1.97	1.30	1.17	1.08	0.65	1.23
pci_bridge32_b_md3	1.910	1.024	0.916	68.2	49.8	49.8	0.62	0.61	0.61	656	849	1	0	0	0	3.46	1.65	1.43	1.34	1.09	1.04
Norm. Avg.	1.35	1.08	**1.00**	1.13	**0.96**	1.00	1.02	1.02	**1.00**	398.09	445.95	**1.00**		—		1.47	1.09	**1.00**	1.21	**0.94**	1.00

TABLE III
COMPARISON BETWEEN OUR ALGORITHM AND STATE-OF-THE-ART PLACERS

Benchmark	#Cell	Density	Total Disp. (sites)				Runtime (s)			
			[18]-Imp	[15]	[16]	Ours	[18]-Imp	[15]	[16]	Ours
des_perf_1	112644	90.58%	279545	474789	242622	188719	6.1	7.5	2.4	2.7
des_perf_a	108292	42.90%	81452	73057	72561	71049	2.5	3.8	2.3	2.1
des_perf_b	112644	49.71%	81540	72429	71888	70959	2.2	3.9	2.3	2.3
edit_dist_a	127419	45.54%	59814	60971	62961	57264	1.8	4.9	2.8	2.3
fft_1	32281	83.55%	54501	53389	46121	38938	1.0	1.3	0.7	1.3
fft_2	32281	49.97%	25697	21018	20979	20381	0.4	1.1	0.6	0.6
fft_a	30631	25.09%	19613	18150	18304	17897	0.2	1.2	0.6	0.6
fft_b	30631	28.19%	28461	21234	21671	20852	0.4	1.2	0.6	0.6
matrix_mult_1	155325	80.24%	80235	73682	71793	61992	4.0	5.4	3.6	3.2
matrix_mult_2	155325	79.03%	75810	65959	65876	58250	4.2	5.4	3.7	3.1
matrix_mult_a	149655	41.95%	46001	40736	40298	39683	1.6	5.7	3.4	3.0
matrix_mult_b	146442	30.90%	40059	37243	37215	36658	1.2	5.6	3.2	3.2
matrix_mult_c	146442	30.83%	42490	40942	40710	39767	1.4	5.6	3.2	2.7
pci_bridge32_a	29521	38.39%	27832	26674	26289	25960	0.3	1.2	0.6	0.4
pci_bridge32_b	28920	14.30%	27864	26160	26028	26120	0.2	1.0	0.4	0.6
superblue11_a	927074	42.92%	1786342	1983090	1742941	1595907	29.7	50.3	26.3	22.2
superblue12	1287037	44.72%	2015678	1995140	1963403	1713915	103.6	56.5	38.6	31.6
superblue14	612583	55.78%	1599810	1497490	1566966	1330885	16.7	48.1	17.7	13.8
superblue16_a	680869	47.85%	1173106	1147530	1135186	1055668	20.7	41.8	18.7	14.9
superblue19	506383	52.33%	806529	808164	781928	705509	10.5	29.6	13.2	11.9
Norm. Avg.			1.19	1.16	1.08	**1.00**	1.04	2.09	1.08	**1.00**

size causes 2% degradation. On the other hand, performing MGL takes less time with a smaller window in most of the designs, except for dense regions where smaller windows need more expansion iterations. Therefore, this parameter is left viable for users to adjust based on site size and aspect ratio, cell size, core utilization, etc.

D. Efficiency of Multithreading

In the fourth experiment, we verify the efficiency of multithreaded MGL. Since the solutions are identical with different numbers of threads and multithreading is only implemented for MGL, we only show the running time of MGL in Table V, where 1.6×, 2.9×, and 4.5× speedups are achieved with two, four, and eight threads, respectively.

E. Effectiveness and Runtime Breakdown

In the fifth experiment, we break down the effectiveness and running time of MGL and two postprocessing stages. Table VI lists the average and maximum displacement and running timing for these three stages. We can see that the maximum displacement decreases by 66% through maximum displacement optimization and the average displacement decreases by 1% through fixed row and fixed-order optimization. The table also shows that each of the three stages takes about one third of

TABLE IV
TRADEOFF ON WINDOW SIZE

Benchmark	Avg. Disp.			Max. Disp.			Running Time		
	S*	M†	L‡	S*	M†	L‡	S*	M†	L‡
des_perf_1	0.962	0.964	0.968	48.4	49.5	69.8	3.17	4.74	8.34
des_perf_a_md1	0.937	0.919	0.940	92.7	60.7	60.7	0.90	1.81	3.74
des_perf_a_md2	1.237	1.148	1.142	49.7	48.1	48.1	0.77	1.67	3.23
des_perf_b_md1	0.677	0.675	0.676	52.4	53.8	47.1	0.82	1.28	2.14
des_perf_b_md2	0.626	0.618	0.617	37.1	27.3	26.5	0.71	1.31	2.62
edit_dist_1_md1	0.674	0.664	0.656	13.9	5.8	7.3	0.58	0.98	1.93
edit_dist_a_md2	0.620	0.614	0.613	24.1	16.5	16.4	0.71	1.30	2.77
edit_dist_a_md3	0.797	0.783	0.771	56.1	74.2	83.7	1.94	2.78	5.47
fft_2_md2	0.738	0.721	0.722	21.4	10.0	13.0	0.19	0.29	1.47
fft_a_md2	0.562	0.563	0.566	34.3	34.3	34.3	0.15	0.22	0.35
fft_a_md3	0.533	0.531	0.533	11.0	11.0	11.0	0.10	0.15	0.30
pci_bridge32_a_md1	0.655	0.652	0.653	45.8	45.7	45.7	0.23	0.33	1.22
pci_bridge32_a_md2	0.861	0.839	0.829	38.3	18.7	18.1	0.30	0.47	1.26
pci_bridge32_b_md1	0.796	0.781	0.778	54.2	51.4	51.4	6.72	0.31	0.45
pci_bridge32_b_md2	0.745	0.704	0.698	72.3	61.7	61.7	0.16	0.32	0.65
pci_bridge32_b_md3	0.966	0.925	0.879	49.8	49.8	49.8	0.17	0.34	0.71
Norm. Avg.	1.02	1.00	**1.00**	1.31	**1.00**	1.06	1.92	**1.00**	2.26

* Window expands horizontally (vertically) by four (two) rows for each iteration.
† Window expands horizontally (vertically) by six (four) rows for each iteration.
‡ Window expands horizontally (vertically) by eight (six) rows for each iteration.

TABLE V
RUNNING TIME OF MULTITHREADED MGL

Benchmark	MGL Running Time (s)			
	1T	2T	4T	8T
des_perf_1	15.66	9.16	6.12	4.74
des_perf_a_md1	9.61	5.29	2.97	1.81
des_perf_a_md2	8.52	4.74	2.62	1.67
des_perf_b_md1	6.06	3.44	2.04	1.28
des_perf_b_md2	7.01	3.99	2.19	1.31
edit_dist_1_md1	5.14	3.14	1.64	0.98
edit_dist_a_md2	7.35	4.23	2.36	1.30
edit_dist_a_md3	10.55	6.76	3.88	2.78
fft_2_md2	1.44	0.87	0.46	0.29
fft_a_md2	0.91	0.55	0.32	0.22
fft_a_md3	0.85	0.49	0.24	0.15
pci_bridge32_a_md1	1.63	1.08	0.55	0.33
pci_bridge32_a_md2	1.79	1.23	0.70	0.47
pci_bridge32_b_md1	1.15	0.80	0.46	0.31
pci_bridge32_b_md2	1.32	0.86	0.48	0.32
pci_bridge32_b_md3	1.38	0.88	0.48	0.34
Norm. Avg.	1.00	0.61	0.34	0.22

the total running time on average but the maximum displacement optimization dominates in two congested designs. Fig. 6 shows an example of the maximum displacement optimization in which red cells are of the same type while red lines connect cells to their corresponding GP positions. Cells with large displacement in Fig. 6(a) are moved to closer locations in Fig. 6(b).

F. Effectiveness of Pin Access Refinement

In the sixth experiment, we evaluate the effectiveness of pin access refinement techniques. Table VII lists the average and maximum displacement and the numbers of inaccessible pins with the techniques turning on and off. We can see that the average and maximum displacement increase by 4% and 5% through pin access refinement but numbers of inaccessible pins reduce by more than a thousand times on average. In addition, we route the placement results with CUGR [31] and Dr. CU 2.0 [32], and evaluate the actual impact of pin

TABLE VI
EFFECTIVENESS AND RUNTIME BREAKDOWN

Benchmark	Avg. Disp.			Max. Disp.			Running Time		
	L*	M*	F*	L*	M*	F*	L*	M*	F*
des_perf_1	0.964	0.915	0.870	49.5	10.0	10.1	4.74	20.51	1.65
des_perf_a_md1	0.919	0.919	0.913	60.7	60.7	60.7	1.81	0.17	1.18
des_perf_a_md2	1.148	1.148	1.143	48.1	48.1	48.1	1.67	0.27	1.17
des_perf_b_md1	0.675	0.662	0.659	53.8	11.4	11.4	1.28	0.95	1.27
des_perf_b_md2	0.618	0.619	0.617	27.3	24.1	24.1	1.31	0.35	1.43
edit_dist_1_md1	0.664	0.664	0.660	5.8	5.7	5.8	0.98	0.46	1.60
edit_dist_a_md2	0.614	0.614	0.612	16.5	16.4	16.4	1.30	0.52	1.60
edit_dist_a_md3	0.783	0.762	0.760	74.2	23.3	23.3	2.78	11.46	1.67
fft_2_md2	0.721	0.721	0.716	10.0	6.5	6.3	0.29	0.77	0.35
fft_a_md2	0.563	0.563	0.562	34.3	34.3	34.3	0.22	0.20	0.32
fft_a_md3	0.531	0.531	0.529	11.0	11.0	11.0	0.15	0.10	0.28
pci_bridge32_a_md1	0.652	0.652	0.650	45.7	45.7	45.7	0.33	0.47	0.30
pci_bridge32_a_md2	0.839	0.834	0.832	18.7	18.1	18.1	0.47	0.43	0.29
pci_bridge32_b_md1	0.781	0.781	0.776	51.4	51.4	51.4	0.31	0.38	0.29
pci_bridge32_b_md2	0.704	0.704	0.698	61.7	61.7	61.7	0.32	0.48	0.27
pci_bridge32_b_md3	0.925	0.925	0.916	49.8	49.8	49.8	0.34	0.24	0.28
Norm. Avg.	1.01	1.01	**1.00**	1.66	1.00	**1.00**	0.29	0.29	0.28

* L: multi-row global legalization; M: max. disp. opt.; F: fixed row & order opt.

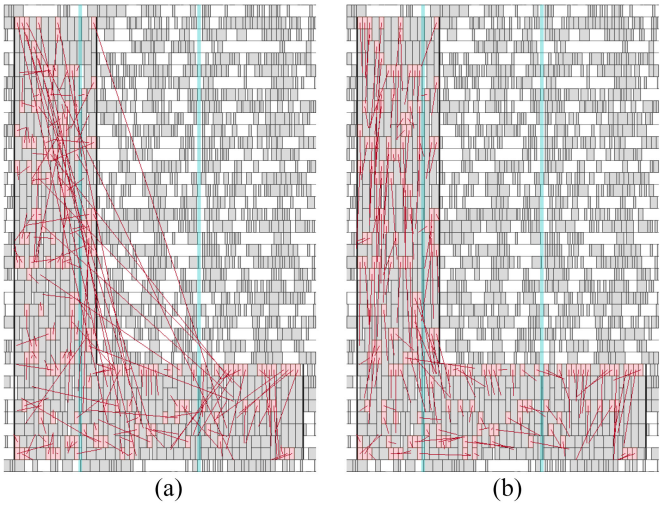


Fig. 6. Maximum displacement optimization. (a) Before. (b) After.

TABLE VII
EFFECTIVENESS OF PIN ACCESS REFINEMENT

Benchmark	Avg. Disp.		Max. Disp.		Pin Access		DRV	
	Off	On	Off	On	Off	On	Off	On
des_perf_1	0.654	0.870	3.7	10.1	7179	216	272584	295232
des_perf_a_md1	0.889	0.913	60.7	60.7	6429	2	97534	88087
des_perf_a_md2	1.113	1.143	48.1	48.1	6396	2	87571	73217
des_perf_b_md1	0.611	0.659	11.6	11.4	6257	17	63624	53696
des_perf_b_md2	0.599	0.617	24.5	24.1	6517	0	56347	45918
edit_dist_1_md1	0.638	0.660	5.8	5.8	7863	0	124029	110427
edit_dist_a_md2	0.590	0.612	16.4	16.4	7979	0	1884114	1857284
edit_dist_a_md3	0.710	0.760	23.3	23.3	8169	194	2176311	2183820
fft_2_md2	0.657	0.716	5.6	6.3	2171	75	17079	12814
fft_a_md2	0.544	0.562	34.3	34.3	2360	11	59078	57282
fft_a_md3	0.511	0.529	11.0	11.0	1928	5	41389	37625
pci_bridge32_a_md1	0.634	0.650	45.7	45.7	1712	1	19566	17029
pci_bridge32_a_md2	0.796	0.832	18.1	18.1	1826	18	19750	16713
pci_bridge32_b_md1	0.767	0.776	51.4	51.4	1676	0	20273	18122
pci_bridge32_b_md2	0.689	0.698	61.7	61.7	1710	0	19964	17759
pci_bridge32_b_md3	0.885	0.916	49.8	49.8	1780	1	18658	16440
Norm. Avg.	**0.95**	1.00	**0.96**	1.00	1007	**1**	1.14	**1.00**

access refinement on routing with the total number of DRVs. With refinement techniques, the total numbers of DRVs are reduced by 14% on average. The improvement is more significant in designs with fewer DRVs. For other congest designs, the routers have trouble eliminating DRVs.

V. Conclusion

We presented a legalization method for mixed-cell-height circuits with consideration of routability constraints, such as pin access, pin short, edge spacing, and fence regions. We proposed an MGL that minimizes the total displacement of the cells within a window toward their given GP positions. We formulated and solved the maximum displacement optimization into by an MCF. Finally, we formulated the fixed row and fixed-order optimization problem with a weighted sum of the maximum and average displacement as objective into another MCF problem for further optimization. Comparing with the champion of the ICCAD 2017 Contest [9], we achieved 35% less average displacement, 13% less maximum displacement, and much fewer routability-driven violations. We also compared with the state-of-the-art work and achieved an 8% average displacement improvement.

Acknowledgment

The authors would like to thank Prof. Jianli Chen for helpful discussions and binary sharing of [16] and [17].

References

[1] H. Li, W.-K. Chow, G. Chen, E. F.Y. Young, and B. Yu, "Routability-driven and fence-aware legalization for mixed-cell-height circuits," in *Proc. 55th Annu. Design Autom. Conf. (DAC)*, San Francisco, CA, USA, 2018, pp. 1–6.

[2] H. Li *et al.*, "Deep learning analysis for split manufactured layouts with routing perturbation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, early access, Nov. 11, 2020, doi: 10.1109/TCAD.2020.3037297.

[3] S. M. Y. Sherazi *et al.*, "CFET standard-cell design down to 3Track height for node 3nm and below," in *Proc. SPIECFET Design Process Technol.Co-Optim. Manuf. XIII*, vol. 10962, 2019, Art. no. 1096206.

[4] S.-H. Baek, H.-Y. Kim, Y.-K. Lee, D.-Y. Jin, S.-C. Park, and J.-D. Cho, "Ultra-high density standard cell library using multi-height cell structure," in *Proc. SPIE Smart Struct. Devices Syst.*, vol. 7268, 2008, Art. no. 72680C.

[5] C. Santos, R. Reis, G. Godoi, M. Barros, and F. Duarte, "Multi-bit flip-flop usage impact on physical synthesis," in *Proc. 25th Symp. Integr. Circuits Syst. Design (SBCCI)*, 2012, pp. 1–6.

[6] Y.-X. Chiang *et al.*, "Designing and benchmarking of double-row height standard cells," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Hong Kong, China, 2018, pp. 64–69.

[7] D. D. Sherlekar, "Cell architecture for increasing transistor size," U.S. Patent 8 631 374, Jan. 2014.

[8] F. Gessler, P. Brisk, and M. Stojilovic, "A shared-memory parallel implementation of the RePlAce global cell placer," in *Proc. 33rd Int. Conf. VLSI Design 19th Int. Conf. Embedded Syst.*, 2020, pp. 78–83.

[9] N. K. Darav, I. S. Bustany, A. Kennings, and R. Mamidi, "ICCAD-2017 CAD contest in multi-deck standard cell legalization and benchmarks," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Irvine, CA, USA, 2017, pp. 867–871.

[10] M. Khasawneh and P. H. Madden, "Hill climbing with trees: Detail placement for large windows," in *Proc. Int. Symp. Phys. Design (ISPD)*, 2020, pp. 9–16.

[11] G. Wu and C. Chu, "Detailed placement algorithm for VLSI design with double-row height standard cells," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 9, pp. 1569–1573, Sep. 2016.

[12] S. A. Dobre, A. B. Kahng, and J. Li, "Design implementation with non-integer multiple-height cells for improved design quality in advanced nodes," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 4, pp. 855–868, Apr. 2018.

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

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

[15] C.-H. Wang *et al.*, "An effective legalization algorithm for mixed-cell-height standard cells," in *Proc. 22nd Asia South Pac. Design Autom. Conf. (ASP-DAC)*, Chiba, Japan, 2017, pp. 450–455.

[16] J. Chen, Z. Zhu, W. Zhu, and Y.-W. Chang, "Toward optimal legalization for mixed-cell-height circuit designs," in *Proc. 54th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Austin, TX, USA, 2017, pp. 1–6.

[17] Z. Zhu, J. Chen, W. Zhu, and Y.-W. Chang, "Mixed-cell-height legalization considering technology and region constraints," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 12, pp. 5128–5141, Dec. 2020.

[18] W.-K. Chow, C.-W. Pui, and E. F.Y. Young, "Legalization algorithm for multiple-row height standard cell design," in *Proc. 53nd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Austin, TX, USA, 2016, pp. 1–6.

[19] Y. Lin *et al.*, "MrDP: Multiple-row detailed placement of heterogeneous-sized cells for advanced nodes," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 6, pp. 1237–1250, Jun. 2018.

[20] J. Vygen, "Algorithms for detailed placement of standard cells," in *Proc. Design Autom. Test Eur. (DATE)*, Paris, France, 1998, pp. 321–324.

[21] S. I. Heo, A. B. Kahng, M. Kim, L. Wang, and C. Yang, "Detailed placement for IR drop mitigation by power staple insertion in sub-10nm VLSI," in *Proc. IEEE Design Autom. Test Eur. Conf. Exhibit. (DATE)*, Florence, Italy, 2019, pp. 830–835.

[22] Z. Zhu *et al.*, "Mixed-cell-height legalization considering complex minimum width constraints and half-row fragmentation effect," *Integr. VLSI J.*, vol. 71, pp. 1–10, Mar. 2020.

[23] I. S. Bustany, D. Chinnery, J. R. Shinnerl, and V. Yutsis, "ISPD 2015 benchmarks with fence regions and routing blockages for detailed-routing-driven placement," in *Proc. Int. Symp. Phys. Design (ISPD)*, 2015, pp. 157–164.

[24] V. Yutsis, I. S. Bustany, D. Chinnery, J. R. Shinnerl, and W.-H. Liu, "ISPD 2014 benchmarks with sub-45nm technology rules for detailed-routing-driven placement," in *Proc. Int. Symp. Phys. Design (ISPD)*, 2014, pp. 161–168.

[25] P. Kovács, "Minimum-cost flow algorithms: An experimental evaluation," *Optim. Methods Softw.*, vol. 30, no. 1, pp. 94–127, 2015.

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

[27] G. Chen *et al.*, "RippleFPGA: Routability-driven simultaneous packing and placement for modern FPGAs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 10, pp. 2022–2035, Oct. 2018.

[28] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, "Chapter IV network flows," in *Optimization* (Handbooks in Operations Research and Management Science), vol. 1. Amsterdam, The Netherlands: Elsevier, 1989, pp. 211–369. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0927050789010054

[29] Z. Király and P. Kovács, "Efficient implementations of minimum-cost flow algorithms," 2012. [Online]. Available: arXiv:1207.6381.

[30] X. Li, J. Chen, W. Zhu, and Y.-W. Chang, "Analytical mixed-cell-height legalization considering average and maximum movement minimization," in *Proc. Int. Symp. Phys. Design (ISPD)*, 2019, pp. 27–34.

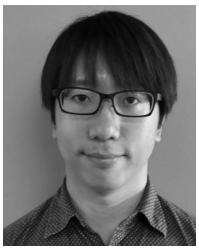
[31] J. Liu, C.-W. Pui, F. Wang, and E. F.Y. Young, "CUGR: Detailed-routability-driven 3D global routing with probabilistic resource model," in *Proc. IEEE 57th Annu. Design Autom. Conf. (DAC)*, San Francisco, CA, USA, 2020, pp. 1–6.

[32] H. Li, G. Chen, B. Jiang, J. Chen, and E. F.Y. Young, "Dr. CU 2.0: A scalable detailed routing framework with correct-by-construction design rule satisfaction," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Westminster, CO, USA, 2019, pp. 1–7.

Haocheng Li received the B.Eng. degree from the School of Information and Communications Engineering, Xi'an Jiaotong University, Xi'an, China, in 2016, and the Ph.D. degree from the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong, in 2020.

He joined Cadence Design System, San Jose, CA, USA, in 2021, where he is currently a Lead Software Engineer, working on routing algorithms with advanced technology nodes. His research interests include electronic design automation, hardware security, and combinatorial optimization.

Dr. Li received the best paper award in ISPD 2017.



Wing-Kai Chow (Member, IEEE) received the B.Sc. degree from Hong Kong Polytechnic University, Hong Kong, in 2009, and the M.Sc. degree in computer science and the Ph.D. degree in computer science and engineering from the Chinese University of Hong Kong, Hong Kong, in 2012 and 2018, respectively.

He joined Cadence Design System, Austin, TX, USA, in 2016, where he is currently a Principal Software Engineer, working on routing algorithms with advanced technology nodes. His current research interest include clock network synthesis, and placement and routing algorithms.

Gengjie Chen received the B.Sc. degree from the Department of Electronic and Communication Engineering, Sun Yat-sen University, Guangzhou, China, in 2015, and the Ph.D. degree from the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong, in 2019.

His research interests include combinatorial optimization, numerical optimization, and electronic design automation.

Dr. Chen received the ACM Outstanding Ph.D. Dissertation Award in Electronic Design Automation in 2020, the first place at ACM Student Research Competition in 2019, the first place at ACM SIGDA Student Research Competition in 2018, the best paper award at ICCAD 2017, the Hong Kong Ph.D. Fellowship from 2015 to 2019, and eight ICCAD/ISPD contest awards.

Bei Yu (Member, IEEE) received the Ph.D. degree from the University of Texas at Austin, Austin, TX, USA, in 2014.

He is currently an Assistant Professor with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong.

Dr. Yu received seven Best Paper Awards from ASPDAC 2021, ICTAI 2019, *Integration, the VLSI Journal* in 2018, ISPD 2017, SPIE Advanced Lithography Conference 2016, ICCAD 2013, ASPDAC 2012, and six ICCAD/ISPD contest awards. He has served as TPC Chair of ACM/IEEE Workshop on Machine Learning for CAD, and in many journal editorial boards and conference committees. He is an Editor of IEEE TCCPS Newsletter.

Evangeline F.Y. Young (Senior Member, IEEE) received the B.Sc. degree in computer science from the Chinese University of Hong Kong (CUHK), Hong Kong, and the Ph.D. degree from the University of Texas at Austin, Austin, TX, USA, in 1999.

She is currently a Professor with the Department of Computer Science and Engineering, CUHK. Her research interests include EDA, optimization, algorithms, and AI. Her research focuses on floorplanning, placement, routing, DFM, and EDA on physical design in general.

Dr. Young research group has won best paper awards from ICCAD 2017, ISPD 2017, SLIP 2017, and FCCM 2018, and several championships and prizes in renown EDA contests, including the 2018–2020, 2015–2016, and 2012–2013 CAD Contests at ICCAD, DAC 2012, and ISPD 2015–2020 and 2010–2011. She has served on the organization committees of ICCAD, ISPD, ARC, and FPT and on the program committees of conferences, including DAC, ICCAD, ISPD, ASP-DAC, SLIP, DATE, and GLSVLSI. She also served on the editorial boards of IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, *ACM Transactions on Design Automation of Electronic Systems*, and *Integration, the VLSI Journal*.