PUFFER: A Routability-Driven Placement Framework via Cell Padding with Multiple Features and Strategy Exploration

Zhijie Cai[1], Peng Zou[1], Zhengtao Wu[2], Xingyu Tong[1], Jun Yu[1], Jianli Chen[1], Yao-Wen Chang[3, 4]
[1]State Key Lab of ASIC & System, Fudan University, Shanghai 200433, China
[2]Shanghai LEDA Technology Co., Ltd., Shanghai 201203, China
[3]Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 10617, Taiwan
[4]Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan
zjcai22@m.fudan.edu.cn; chenjianli@fudan.edu.cn; ywchang@ntu.edu.tw

***Abstract*—Placement is a critical stage in VLSI physical design, especially for routability optimization. Due to the large scale and high integration introduced by the advanced semiconductor manufacturing technology, there remains a significant challenge in routability in the placement stage, which will affect the subsequent routing process. This paper proposes a placement framework, called PUFFER, to optimize routability by cell padding and strategy exploration. The framework first estimates congestion by imitating the behaviors of routing detours and clustered cell spreading. Then it calculates cell padding based on multiple features inspired by the characteristics of convolutional and graph neural networks. Besides, it applies a Bayesian-based method to explore a better placement strategy. Compared with a commercial tool and the state-of-the-art academic RePlAce placer, experiments on industrial benchmarks show that our framework achieves the best routability on average, with a 2.7× speedup over the commercial tool.**

***Index Terms*—physical design, placement, routability, cell padding**

I. Introduction

Placement is a critical stage in VLSI physical design, especially for routability optimization. Placement determines the cell locations and greatly affects net routing paths, thereby impacting the power, performance, and area (PPA) of a circuit. With the advancement of semiconductor manufacturing technology, the routing complexity increases substantially, incurring great challenges to the routability of a circuit design. As a result, it is of great importance to consider routability during placement.

A routability-driven placement algorithm typically consists of a basic placement engine and a routability optimizer. The basic placement engine solves the fundamental placement problems: minimizing wirelength and eliminating cell overlapping. There are two popular analytical placement engines for modern circuit designs, quadratic placement and non-linear placement. In quadratic placement, an iterative framework with lower and upper bounds is often applied. The lower bound is described by a quadratic wirelength model to minimize wirelength, and the upper bound is generated by spreading cells in high-density regions to remove cell overlaps [1]. In non-linear placement, the objective function is formulated by a non-linear wirelength model subject to a density constraint. By relaxing the density constraint into the objective function, the placement problem can be solved by gradient-descent-based methods to obtain a high-quality placement solution with the desired tradeoff between wirelength and cell density.

The target of a routability optimizer is to evaluate the congestion of the current placement solution and then guide the basic engine accordingly to reduce routing congestion. Typically, the evaluation is presented in the form of congestion maps, which could be generated with a probabilistic model [2], an integrated global router [3], [4], [5], or recently emerging machine-learning-based techniques [6], [7]. Many existing approaches employ the congestion information guided by the congestion map. One is to spread cells in congested regions by cell inflation and rough legalization, such as POLAR 2.0 [8] and Ripple 2.0 [9] or white space allocation [10], [11]. NTUplace4h [12] and NTUplace4dr [4] choose to modify the density constraint according to the congestion information. A more explicit method is modeling routing demands or congestion as a new optimizing target and integrating it into the objective function. Lin *et al.* [13] proposed a congestion-aware net penalty model which treats nets as movable soft modules whose shapes are changeable and targets minimizing the total congestion covered by all nets. Liu *et al.* [7] developed a full-convolutional-network-based model to predict the routing congestion and integrated the model into the objective as a penalty term to optimize routability explicitly.

However, most recent studies focused on either predicting congestion or optimizing routability. In this work, we focus on both parts and present a routability-driven placement framework via cell padding with multiple features and strategy exploration, called PUFFER. Just like the puffer fish, cells in our placement framework could adjust their sizes according to their statuses. The underlying placement engine is based on ePlace[14], which models the density function by an electrostatic system. Finally, we compare the routing congestion of our PUFFER with a commercial EDA tool and evaluate the placement result by its global router. We summarize the main contributions of our work as follows:

- We develop a placement framework, called PUFFER, to optimize routability with cell padding during global placement and legalization. The consistent cell padding could preserve the optimization effect in the placement process, thus resulting in better routability.
- We propose a congestion estimation technique by imitating the behaviors of routing detours in the congested regions and clustered cell spreading with an accurate routing resource model to generate a more accurate congestion map quickly.
- We present a cell padding formula considering multiple features, which are inspired by the characteristics of convolutional and graph neural networks (CNN and GNN) and can accumulate accurate congestion information to alleviate congestion more effectively.
- We present a Bayesian-based strategy exploration scheme to develop an effective cell padding method, which could replace the experience-based manual parameter tuning process. In particular, our strategy exploration scheme is also suitable for other black-box problems with configurable strategy parameters.
- We demonstrate the effectiveness of PUFFER by comparing it with a state-of-the-art commercial EDA tools and an academic routability-driven placer, RePlAce, on industrial designs. Experimental results show that our algorithm achieves the best routability with 2.7× and 1.4× speedups on average, respectively.

The rest of this paper is organized as follows. Section II gives the placement background and formulates the routability-driven placement problem. Section III presents our routability-driven placement framework. Section IV shows the experimental results. Finally, our conclusions are given in Section V.

This work was supported by the National Key Research and Development Program of China under Grant 2022YFB4400500, and partially supported by AnaGlobe, ASUS, Delta Electronics, Maxeda Technology, TSMC, MOST/NSTC of Taiwan under Grant NSTC 110-2224-E-002-012, NSTC 111-2218-E-002-021, MOST 110-2221-E-002-177-MY3, and MOST 111-2622-8-002-023-SB.

II. Preliminaries

In this section, we first formulate the routability-driven placement problem and then introduce the electrostatic-based placement. Finally, global routing congestion is presented.

A. Routability-driven Placement Problem

Given a circuit netlist $H = (V, E)$, where $V = \{v_1, v_2, \cdots, v_n\}$ represent cell instances and $E = \{e_1, e_2, \cdots, e_m\}$ represent nets, a routability-driven placer determines legal locations for all movable cells to minimize the routing wirelength and overflow. A placement solution can be represented as $(\mathbf{x}, \mathbf{y}) = (x_1, \cdots, x_n, y_1, \cdots, y_n)$, where (x_i, y_i) denotes the coordinate of cell i.

In analytical global placement, the problem can be formulated as an unconstrained minimization problem in the following objective function by relaxing the density constraint as a penalty term:

$$\min_{(\mathbf{x},\mathbf{y})} f(\mathbf{x}, \mathbf{y}) = W(\mathbf{x}, \mathbf{y}) + \lambda D(\mathbf{x}, \mathbf{y}), \tag{1}$$

where $W(\mathbf{x}, \mathbf{y})$ denotes wirelength, $D(\mathbf{x}, \mathbf{y})$ denotes the density penalty to spread cells, and λ is the density penalty factor to balance the weight between wirelength and density. After obtaining the global placement result, we need to legalize the cell positions to remove cell overlaps and DRC violations.

B. Electrostatic-based Placement

The electrostatic-based global placement algorithm [14] is one of the state-of-the-art placement engines. It adopts the weighted-average (WA) wirelength model [15], [16] to approximate the HPWL,

$$W_{e_{ix}}(\mathbf{x}, \mathbf{y}) = \frac{\sum_{j \in e_i} x_j e^{\frac{x_j}{\gamma}}}{\sum_{j \in e_i} e^{\frac{x_j}{\gamma}}} - \frac{\sum_{j \in e_i} x_j e^{\frac{-x_j}{\gamma}}}{\sum_{j \in e_i} e^{\frac{-x_j}{\gamma}}}, \tag{2}$$

where $W_{e_i}(\mathbf{x}, \mathbf{y}) = W_{e_{ix}}(\mathbf{x}, \mathbf{y}) + W_{e_{iy}}(\mathbf{x}, \mathbf{y})$ is the weighted wirelength of net e_i, and γ is smoothing parameter used to adjust the accuracy.

To get the density penalty, the placement region is first divided into uniform bins by an $M \times M$ grid. With an analogy to an electrostatic system, the density penalty can be transferred to the total electric potential energy as follows:

$$D(\mathbf{x}, \mathbf{y}) = \sum_{v_i \in V, v_i \in b_{m,n}} w_i h_i \psi(m, n), \tag{3}$$

where $w_i h_i$ is the area of cell i, $b_{m,n}$ is the bin in row m and column n, and $\psi(m, n)$ denotes the electric potential of $b_{m,n}$,

$$\psi(m, n) = \sum_{i=0}^{M-1} \sum_{j=0}^{M-1} \frac{a_{i,j}}{\omega_i^2 + \omega_j^2} \cos(\omega_i m) \cos(\omega_j n), \tag{4}$$

$$a_{m,n} = \sum_{i=0}^{M-1} \sum_{j=0}^{M-1} \rho(i, j) \cos(\omega_m i) \cos(\omega_n j), \tag{5}$$

$$\rho(m, n) = \sum_{v_i \in b_{m,n}} w_i h_i, \tag{6}$$

where $\omega_k = 2\pi k / M, k = 0, 1, \cdots, M-1$, $a_{m,n}$ is an intermediate variable when calculating the electric potential, and $\rho(m, n)$ is the charge density of $b_{m,n}$.

With the smooth objective function, we can apply a gradient-based scheme like Nesterov's method [14] to solve the global placement problem.

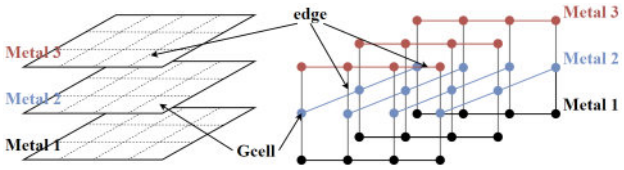


Fig. 1. The graph representation of the global routing problem.

C. Global Routing Congestion

For the global routing problem, the routing region can be partitioned into a three-dimensional array of global routing grids (Gcells) and modeled by a grid graph . Each node denotes a Gcell, and each edge corresponds to a boundary between two abutting Gcells. Figure 1 illustrates the grid graph model.

In this work, we use a Gcell-based routing resource model, which evaluates routing capacity and demand with respect to Gcells instead of routing edges. In this manner, the local nets' routing demands can be captured more precisely [17]. The routing overflow is defined as follows:

$$overflow(g) = \max\left(0, \frac{Dmd(g)}{Cap(g)}\right), \tag{7}$$

where $Dmd(g)$ denotes the routing demand of Gcell g, and $Cap(g)$ represents the routing capacity of Gcell g.

III. Our Proposed Algorithm

Figure 2 summarizes our routability-driven placement framework, which includes three major stages: global placement, routability optimization, and Legalization. In the global placement stage, the basic placement engine will spread cells out while optimizing the wirelength. When the cells spread sufficiently, a routability optimizer will be invoked to analyze the congestion information of the current placement result. Then, based on the congestion and placement information, we extract multiple features and generate cell padding to guide the subsequent global placement to reduce routing congestion. Finally, cells will be placed in a legal location to eliminate the cell overlaps and DRC violations. Meanwhile, the routability optimizer will also be called to offer padding-history-aware guidance for legalization. Inside the routability optimizer, some strategy parameters determine the effect of the optimizer. Instead of choosing the parameter manually, we adopt the bayesian-based method to explore the strategy space of the cell padding scheme. The following sections will detail the schemes mentioned above.

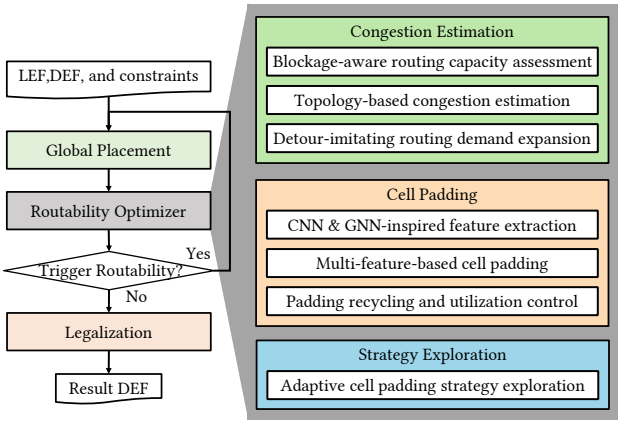


Fig. 2. Algorithm flow.

A. Routing-detour-imitation-based Congestion Estimation

According to our observation, cells in global placement tend to form bunches of clusters since the placement region is divided into bins by a grid, and cells in the same grid often overlap severely. This phenomenon makes a precise routing solution at this stage less meaningful as the overlapped cells inevitably lead to high routing demand. Based on this finding, we propose a fast routing congestion estimation algorithm to produce a 2D congestion map that can imitate the behaviors of routing detours and cell spreading in the same cluster. The congestion estimation algorithm consists of three parts: blockage-aware routing capacity map generation, probabilistic-based routing demand calculation, and detour-imitation routing expansion.

1) Blockage-aware Routing Capacity Assessment: When generating a blockage-aware routing capacity map, we use a Gcell-based routing resource model, which evaluates routing capacity and demand with respect to Gcells instead of routing edges. In this way, the routing capacity and demand could be carefully considered.

The routing capacity consists of two parts: the basic capacity and the blocked capacity induced by blockages. The basic capacity can be calculated according to technology information, like the metal wire width and spacing. To deduct the capacity occupied by the blockage, we consider a series of blockages such as pin obstructions, power and ground nets, and macro cells. Based on the shapes and locations of the blockages, we subtract the corresponding routing resource from the Gcell. Equation (8) models the routing capacity.

$$Cap_{H/V}(g) = \sum_{l.pd=H/V}^{l\in L} \frac{GcellLength_{H/V}}{MetalWidth_l + WireSpacing_l} - \sum_{b\in Blk} \frac{OL_{H/V}(b,g)}{MetalWidth_{b.l} + WireSpacing_{b.l}}, \tag{8}$$

where $Cap_{H/V}(g)$ is the horizontal or vertical routing capacity of Gcell g, $l.pd$ denotes the preferred routing direction of metal layer l, L represents the set of metal layers, Blk represents the set of blockages, $OL_{H/V}(b,g)$ is the horizontal or vertical overlaps between blockage b and Gcell g, and $b.l$ is the metal layer where the obstacle is located.

2) Topology-based Congestion Estimation: After obtaining the capacity map, we need to estimate the routing demand of all nets. First, we use FLUTE [18] to get the RSMT topology of the net. Based on the topology, the net is represented by a set of two-point nets, in which the point denotes the cell pin or the Steiner point. According to the two-point nets, we can estimate the routing demand by the following method. For the "I"-shaped two-point net, in which the two points have the same coordinate in the x or y direction, the Gcells passed by the two-point net will be consumed a unit routing demand in the corresponding direction. For the "L"-shaped two-point net, in which the two points have different coordinates in both directions, the Gcell covered by the bounding box enclosing the two points will require an average routing demand for the L-shaped routing path. Meanwhile, we add a pin penalty demand to capture the routing demand of local nets whose pins are located in the same Gcell. Figures 3 (a) and (b) illustrates the horizontal and vertical demands for a net, where a darker color corresponds to a higher demand.

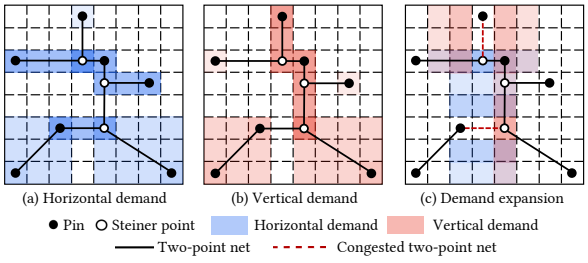


Fig. 3. Congestion estimation. (a) and (b) illustrate the horizontal and vertical routing demands. (c) demonstrates the routing demand expansion.

3) Detour-imitating Routing Demand Expansion: Considering that cells cluster in the global placement stage, the aforementioned routing demand would result in a more concentrated distribution of the routing overflow, which will neglect the surrounding configuration. Instead of directly spreading clustered cells which could bring instability to the electrostatic system, we propose an expanding-based refinement to imitate the behaviors of cell spreading and routing detours. According to the previously generated routing topology, we pick all congested I-shaped two-point nets and try to transfer the original routing demand to the surrounding region with sufficient routing resources. Besides, we adopt the following scheme to distinguish between cell spreading and routing detours. If the point in a two-point net is a Steiner point, an extra routing demand in the perpendicular direction will be added to represent the detour path connecting the expanded I-shaped routing path. However, an extra routing path is unnecessary for the pin in a two-point net since cells could be moved around. Figure 3 (c) illustrates an example of routing demand expansion. The red dotted lines represent the congested I-shaped two-point nets and colored regions correspond to the expanded routing demand.

B. Multi-feature-based Cell Padding

After estimating the congestion map, we pad cells with fillers on both left and right sides to enlarge the cell areas. The padded cells will spread more aggressively in the following global placement iterations. Previous work mainly focuses on using local information to obtain the ratio of cell inflation. However, due to cell clustering, local information might be unable to distinguish different cells in the same cluster. Thus, we propose a multi-feature-based cell padding strategy with a padding control and recycle system to capture the information well.

1) CNN and GNN-inspired Feature Extraction: Since the local information is quite limited, we propose three categories of features, each targeting one aspect. By combining them, we can get a more comprehensive view. The first kind of features is traditional local information like local congestion and local pin density. Instead of truncating the overflow to keep the positive part, we preserve the original value to consider the deviation between the congestion map and the global routing result. The local congestion can be formulated as follows:

$$LCg(c) = \max_{g\in Pos(c)} Cg(g), \tag{9}$$

$$Cg(g) = \begin{cases} max(Cg_H(g), Cg_V(g)), & Cg_H(g)\cdot Cg_V(g) < 0, \\ Cg_H(g) + Cg_V(g), & otherwise, \end{cases} \tag{10}$$

$$Cg_{H/V}(g) = \frac{Dmd_{H/V}(g) - Cap_{H/V}}{\max(Cap_{H/V}, 1)}, \tag{11}$$

where $LCg(c)$ is the local congestion of cell c, $Pos(c)$ is the Gcell set including all Gcells overlapped with cell c, and the $Cg(g)$ is the congestion of Gcell g, a function of the horizontal and vertical congestion $Cg_{H/V}(g)$. Pin density could be calculated by the number of cell pins in each Gcell divided by the number of available sites in each Gcell.

The characteristic of CNN inspires our second kind of features. In a CNN, the convolution kernel will consider the effect of adjacent elements to extract the neighboring information. Therefore, we introduce the surrounding congestion and surrounding pin density targeting a larger spatial region. To get the surrounding feature, we expand the cell's bounding box with a margin in both horizontal and vertical directions. The size of the expanded bounding box equals that of the convolution kernel size, and we choose the mean filter operation as the kernel to conduct the convolution on the expanded region. Figure 4 illustrates an example of the CNN-inspired features.

The last kind of features is motivated by the GNN. Since a CNN can only deal with the data in the Euclidean space, a graph structure like the netlist of a circuit would need some topology-based features to generate a more comprehensive view. Thus, we develop an attribute of pin congestion to each cell, which could aggregate information from

the graph structure in the routing topology. The formulation is given by

$$PCg(c) = \sum_{p \in c.Pin} PCg(p), \tag{12}$$

$$PCg(p) = \min_{l \in Path(p)} \max_{g \in l} Cg(g), \tag{13}$$

where $PCg(c)$ is the pin congestion of cell c, the summation of the congestion of its pin $PCg(p)$. The congestion of pin p is the minimum congestion of all candidate paths in $Path(p)$, and the congestion of a path is the maximum Gcell congestion along the path. The candidate paths $Path(p)$ include all possible L- or Z-shaped routing paths for all two-point nets associated with pin p. Figure 4 demonstrates an example of feature extraction for a cell.

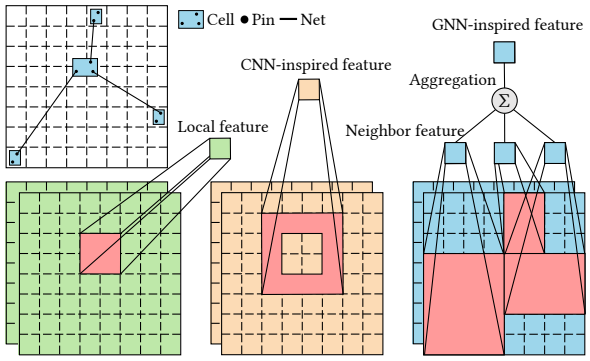


Fig. 4. Feature extraction. Our features include three parts: local features, CNN-inspired features, and GNN-inspired features.

2) Multi-feature-based Cell Padding: After collecting all the required features, we use the formulation below to calculate the expected padding value:

$$Pad(c) = \log(\max(\sum_{i=1}^{|F|} \alpha_i \cdot f_i(c) + \beta, 1)) \cdot \mu, \tag{14}$$

where α_i is the weight of the ith feature f_i, and β and μ are strategy parameters. F is the set of all features and $|F|$ denotes the total number of features. The logarithmic function intends to smooth the distribution of padding values, which could improve the stability of the padding process. All strategy parameters will be explored in a more general way, instead of adjusting manually, to be detailed in Section III-C.

3) Padding Recycling and Utilization Control: Since the global placement is a relatively long process, we adopt the incremental padding strategy in which each round of padding will base on the result of the preceding round. Besides, we develop a padding recycle mechanism to take back the previous padding for the cell away from congested regions. The recycling rate depends on the padding history defined as follows:

$$r_i(c) = \frac{i - pt(c)}{i + \zeta}, \tag{15}$$

where r_i denotes the recycle rate of cell c in ith iteration, $pt(c)$ is the number of times cell c has been padded, and ζ is a strategy parameter determining the recycling efforts. After calculating the padding for each cell, the cells without positive padding need to go through the recycling process to withdraw a part of the history padding to avoid unnecessary padding which might degrade wirelength.

Our routability optimization process is summarized in Algorithm 1. First, we estimate congestion and extract features from the congestion map and placement information. Then we calculate and recycle the padding according to the formulas. Meanwhile, we determine the padding utilization by the following equation:

$$pu_i = pu_{low} + \frac{i-1}{\xi - 1} \cdot (pu_{high} - pu_{low}), \tag{16}$$

where pu_i is the padding utilization of ith iteration in Line 5, pu_{low} and pu_{high} are strategy parameters that determine the respective minimum and maximum padding utilizations during the whole placement stage. ξ denotes the max iteration of routability optimization. Controlling padding utilization can avoid over padding in the early stage to hinder the subsequent optimization. Based on the padding utilization, if the total padding exceeds the predefined utilization, all the padding will be scaled to meet the constraint ensured by Lines 6–9.

Algorithm 1 Cell Padding Algorithm

Input: available placement area A, parameters in the padding formulation $param$, and history padding information HP.
Output: padding list PL.

1: $cm = congestionEstimation()$;
2: $feature = extractFeature(cm)$;
3: $PL = getPadding(feature, param)$;
4: $PL = recyclePadding(PL, HP)$;
5: $padding\_util = getTargetUtilization()$;
6: $padding\_area = getTotalPadding(PL)$;
7: **if** $padding\_area > padding\_util \cdot A$ **then**
8: $\quad sr = \frac{padding\_util \cdot A}{padding\_area}$;
9: $\quad PL = scalePaddingByRatio(PL, sr)$;
10: **return** PL;

There are three conditions to trigger the routability optimizer. One condition is when the density overflow is lower than a threshold τ, where the density overflow is defined as the summation of the cell overflow area of all bins divided by the number of bins. Another condition is that the padding utilization of the preceding round of routability optimization is less than a threshold η, meaning that the padding is converging. The last condition is that the call time of the routability optimizer is less than the maximum iteration ξ. When all three conditions are met, the routability optimizer will be called. Otherwise, the wirelength-driven placement engine will optimize both wirelength and density until it converges. The aforementioned τ, η, and ξ are strategy parameters.

C. Bayesian-based Strategy Exploration

During the whole placement stage, various strategy parameters need to be determined. Most previous studies choose the parameters manually according to the experimental results. To avoid repeated time-consuming manual parameter tuning, we propose a Bayesian-based strategy exploration scheme to search for a better configuration, which is also suitable for other black-box problems with optional strategies and configurable parameters. Since placement can be considered as an evaluation-expensive black-box derivative-free process, we apply the sequential model-based global optimization (SMBO) with the tree-structured Parzen estimator (TPE) approach in [19]. The strategy exploration scheme is shown in Algorithm 2 and Algorithm 3.

The basic step for strategy exploration is the parameter exploration described in Algorithm 2. The parameters could be continuous ones in formulas or discrete ones to choose different strategies. Based on the given parameter list and parameter range, we use SMBO to generate parameters by history observations and evaluate parameters with placement and global routing. The objective for the evaluation is set as the total overflow ratio of both horizontal and vertical directions. The parameter exploration will stop when the results do not improve for certain iterations or the total iterations exceed the limit. The two termination criteria can be determined by the computing resources. When the parameter exploration terminates, we will adjust the parameter ranges according to the observed trends.

Algorithm 3 describes our strategy exploration scheme, which includes two steps, global and local exploration. First, we set initial ranges for all parameters and conduct exploration for all parameters simultaneously to get rough ranges (Line 1–2). Then we split parameters into groups by their relevance. Parameters with strong ties will be assigned to the same group. Next, we explore each group of parameters

Algorithm 2 Parameter Exploration

Input: parameter list PL, parameter range PR, evaluation function $Eval$, exploration time limit TC, and early stop limit EC .
Output: determined parameters range PR.

$obs = \varnothing$;
$tc = 0$;
$npc = 0$;
$best\_result = -\infty$;
while $tc < TC$ && $npc < EC$ **do**
  $param\_value = getParam(PR, obs)$;
  $result = Eval(param\_value)$;
  $obs = obs \cup (param\_value, result)$;
  if $result < best\_reslut$ **then**
    $best\_result = result$;
    $npc = 0$;
  $tc += 1$;
  $npc += 1$;
$PR = updateParamRange(obs)$;
return $npc > EC$;

Algorithm 3 Strategy Exploration

Input: parameter list PL, exploration time limit TC.
Output: determined parameters PV.

$param\_range = getInitialParamRange(PL)$;
$param\_range = paramExploration(PL, param\_range)$;
$param\_group\_list = groupParam(PL)$;
$tc = 0$;
$early\_stop = false$;
while !$early\_stop$ && $tc < TC$ **do**
  $early\_stop = true$;
  for $pgl \in param\_group\_list$ **do**
    $flag = paramExploration(pgl, param\_range)$;
    $early\_stop = early\_stop \ \& \ flag$;
  $tc += 1$;

with others fixed in the middle value of the ranges. It is worth noting that the exploration of different groups can be conducted in parallel to speed up the process. We get the final parameter ranges when all the group explorations stop early. Based on the ranges, we determine the final configuration by taking the median of the range. To demonstrate the robustness of our cell padding method and the adaptability of our strategy exploration, we use a small design with the routability problem to conduct the strategy exploration and apply the result to other large-scale benchmarks.

D. White-space-assisted Legalization

After finishing global placement, we need to place cells in legal positions while eliminating overlaps and DRC violations. According to our observation, after global placement, the distribution of cells is still clustered with a smaller size of clusters compared with that in the global placement stage. Since legalization will minimize the displacement of cells with respect to the global placement result, cells of the same cluster will cling together, degrading routability. Consequently, we inherit the padding from global placement to introduce white space in the congested region and make the whole placement process more consistent.

Different from global placement, the width of a cell should be the multiple of the technology site in legalization. So we discretize the padding using a staircase function defined as follows:

$$DisPad(c) = \left\lfloor \theta \cdot \left(\frac{Pad(c)}{mp} + \frac{1}{2} \right) \right\rfloor, \tag{17}$$

where $DisPad(c)$ denotes the discrete padding of cell c, θ is a strategy parameter, and mp is the maximum padding of all cells. To control the padding utilization, we limit the total padding area to less than 5% of the total movable cells area. If the total padding exceeds the limit, we pick the cells with the smallest padding in each discrete padding level and relegate them to reduce the total padding area until meeting the utilization constraint. After adding cell padding, we employ an Abacus-based[20] algorithm for legalization.

IV. Experimental Results

We implemented our routability-driven placement framework in the C++ language. The experiments were conducted on the 64-bit CentOS Linux workstation with the Intel(R) Xeon(R) Platinum 8260 CPU @ 2.40GHz using eight threads. Our benchmarks contain ten large-scale industrial designs as shown in Table I, where the numbers of fixed macros, movable standard cells, nets, and pins of all movable cells are denoted by "#Macros", "#Cells", "#Nets", and "#Pins", respectively.

TABLE I
STATISTICS OF THE BENCHMARKS.

Benchmark	#Macros	#Cells	#Nets	#Pins
OR1200	22	122K	193K	660K
ASIC_ENTITY	45	149K	155K	630K
BIT_COIN	43	760K	760K	3151K
MEDIA_SUBSYS	70	1228K	1296K	5235K
MEDIA_PG_MODIFY	70	1228K	1296K	5235K
A53_ADB_WRAP	7	1232K	1300K	5242K
CT_SCAN	39	1249K	1317K	5282K
CT_TOP	38	1270K	1272K	4091K
E31_ECOREPLEX	56	1533K	1537K	6303K
OPENC910	332	1590K	1741K	7276K

To verify the effectiveness of our routability-driven placement framework, we directly compared our proposed framework PUFFER with a leading commercial placement tool (denoted by Commercial_Inn) and RePlAce [5] using the global router in Innovus to evaluate the routability of the placement result. The experimental results are summarized in Table II. In this table, the "HOF", "VOF", "WL", and "RT(s)" are the routing overflow ratios in the horizontal and vertical directions, global routing wirelength, and runtime, respectively. The first three results are reported by the Innovus global router. Since the values of "HOF" and "VOF" are relatively small, we compared the average value instead of the average ratio. Besides, based on the industrial experience, a design with less than 1%–2% overflows reported by the global router is acceptable, as the overflow ratios of this magnitude can be fixed in the subsequent steps. Therefore, we took 1% as the pass criterion for routability.

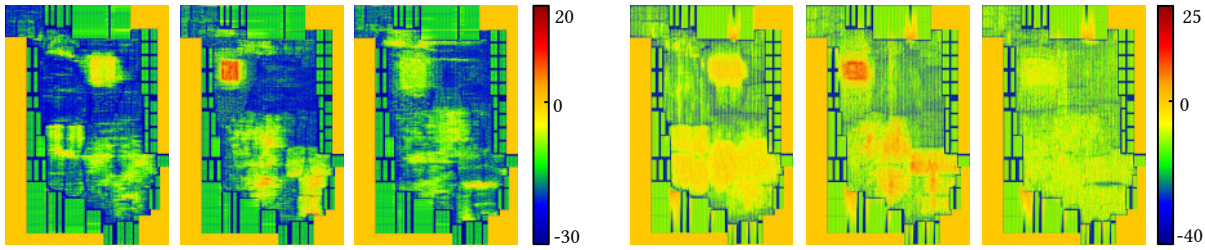
From Table II, our algorithm achieves the best horizontal and vertical congestion ratios on average and obtains the same pass counts as Commercial_Inn. As for the runtime, our PUFFER placer achieves respective 2.7× and 1.4× speedups on average compared with Commercial_Inn and RePlAce. Besides, 4.5% of extra wirelength compared with Commercial_Inn is acceptable since the objective of our placement framework is the HPWL instead of the actual routing wirelength. Moreover, using the Innovus global router as the evaluator is also in favor of the "Commercial_Inn" placer. Figure 5 shows the comparison of horizontal and vertical congestion maps of the placement results generated by Commercial_Inn, RePlAce, and our PUFFER, reported by the Innovus global router. The congestion maps also demonstrate the effectiveness of our algorithm. The experimental results show the robustness and practicality of our routability-driven placement framework.

V. Conclusions

We have proposed a routability-driven placement framework, called PUFFER, via cell padding with multiple features and strategy exploration. To generate cell padding, we have first presented a probabilistic-based congestion estimation algorithm that could imitate the behaviors of routing detours and spreading cluster cells. Then, based on the congestion map and placement information, we have generated cell padding according to the features inspired by the characteristics of

TABLE II
COMPARISON OF HOF, VOF, WL, AND RT WITH COMMERCIAL_INN, REPLACE, AND OUR PUFFER ON INDUSTRIAL BENCHMARKS.

Benchmark	Commercial_Inn				RePlAce [5]				Our PUFFER			
	HOF(%)	VOF(%)	WL	RT(s)	HOF(%)	VOF(%)	WL	RT(s)	HOF(%)	VOF(%)	WL	RT(s)
OR1200	0.88	0.21	3724999	1006	0.92	1.33	3238951	449	0.79	0.96	3145834	243
ASIC_ENTITY	0.27	0.07	16562470	804	0.4	0.08	17699450	487	0.25	0.04	17237170	364
BIT_COIN	0.03	0.07	10216500	3551	0.01	0.04	12756620	2400	0	0.05	12136850	1471
MEDIA_SUBSYS	0.67	5.83	30304690	8005	4.44	14.84	33373000	3350	0.38	3.03	31900040	3195
MEDIA_PG_MODIFY	0.15	0.39	30524130	7643	0.88	2.21	33768920	2884	0.07	0.54	34008440	1630
A53_ADB_WRAP	0.59	2.4	30438870	7074	3.34	14.44	33464500	3388	0.43	3.7	32607770	3119
CT_SCAN	0	0.1	32966640	5316	0.57	0.25	34120310	3017	0.01	0.01	33743970	1917
CT_TOP	0	0.07	27003190	3887	0	0.04	27632000	1988	0	0.03	27222070	1350
E31_ECOREPLEX	0.01	0.14	22108530	6641	0	0.3	27342060	6581	0	0.15	25436660	4932
OPENC910	0.81	0.14	45595670	9491	1.74	0.15	52682470	6086	0.96	0.11	49007690	5354
Average	0.341	0.942	0.954	2.699	1.230	3.368	1.035	1.424	0.289	0.862	1.000	1.000
Pass Count	10	8	-	-	7	6	-	-	10	8	-	-



(a) Commercial_Inn Hmap (b) RePlAce Hmap (c) Our PUFFER Hmap (d) Commercial_Inn Vmap (e) RePlAce Vmap (f) Our PUFFER Vmap

Fig. 5. Congestion maps for MEDIA_SUBSYS reported by the commercial global router in Innovus. (a), (b) and (c) are the horizontal congestion maps for the placement results of a leading commercial placer (denoted by Commercial_Inn), RePlAce, and our framework PUFFER. (d), (e) and (f) are the vertical congestion maps for the placement results of Commercial_Inn, RePlAce, and PUFFER.

CNN and GNN. We have also designed a padding recycle scheme and utilization control method to make the most of cell padding. In addition, we have applied a Bayesian-based strategy exploration to search for a better configuration for our placement framework. Experimental results have shown that our algorithm achieves competitive routability with faster runtime. In the future, we plan to enhance our strategy exploration by introducing more optional strategies and apply our strategy exploration to tackle different problems.

REFERENCES

[1] W.-K. Chow and E. F. Young, "Placement: From wirelength to detailed routability," *IPSJ Transactions on System LSI Design Methodology*, vol. 9, pp. 2–12, 2016.

[2] P. Spindler and F. M. Johannes, "Fast and accurate routing demand estimation for efficient routability-driven placement," in *Design, Automation & Test in Europe Conference & Exhibition*, 2007, pp. 1–6.

[3] U. Brenner, A. Hermann, N. Hoppmann, and P. Ochsendorf, "BonnPlace: A self-stabilizing placement framework," in *Proceedings of ACM International Symposium on Physical Design*, 2015, pp. 9–16.

[4] C.-C. Huang, H.-Y. Lee, B.-Q. Lin, S.-W. Yang, C.-H. Chang, S.-T. Chen, Y.-W. Chang, T.-C. Chen, and I. Bustany, "NTUplace4dr: A detailed-routing-driven placer for mixed-size circuit designs with technology and region constraints," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 3, pp. 669–681, 2017.

[5] C.-K. Cheng, A. B. Kahng, I. Kang, and L. Wang, "RePlAce: Advancing solution quality and routability validation in global placement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 9, pp. 1717–1730, 2019.

[6] J. Chen, J. Kuang, G. Zhao, D. J.-H. Huang, and E. F. Young, "PROS: A plug-in for routability optimization applied in the state-of-the-art commercial EDA tool using deep learning," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, 2020, pp. 1–8.

[7] S. Liu, Q. Sun, P. Liao, Y. Lin, and B. Yu, "Global placement with deep learning-enabled explicit routability optimization," in *Design, Automation & Test in Europe Conference & Exhibition*, 2021, pp. 1821–1824.

[8] T. Lin and C. Chu, "POLAR 2.0: An effective routability-driven placer," in *Proceedings of ACM/IEEE Design Automation Conference*, 2014, pp. 1–6.

[9] X. He, Y. Wang, Y. Guo, and E. F. Young, "Ripple 2.0: Improved movement of cells in routability-driven placement," *ACM Transactions on Design Automation of Electronic Systems*, vol. 22, no. 1, pp. 1–26, 2016.

[10] C. Li, M. Xie, C.-K. Koh, J. Cong, and P. H. Madden, "Routability-driven placement and white space allocation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 5, pp. 858–871, 2007.

[11] W.-T. J. Chan, P.-H. Ho, A. B. Kahng, and P. Saxena, "Routability optimization for industrial designs at sub-14nm process nodes using machine learning," in *Proceedings of ACM International Symposium on Physical Design*, 2017, pp. 15–21.

[12] 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*, vol. 33, no. 12, pp. 1914–1927, 2014.

[13] J.-M. Lin, C.-W. Huang, L.-C. Zane, M.-C. Tsai, C.-L. Lin, and C.-F. Tsai, "Routability-driven global placer target on removing global and local congestion for VLSI designs," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, 2021, pp. 1–8.

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

[15] M.-K. Hsu, Y.-W. Chang, and V. Balabanov, "TSV-aware analytical placement for 3D IC designs," in *Proceedings of ACM/IEEE Design Automation Conference*, 2011, pp. 1–6.

[16] 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*, vol. 32, no. 4, pp. 497–509, 2013.

[17] A. B. Kahng, L. Wang, and B. Xu, "TritonRoute-WXL: The open source router with integrated DRC engine," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 4, pp. 1076–1089, 2022.

[18] C. Chu and Y.-C. Wong, "FLUTE: Fast lookup table based rectilinear steiner minimal tree algorithm for vlsi design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 1, pp. 70–83, 2007.

[19] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Advances in Neural Information Processing Systems*, vol. 24, 2011.

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