RL-Legalizer: Reinforcement Learning-based Cell Priority Optimization in Mixed-Height Standard Cell Legalization

†Sung-Yun Lee, †Seonghyeon Park, †Daeyeon Kim, †Minjae Kim, ‡Tuyen P. Le and †*Seokhyeong Kang
†*Dept. of Electrical Engineering, POSTECH, Rep. of Korea*
‡*AgileSoDA Company, Rep. of Korea*
*shkang@postech.ac.kr

***Abstract*—Cell legalization order has a substantial effect on the quality of modern VLSI designs, which use mixed-height standard cells. In this paper, we propose a deep reinforcement learning framework to optimize cell priority in the legalization phase of various designs. We extract the selected features of movable cells and their surroundings, then embed them into cell-wise deep neural networks. We then determine cell priority and legalize them in order using a pixel-wise search algorithm. The proposed framework uses a policy gradient algorithm and several training techniques, including grid-cell subepisode, data normalization, reduced-dimensional state, and network optimization. We aim to resolve the suboptimality of existing sequential legalization algorithms with respect to displacement and wirelength. On average, our proposed framework achieved 34% lower legalization costs in various benchmarks compared to that of the state-of-the-art legalization algorithm.**

***Index Terms*—deep reinforcement learning, cell-wise neural network, displacement, wirelength**

I. Introduction

In advanced technology, mixed-height standard cells are commonly used to derive various design benefits. Using multi-height cells can efficiently address the occurrence of complex corner cases and improve pin accessibility and routability, which are critical to chip performance [1], [2]. However, the design complexity owing to mixed-height cells leads to difficulty in cell placement [3]. The placement becomes more complicated and critical to design quality because the legalization of multi-height cells affects multiple rows where many other cells can be placed. Moreover, power rail alignment becomes tricky, especially for even row-height cells. Numerous design constraints, such as high utilization, hard macro blocks, and group regions, further increase the complexity of this problem [2], [4]. As the design complexity increases, the cell placement phase leads to a dramatic variation in the final design quality.

Since the ICCAD-2017 CAD contest [4], various algorithms for the mixed-height cell legalization have been proposed to resolve the increasing constraints, including complex design rules at sub-14 *nm*, design utilization, irregular placeable regions, edge spacing, and pin accessibility. Zhu *et al.* [6] used an iterative cell reassignment technique and a technology-aware legalization algorithm. Li *et al.* [7], [8] proposed a window-based cell insertion method and network flow-based optimization. Do *et al.* [9] applied a pixel-wise search algorithm and a cell swap technique to obtain a fast algorithm. Chen *et al.* [10] presented a robust modulus-based matrix splitting method to solve the linear complementarity problem. Netto *et al.* [11] presented a convolutional neural network (CNN)-based algorithm selection framework to exploit the best among the three mentioned state-of-the-art algorithms [6]–[9] for each design.

These rule-based sequential algorithms, however, cannot achieve universally optimal solutions for different circuit designs. The rule that achieves the best quality of result (QoR) for one design may not produce satisfactory results for other designs that have different characteristics. Some recent works [5]–[8] legalized cells based on cell horizontal locations, while Do *et al.* [9] legalized them in order of cell sizes. However, a considerable number of cells have the same ordering conditions; multiple cells are positioned at the same x-coordinate or have the same size. Therefore, the cells with the same condition are legalized in a random order. The presented cell priority rules are reasonable but provide suboptimal results in some designs. In most designs, the quality of the legalization results fluctuates depending on the sequence of cell legalization. We describe the effects of the cell priority on the legalization in Section II-C.

Reinforcement learning (RL) is generally used to resolve the limitations of many sequential algorithms. In electronic design automation (EDA), an RL is a promising paradigm that can explore the hidden potential of the sequential EDA algorithms [12]. Lu *et al.* [13] used an RL in an iterative gate-sizing problem for timing optimization. Liao *et al.* [14] used an RL in the global routing phase to optimize a sequential A* search algorithm. An RL is applied in detailed routing to optimize the wirelength, wire congestion, or design rules [15]–[17]. Recently, deep RL has also been actively used and studied at the standard cell placement phase. Agnesina *et al.* [18] optimized placement parameters in a commercial tool using RL. Kirby *et al.* [19] improved a force-based global placement algorithm with RL, and Mallapaa *et al.* [20] used RL in the detailed placement for coarse arrangements of standard cells.

In this paper, we propose a deep RL framework to optimize the cell priority on legalization considering various features of the cells and their surroundings. We extract the features of all legalizable cells in a design and train cell-wise deep neural network (DNN) models with a policy gradient algorithm. Using a policy network, we prioritize the cells to get the largest expected reward from the extracted features. We use the pixel-wise search algorithm to legalize cells by searching for multiple legal locations and selecting the best one. A value network estimates the reward from the updated features and boosts training convergence. The main contributions are as follows:

- We propose a deep RL framework to optimize the priority of the mixed-height cells to be legalized. We empirically demonstrate that the cell priority highly affects the QoRs of designs. In our framework, we consider various properties of cells and their surroundings to optimize cell priority and achieve high design quality for various designs.
- We develop a deep RL-based mixed-height cell legalizer (*RL-Legalizer*) with considering various design constraints and complex design rules, including maximum displacement, target utilization, edge spacing, and fence regions.
- We partition the designs into multiple grid-cells (*Gcells*) and train each Gcell as a subepisode in the RL training algorithm. A Gcell partitioning makes our framework robust for much larger designs and improves training performance.
- Our framework can be applied to any sequential legalization algorithms. Using RL with the pixel-wise search legalization algorithm, we achieve less cell movement from the global placement results and less wirelength than the cell size-ordered state-of-the-art legalizer.

The remainder of this paper is organized as follows. Section II describes the background of mixed-height cell legalization, our legalization algorithm, and verification experiments. Section III presents our RL framework, DNN models, training algorithms, and RL techniques. Section IV reports the experimental setup and the results on the several benchmarks. We conclude the paper in Section V.

II. Preliminaries

A. Cell Legalization

Mixed-height standard cell legalization is a part of the detailed placement phase. Cells of different heights should be placed in legal

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT). (No.2021-0-00754, Software Systems for AI Semiconductor Design and No.2022-0-01172, DRAM PIM Design Base Technology Development). The authors thank Dr. J. Jung for providing valuable feedback.

978-3-9819263-7-8/DATE23/© 2023 EDAA

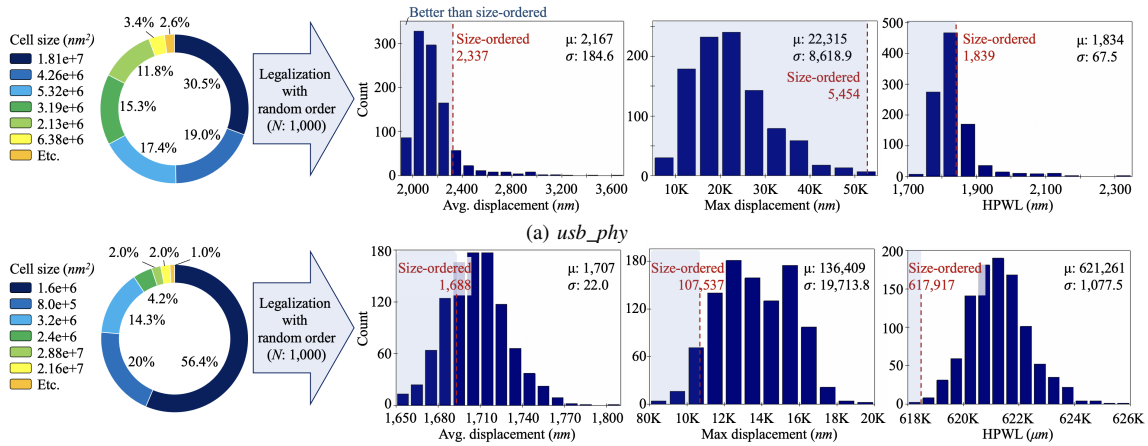


Fig. 1. Distribution of cell size in two example benchmarks and the distribution of resulting metrics, including the average displacement, maximum displacement, and HPWL, of the random-ordered cell legalization. The two example benchmarks are (a) *usb_phy* implemented with 75% utilization and aspect ratio 1.0 in 45 *nm* Nangate technology [31] and (b) *pci_bridge32_b_md3* in the contest benchmarks [4]. We performed the random-ordered legalization 1,000 times. The red dashed lines represent the result of cell legalization sorted by size, and the blue regions represent improvement potentials from the size-ordered results. μ and σ represent the mean value and standard deviation of each distributed QoR, respectively.

locations, aligning the cells with placement sites and power rails, and adhering to design rules. In general, in legalization problem, we assume that the result of the preceding global placement is well-optimized with respect to timing or wirelength. The main objective of cell legalization is to minimize cell movement (*displacement*) with respect to the global placement result. Furthermore, the maximum displacement and estimated wirelength should be considered, because a large displacement can cause critical paths, and wirelength is a crucial design cost. Therefore, our objective is to minimize the legalization cost, including the average displacement, the maximum displacement, and the half-perimeter wirelength (HPWL). In the problem, we address various constraints that include maximum displacement, target utilization, edge spacing, fence regions, and other design rules [4].

B. Pixel-wise Search Algorithm

We use a pixel-wise search algorithm to legalize the mixed-height cells (Fig. 2). First, the entire design is divided into pixels of minimum width and height, i.e., in the unit of placement site and spacing of power rails. Then, we explore available pixel locations for each target cell using a diamond searching method within a search space. The search boundary is determined to be proportional to the maximum displacement constraint and cell size. Finally, the location with the minimum displacement is designated to legalize the cell.

Because the pixel-wise search algorithm is considerably fast and efficient, it is suitable for the time-consuming process of RL training. This algorithm is also used in one of the state-of-the-art open-sourced academic legalization tools [9], [26]. It determines the legalization order by sorting the cells in descending order of cell size. However, the size-ordered cell priority produces suboptimal legalization results. To compensate for the suboptimality, it uses various heuristic methods, such as rearrangement, corner-weight movement, and cell swapping. In this work, we aim to find the optimal cell priority by using the RL technique. We will compare the legalization quality between the size-ordered method with the heuristic methods and our RL-ordering method without any heuristic methods.

C. Effect of Cell Priority

We analyzed the effect of the cell sequence on mixed-height cell legalization. Do *et al.* [9] sorted movable cells in descending order according to cell size. Because cells with large widths or heights are more likely to affect adjacent cells or multiple rows, the cell size-ordered legalization seems reasonable. However, legalizing the large cells in the early stages often leads to significant displacements of adjacent cells, because a large area is occupied. In addition, in most

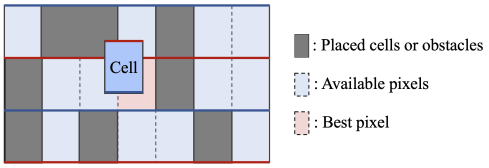


Fig. 2. The pixel-wise search legalization algorithm explores the available pixels (blue regions) within a search space, then elects the best pixel with respect to displacement (a red region).

design benchmarks, more than 30% cells have the same size based on the largest portion (Fig. 1). This large proportion indicates that a significant number of cells are likely to be legalized in random order.

To demonstrate the effect of cell sequence on the quality of legalization, we performed a simple simulation. We iterated the academic legalizer [26] 1,000 times with a random sequence and presented the distribution of the QoRs, including the average displacement, maximum displacement, and HPWL, and compared with the size-sorted results (Fig. 1). For the two sample benchmarks, the QoRs distribute over a wide range as the order of the cells changes. In addition, the QoRs can be significantly improved from the size-sorted results. Therefore, optimizing the cell priority by considering more features of cells and surroundings can improve the design quality.

III. RL-Legalizer Framework

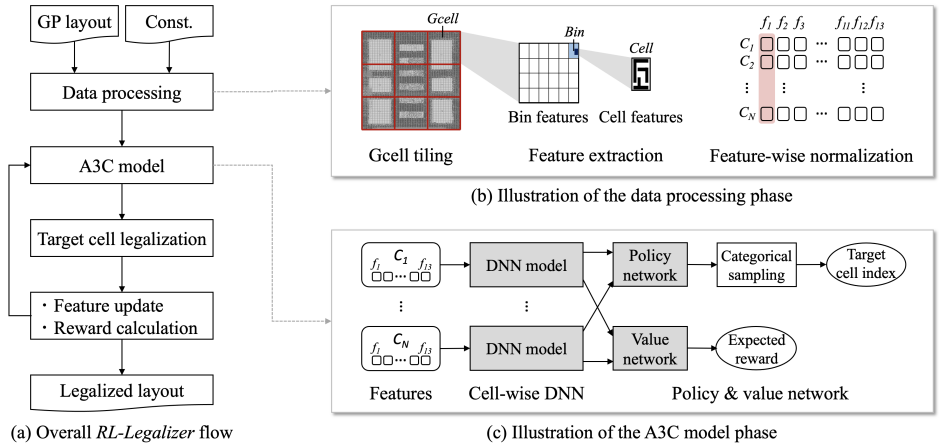
In this section, we present our cell priority optimization problem in the Markov decision process (MDP) and define the state, action, and reward function. We then introduce our overall deep RL framework and legalization flow. Our policy gradient training algorithm, cell-wise neural network models, and strategies for efficient training are described in detail.

A. Markov Decision Process

A cell priority optimization problem is a sequential decision of the cell (action) in a given environment (state) to obtain the maximum design quality (reward). This problem can be represented as an MDP, and our deep RL setup is as follows.

- **State** s: A state is an $N \times F$ array of features for given movable cells and [1]*bins*, where N represents the number of movable cells, and F is 13, the number of features for each cell (Table I). For each cell, 13 features consist of three categories: (i) cell features,

[1]Designs are divided into Gcells, and each Gcell is divided into multiple bins to extract features of cell surroundings. ∼20 cells are placed in a bin.



(a) Overall *RL-Legalizer* flow

(b) Illustration of the data processing phase

(c) Illustration of the A3C model phase

Fig. 3. (a) Our deep reinforcement learning framework to optimize cell priority in the mixed-height standard cell legalization. (b) The data processing phase consists of Gcell tiling, extraction of features of cells and surroundings (bin), and feature-wise normalization. (c) In the A3C model phase, features are embedded in cell-wise DNN models. The index of the target cell and the expected reward are obtained through policy and value networks, respectively.

(ii) bin features, and (iii) others. The first seven features are the cell features, which indicate the coordinates, size, net count of the cell, the number of overlapped cells, and the average Manhattan distance of the two nearest obstacles or design boundaries from the cell. The next four features are the bin features, which indicate the total cell area, placeable area, excluding macros or legalized cells, the total number of overlapped cells, and bin density error (Equation 1) [21] of the bin region, where the cell is placed. Bin density error is a squared error between the cell area in the bin B_i (CA_{B_i}) and the average cell area of bins in the design ($CA_{B,avg}$).

$$DE_{B_i} = (CA_{B_i} - CA_{B,avg})^2 \tag{1}$$

The remaining two features indicate the number of movable cells in the Gcell where the cell is placed after global placement and the number of legalized cells in the Gcell.

- **Action** a: An $N \times 1$ vector is constructed through the policy network $\pi_\theta(s)$. Each element of the vector indicates the legalization priority of each cell. We select the target cell by categorical sampling, where one of the cells with high priorities is selected from the vector. The target cell is then legalized using the pixel-wise search algorithm.
- **Reward** $r(s, a)$: We define a reward function as the sum of the reciprocal of the displacement and the reciprocal of ΔHPWL caused by target cell legalization, which can be formulated as

$$r(s,a) = \begin{cases} k_1 \cdot disp_{c_i}^{-1} + k_2 \cdot \Delta hpwl^{-1}, & \text{if } disp_{c_i} > k_1 \\ 1 + k_2 \cdot \Delta hpwl^{-1}, & \text{if } disp_{c_i} \leq k_1 \\ -5, & \text{if } fail, \end{cases} \tag{2}$$

where [2]k_1 is a threshold of the inevitable displacement, and [2]k_2 normalizes the unit of ΔHPWL. If the displacement is less than the threshold, the reward is simplified to normalize the reward and prevent divergence of the value. If the legalizer fails to search pixels for the target cell, a penalty, -5, is imposed on the reward, and the corresponding episode is terminated, followed by the next episode. The failure penalty is empirically determined by considering the reward scale in various designs. As our RL algorithm updates the neural networks to maximize the cumulative reward, our reward function can minimize the displacement and HPWL.

B. Overall Framework

Our proposed deep RL framework consists of a legalizer environment and an asynchronous advantage actor-critic (A3C) agent (Fig. 3). We aim to minimize the displacement and HPWL from the global placement result by optimizing the cell priority in the legalization problem. In overall flow (Fig. 3a), given a global placement result (LEF/DEF) and design constraints, our legalizer first parses and initializes the geometric data. We then divide the design into multiple Gcells and consider each Gcell a subepisode in the training stage. In an episode, we proceed with the subepisodes for all Gcells, and the subepisodes are performed in the order of the number of cells in each Gcell to prevent legalization failure. In each subepisode, we train the neural network model to optimize the priority of cells in the corresponding Gcell. We extract and normalize state features of bins and cells (Fig. 3b). To normalize the features with various units, we leverage feature-wise normalization. The state features are embedded in a cell-wise DNN model, where 13 features of each cell are embedded in each model with the same parameters in parallel (Fig. 3c). The policy network constructs a cell priority vector to select the target cell by categorical sampling, and the value network provides an expected reward. The target cell is legalized using the pixel-wise search algorithm, and the features of affected cells and reward are updated. The state-action iteration is repeated until all movable cells in the subepisode are legalized. Therefore, we optimize the cell priority of various designs using the trained models.

[2]We set k_1 as the minimum width of the placement site. The placement sites of the contest technology [4] and 45 *nm* Nangate technology [31] are 200 and 190 *nm*, respectively. We set k_2 so that the $k_2 \cdot \Delta hpwl^{-1}$ term becomes between 0 and 1. If $\Delta hpwl$ is zero, the term becomes 1.

TABLE I
FEATURES OF EACH CELL AND SPECIFICATIONS

Feature	Specification
X_{c_i}	X-coordinate of the i-th cell (c_i)
Y_{c_i}	Y-coordinate of c_i
W_{c_i}	Width of c_i
H_{c_i}	Height of c_i
N_{c_i}	Number of nets connected to c_i
OV_{c_i}	Number of overlapped cells to c_i
OD_{c_i}	Avg. distance of the 2 nearest obstacles from c_i
CA_{B_i}	Total cell area in the bin region of c_i (B_i)
A_{B_i}	Area of placeable region in B_i
OV_{B_i}	Number of overlapped cells in B_i
DE_{B_i}	Bin density error of B_i
NC_{G_i}	Number of cells in the Gcell of c_i (G_i)
NLC_{G_i}	Number of legalized cells in G_i

G_i and B_i represent the Gcell and bin, respectively, where c_i is placed.

C. Training Flow

1) Training algorithm: Our framework leverages an A3C algorithm [22] which asynchronously optimizes the neural network gradient with multiple actor-critic agents. The asynchronous update decreases the correlation between data in the training process and achieves faster training convergence by using the latest policy function. We also use a mini-batch training technique [23] with batch size B and construct loss functions for efficient training. Algorithm 1 presents the training method of policy and value networks. In a single A3C agent, we construct a local network initialized with the global network parameters (Line 1). The training algorithm repeated 1,000

Algorithm 1: Training policy and value network

Inputs: Maximum episode (E_{max}), set of Gcells (G), batch size (B), global network parameters (θ, ϕ)

1 Initialize local network parameters $\theta', \phi' \leftarrow \theta, \phi$
2 **while** $ep < E_{max}$ **do**
3 **for all** $ep_{sub} \in G$ **do**
4 $t \leftarrow 1$
5 **while** *not* $ep_{sub}.done$ **do**
6 $s_t \leftarrow GetState()$
7 $a_t \leftarrow GetAction(s_t)$
8 $r_t \leftarrow GetReward(s_t, a_t)$
9 $s_{t+1} \leftarrow GetState()$
10 **if** t mod $B == 0$ **then**
11 $Loss \leftarrow GetLoss(s_{t:t+B}, a_{t:t+B-1})$
12 $\Delta\theta, \Delta\phi \leftarrow Backpropagation(Loss/B)$
13 $\theta, \pi \leftarrow \theta'+\Delta\theta, \phi+\Delta\phi$
14 $t \leftarrow t+1$

episodes (Line 2), and all subepisodes are repeated in an episode (Line 3). Each subepisode is performed by repeating steps until all movable cells in the corresponding Gcell are legalized (Line 5). In step t, the state features s_t of the movable cells are extracted, action a_t is performed through the legalizer, reward r_t is computed, and the next state features s_{t+1} are updated (Lines 6-9). At every B steps, the loss functions are constructed from the state-action trajectories, and the global network parameters are updated by backpropagation (Lines 10-13).

2) Loss function: We construct the loss function including (i) policy loss, (ii) value loss, and (iii) entropy loss as follows:

$$Loss = L_{policy} + \beta \cdot L_{value} + \eta \cdot L_{entropy}, \tag{3}$$

where β and η are loss coefficients.

$$L_{policy} = \sum_{t=1}^{B} -log\pi_\theta(a_t|s_t) \cdot Adv(s_t, a_t) \tag{4}$$

(i) The policy loss (Equation 4) updates the policy network with a type of cross-entropy loss which implies dissimilarity between the ideal and expected probabilities. Because the ground truth is absent in RL, the advantage function [22] is multiplied, which evaluates how the selected action is advantageous compared to the expected reward. The advantage function is computed as follows:

$$Adv(s_t, a_t) = Q^{\pi_\theta}(s_t, a_t) - V_\phi(s_t), \tag{5}$$

where the first term is the action value function with θ-parameterized policy function π_θ, and the second term is the ϕ-parameterized state value function. The action value function represents the γ-discounted expected reward (Equation 6), and the state value function represents the expected future reward.

$$Q^{\pi_\theta}(s_t, a_t) = \mathbb{E}_{\pi_\theta}\left[\sum_{k=0}^{B-1} \gamma^k \cdot r_{t+k} | s_t, a_t\right] \tag{6}$$

$$L_{value} = \sum_{t=1}^{B} SmoothL1(Q^{\pi_\theta}(s_t, a_t), V_\phi(s_t)) \tag{7}$$

(ii) The value loss (Equation 7) updates the value network from the error between the expected and discounted rewards. We leverage a smooth-L1 loss function, and unlike the L1 loss function, it is differentiable at all points and more robust to outliers than the mean squared error (MSE) function. The value loss is critical for training convergence because the stable reward estimation can assess the network models precisely.

$$L_{entropy} = \sum_{t=1}^{B} \sum_i \pi_\theta(i|s_t) \cdot log\pi_\theta(i|s_t) \tag{8}$$

(iii) The entropy loss (Equation 8) causes exploration and prevents convergence to the local optimum.

D. Cell-wise Network

We construct a cell-wise network structure to handle various designs with a different number of cells using DNNs (Fig. 4). However, the cell-wise DNNs may lack information on correlation with other cells [14], [15]. To alleviate this latent limitation, we supplement features for the surrounding environment (bin features) using a feature ablation technique [24]. Also, we implement a feature-wise L2 normalization, because this relative normalization can lead to a relative priority of cells. Our network architecture consists of cell-wise hidden layers, a policy network, and a value network. Each hidden layer has the same network parameters, which represents that the features of each cell are processed in parallel. Each cell-wise hidden layer comprises two sequential pairs of a 256-dimensional fully-connected (FC) layer and a ReLU activation layer. The policy network comprises an FC layer and a SoftMax activation layer, which constructs a cell priority vector. Meanwhile, the value network includes an FC layer and an average function, which provide a scalar value of the expected reward.

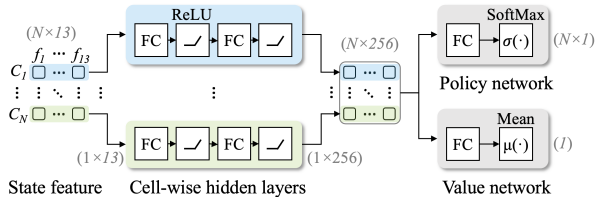


Fig. 4. Cell-wise neural network architecture of the policy and value networks, where N represents the number of movable cells. The dimension of the tensors is presented. The policy network constructs an $N \times 1$ cell priority vector, and the value network provides a scalar value of the expected reward.

E. Our RL Strategies

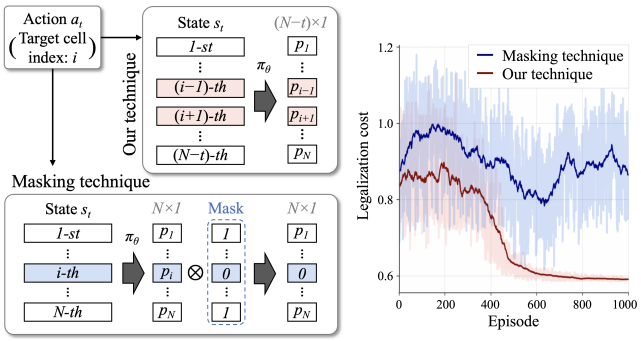
In this subsection, we present some challenges faced by our RL framework and strategies to resolve them.

1) Gcell subepisode: RL training, in general, deteriorates when the number of steps in an episode is large. For example, estimating future rewards becomes challenging for the value network, which impedes training convergence. Limiting the number of steps in a single episode stabilizes the training convergence. Therefore, we divide the designs into multiple Gcells with a limited number of cells, i.e., we partition an episode into multiple subepisodes. Gcell partitioning also enables our RL framework to be applied to advanced designs that are larger and have more cells. To minimize the impact of the order between Gcells on the order of cell legalization, we partition the designs into sufficiently large [3]Gcells. For example, the maximum number of Gcells is set to be 5×5 in benchmarks we used (Tables II–III).

2) Reduced-dimensional state: In our framework, all given cells in a subepisode should be sampled only once. Once a cell is legalized, it should not be selected again in later steps of the subepisode. Masking techniques are commonly used to resolve the problem, where the masking layer consists of Boolean elements; *False* in the indices of the legalized cells and *True* in the others (Fig. 5a). The output vector of the policy network is multiplied by a masking layer before the next action sampling based on the priority vector. In this method, the SoftMax layer is not used to prevent all cell priority elements from being zero in the later steps of the subepisode. However, the features of the legalized cells remain in the state, leading to unclear cell priority and, therefore, poor training convergence. We improve the quality of training by removing the legalized cells from the state at each step. The cell priority is determined based on only non-legalized cell features, excluding redundant features. To gather trajectories of the state, action, and reward from the reduced-dimensional state, we construct tractable loss tensors and concatenate them at each step. We compare the training convergence of our technique and the masking technique, which shows that our proposed method achieves a much faster convergence (Fig. 5b).

3) Network optimization: We use Bayesian optimization [27] to optimize the dimension and hyperparameters of our network for high training performance. The legalization cost is minimized by exploration and exploitation of the parameter configuration and objective function. We first optimize the network dimension of hidden layers in

[3]We empirically set Gcell size to $200K \times 200K$. Tables II–III verify that the impact of the partitioning is negligible, comparing [26] and [26]+G.



(a) Technique comparison (b) Training convergence

Fig. 5. (a) Comparison between our reduced-dimensional state technique (upper, red) and masking technique (lower, blue) to deal with the legalized cell C_i in step t. (b) Comparison of training convergence of the legalization cost, where x-axis represents the number of episodes, and the y-axis represents the legalization cost. A data smoothing method is used, and the light colors represent the variation of four A3C agents.

the range of 64 and 512, which is determined to be 256 as mentioned above. We then optimize hyperparameters, including learning rate α, discount factor γ, batch size B, value loss coefficient β, and entropy loss coefficient η. We limited the iterations of Bayesian optimization to 50 times and adopted the best hyperparameter configurations. As a result, we set α to 0.0003, γ to 0.98, B to 25, β to 0.9, and η to 0.002. Furthermore, we exploited the gradient clipping technique for stable training, by setting the maximum norm of the gradient to 0.1. An Adam optimizer [23] was used for the gradient descent algorithm.

4) Training and test: Circuit designs have considerably different characteristics in terms of cells, macro blocks, density, design area, and irregular fence regions. To ensure that our framework achieves outstanding legalization results, we divided the designs into multiple Gcells and extracted various geometric features. However, constructing Gcell subepisodes and extracting bin features are very time-consuming. For example, if a cell is legalized from bin B_i to bin B_j, the features of all cells in both B_i and B_j should be updated. Cell overlap check and bin density calculation are also expensive.

Using the trained models, our RL framework can achieve optimal solutions by continuously repeating episodes and updating the models for every design. However, this scheme leads to time-consuming legalization, which does not seem pragmatic. Therefore, we train RL models from some designs and test the performance of our *RL-Legalizer* on the other designs using the trained models without updating the networks. In the training process, initial state features are extracted and embedded into the trained model, and all movable cells in the design are legalized in order of the constructed cell priority. We limited the total training time to 72 hours. In the test process, a few seconds of time overhead is added for Gcell partitioning, initial feature extraction, and network processing.

IV. Experiments

A. Experimental Setup

1) Workstation and implementation: Our experiments are conducted on a CentOS Linux 7.9 with an AMD EPYC-Rome CPU with 20 cores at 2.8 GHz and 200 GB RAM. We trained our deep RL models with an Nvidia Titan RTX GPU. We implemented the legalizer in *C++* based on [26], and our RL agents in *PyTorch*. Geometric features were extracted using the boost library [28] for R-tree [25] in the legalizer. We embedded the *C++* legalizer into the *PyTorch* agent by using *ctypes* [29], which is a foreign function library for *Python*. Our A3C agents and network models can be checked in [30]. We employed four A3C agents to asynchronously update the neural networks using multi-processing.

2) Benchmarks: We trained and tested *RL-Legalizer* on open-sourced academic benchmarks, including the ICCAD-2017 contest benchmarks [4] and the OpenCores benchmarks [32]. The OpenCores benchmarks are implemented with 75% utilization and 1.0 aspect ratio in 45 *nm* Nangate technology [31], where 10% of cells in the cell library (LEF) are modified to be multi-heights while maintaining the cell area. First five columns in Table II–III present the characteristics of design benchmarks. The benchmarks have different numbers of cells (from $3K$ to $131K$) and design areas (from $95K\times95K$ to $900K\times900K$). We used 80% of benchmarks for training process and the other 20% for testing our trained model; *des_perf_a_md2*, *fft_a_md2*, *pci_bridge32_b_md1*, *keccak*, and *point_scalar_mult* are designated for the test, which have various design characteristics.

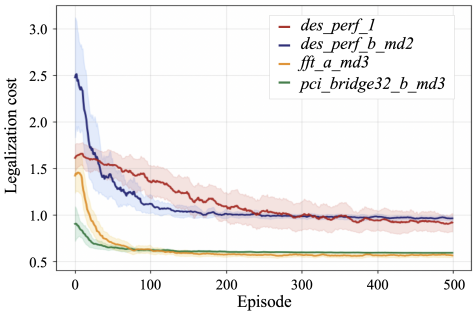


Fig. 6. Training convergence of legalization costs in the training process for four contest benchmarks. All designs except *des_perf_1* reach convergence with the minimum cost before 200 episodes.

B. Experimental Results

We evaluated the performance of the proposed *RL-Legalizer* with respect to the training convergence and the QoRs of cell legalization. We plotted learning curves of legalization costs based on [4] in some training benchmarks whose cost scales are similar (Fig. 6). The legalization costs of the four different designs decrease and converge as the episodes proceed. All designs except *des_perf_1* reach the convergence with the minimum cost before 200 episodes. The solution quality achieved, on average, 58% decreased legalization cost compared to the initial legalization costs, where the initial network model randomly initializes weight parameters.

We also evaluated the improved QoRs achieved by *RL-Legalizer* with respect to the average displacement, maximum displacement, and HPWL (Tables II–III). We first checked the legality of our legalization results and ensured that no design rule violations occurs for all benchmarks; the design rules include placement overlap, edge spacing, power alignment, placement sites, and region constraints. For training benchmarks, we reported the best results after training converged, and for test benchmarks, we reported the QoRs of the first testing results using the trained models. We then compared our legalization results with the results of the size-ordered legalization [26] to present the improvement of QoRs by our RL-based cell priority optimization. Besides, to show that the QoR overheads because of Gcell partitioning are negligible, we also reported the QoRs of [26]+G, where designs are partitioned into Gcells and the size-ordered cell legalization is executed.

Our *RL-Legalizer* outperformed the size-ordered methods on the overall QoR elements. In addition, we obtained the legal result from *des_perf_1*, whereas [26] failed to legalize all cells using the same search algorithm. [26]+G also obtained the legal result, albeit with poor QoRs, because the Gcell partitioning varied the cell priority in the entire design. This represents that the unoptimized cell priority results in legalization failure and bad QoRs for high-density designs. Compared to [26], *RL-Legalizer* achieved, on average, 15% reduced average displacement, 57% reduced maximum displacement, and 5% reduced HPWL for the training benchmarks, excluding the failed design. Compared to [26]+G, with similar QoRs but including *des_perf_1*, we achieved 17%, 79%, and 6% lower legalization cost in average displacement, maximum displacement, and HPWL, respectively. For the five test benchmarks, we achieved 10%, 21%, and 1% better QoRs, respectively, compared to the QoRs of [26]. Our *RL-Legalizer* achieved up to 20,920 *nm* less HPWL than [26] in

TABLE II
COMPARISON OF THE QORS BETWEEN OUR *RL-Legalizer* AND THE SIZE-ORDERED LEGALIZER [26] FOR THE TRAINING BENCHMARKS

Benchmark	# of cells	Area (e+11)	Density	# of Gcells	Avg. disp. (*nm*)			Max. disp. (*nm*)			HPWL (e+5)		
					[26]	[26]+G	Ours	[26]	[26]+G	Ours	[26]	[26]+G	Ours
des_perf_1	112,644	1.98	0.91	3×3	-	3,741	**2,121**	-	241,396	**16,896**	-	17.11	**13.40**
des_perf_a_md1	108,292	8.10	0.55	5×5	1,813	1,813	**1,376**	209,368	209,368	**121,460**	23.06	23.05	**22.42**
des_perf_b_md1	112,644	3.60	0.55	3×3	1,313	1,290	**1,187**	65,173	**64,247**	64,643	21.93	21.88	**21.70**
des_perf_b_md2	112,644	3.60	0.65	3×3	1,281	1,284	**1,200**	**38,720**	**38,720**	**38,720**	21.95	21.95	**21.86**
edit_dist_1_md1	130,661	5.21	0.67	4×4	1,304	1,303	**1,218**	127,218	25,018	**13,770**	40.75	40.75	**40.67**
edit_dist_a_md2	127,419	6.40	0.59	4×4	1,313	1,312	**1,217**	87,003	87,003	**32,800**	51.77	51.77	**51.60**
edit_dist_a_md3	127,419	6.40	0.57	4×4	2,178	2,258	**1,768**	100,003	101,772	**80,136**	54.99	55.18	**54.66**
fft_2_md2	32,281	1.17	0.83	2×2	1,806	1,799	**1,747**	20,170	**14,851**	22,729	4.96	4.95	**4.94**
fft_a_md3	30,631	6.40	0.31	4×4	1,014	1,016	**953**	21,476	21,476	**20,518**	9.63	9.63	**9.60**
pci_bridge32_a_md2	29,521	1.60	0.58	2×2	2,639	2,692	**2,396**	102,385	103,405	**76,163**	6.37	6.39	**6.27**
pci_bridge32_b_md1	28,920	6.40	0.26	4×4	1,489	1,500	**1,393**	68,004	68,004	**67,804**	6.85	6.85	**6.81**
pci_bridge32_b_md2	28,920	6.40	0.18	4×4	1,362	1,363	**1,277**	**85,808**	**85,808**	**85,808**	5.98	5.97	**5.93**
pci_bridge32_b_md3	28,920	6.40	0.22	4×4	1,688	1,689	**1,537**	107,536	107,536	**79,715**	6.18	6.17	**6.11**
aes_cipher_top	10,006	0.16	0.75	1×1	1,756	1,742	**1,548**	8,965	8,828	**8,765**	0.85	0.85	**0.84**
des3	42,788	1.02	0.74	2×2	2,660	2,676	**2,292**	**36,852**	62,695	40,695	2.63	2.64	**2.61**
eth_top	41,871	1.09	0.74	2×2	1,719	1,713	**1,478**	7,434	6,745	**6,675**	3.35	3.35	**3.31**
jpeg_encoder	35,688	0.83	0.75	2×2	1,841	1,803	**1,632**	28,643	**24,054**	**24,054**	1.40	1.39	**1.38**
mc_top	4,576	0.12	0.74	1×1	1,612	1,547	**1,511**	8,743	7,782	**7,329**	0.38	0.38	**0.37**
nova	136,961	3.46	0.71	4×4	1,671	1,656	**1,493**	8,998	**7,749**	7,890	10.37	10.35	**10.20**
sasc_top	442	0.01	0.75	1×1	3,269	3,243	**2,137**	7,464	6,546	**5,823**	0.03	0.03	**0.02**
spi_top	1,486	0.04	0.75	1×1	1,963	1,958	**1,586**	5,933	6,089	**5,160**	0.12	0.12	**0.11**
usb_phy	321	0.01	0.75	1×1	2,336	2,308	**1,798**	5,454	5,044	**5,031**	0.02	0.02	**0.02**
wb_conmax_top	18,961	0.43	0.57	2×2	1,460	1,452	**1,334**	5,183	4,999	**4,823**	4.62	4.62	**4.58**
Norm avg.	-				1.15	1.17	**1.00**	1.57	1.79	**1.00**	1.05	1.06	**1.00**

*The contest benchmarks [4] are in white regions, and the OpenCores benchmarks [32] are in gray regions in Tables II–III.
*Note that the normalized average of the QoRs of [26] are computed excluding the failed design.
*The best results are highlighted in **bold** in Tables II–III.

TABLE III
COMPARISON OF THE QORS BETWEEN OUR *RL-Legalizer* AND THE SIZE-ORDERED LEGALIZER [26] FOR THE TEST BENCHMARKS

Benchmark	# of cells	Area (e+11)	Density	# of Gcells	Avg. disp. (*nm*)			Max. disp. (*nm*)			HPWL (e+5)			Runtime (s)		
					[26]	[26]+G	Ours	[26]	[26]+G	Ours	[26]	[26]+G	Ours	[26]	[26]+G	Ours
des_perf_a_md2	108,292	8.10	0.56	5×5	1,704	1,703	**1,534**	132,798	132,798	**102,119**	23.08	23.08	**22.87**	**3.93**	4.17	19.44
fft_a_md2	30,631	6.40	0.32	4×4	1,081	1,080	**1,013**	**66,699**	**66,699**	68,699	11.06	11.06	**11.04**	**0.98**	1.06	6.15
pci_bridge32_a_md1	29,521	1.60	0.50	2×2	1,494	1,488	**1,408**	**85,143**	**85,143**	**85,143**	4.78	4.79	**4.74**	**0.87**	0.90	3.68
keccak	24,902	0.52	0.75	2×2	1,144	1,141	**1,073**	5,879	5,703	**5,700**	3.82	3.81	**3.81**	**0.43**	0.48	5.42
point_scalar_mult	51,294	1.14	0.75	2×2	1,860	1,823	**1,513**	37,814	34,835	**21,864**	4.06	4.04	**3.96**	**1.09**	1.21	10.39
Norm avg.	-				1.10	1.10	**1.00**	1.21	1.17	**1.00**	1.01	1.01	**1.00**	**0.16**	0.18	1.00

the test benchmarks, although the improvement rate appears low due to the large unit scale. Moreover, only a few additional seconds are needed for the testing process with the trained model, with about 80% of the time spent on the feature extraction phase. The comparison results represent that our *RL-Legalizer* optimizes cell priority well for various designs with different characteristics, which is crucial for better QoRs in mixed-height cell legalization.

V. CONCLUSION

We proposed an RL-based legalizer that optimizes the cell priority in mixed-height standard cell legalization. We used an A3C algorithm and implemented several RL techniques to improve the performance of the training and the quality of cell legalization. Compared to the size-ordered cell legalization, our proposed framework improved the QoRs in terms of the average displacement, maximum displacement, and wirelength by 17%, 79%, and 6%, respectively. Moreover, our RL framework can be adapted to other sequential legalization algorithms. In our future work, we are pursuing to obtain improved legalization results in a short time by improving the legalization algorithm and utilizing various RL algorithms. In addition, we are going to consider more design constraints, such as multi-threshold voltage and minimum implant areas.

REFERENCES

[1] J. Chen, Z. Zhu, W. Zhu and Y.-W Chang, "Toward Optimal Legalization for Mixed-Cell-Height Circuit Designs," *Proc. IEEE/ACM Design Automation Conference*, 2018, pp. 1–6.

[2] 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," *Proc. SPIE 7268, Smart Structures, Devices, and Systems IV*, 2008, pp. 72680C–72680C.

[3] M. Khasawneh and P. H. Madden, "What's So Hard About (Mixed-Size) Placement?," *Proc, ACM International Symposium on Physical Design*, 2022, pp. 57–64.

[4] N. K. Darav, I. S. Bustany, A. Kennings and R. Mamidi, "ICCAD-2017 CAD Contest in Multi-Deck Standard Cell Legalization and Benchmarks," *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2017, pp. 867–871.

[5] C.-H.Wang, Y.-Y.Wu, J. Chen, Y.-W. Chang, S.-Y. Kuo,W. Zhu and G. Fan, "An Effective Legalization Algorithm for Mixed-Cell-Height Standard Cells," *Proc. IEEE Asia and South Pacific Design Automation Conference*, 2017, pp. 450–455.

[6] Z. Zhu, J. Chen, W. Zhu and Y.-W. Chang, "Mixed-Cell-Height Legalization Considering Technology and Region Constraints," *IEEE Trans. on CAD* 39(12) (2020), pp. 5128–5141.

[7] 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," IEEE/ACM Design Automation Conference, 2018, pp. 1–6.

[8] H. Li, W.-K. Chow, G. Chen, B. Yu and E. F. Y. Young, "Pin-Accessible Legalization for Mixed-Cell-Height Circuits," *IEEE Trans. on CAD* 41(1) (2021), pp. 143–154.

[9] S. Do, M. Woo and S. Kang, "Fence-Region-aware Mixed-Height Standard Cell Legalization," *Proc. ACM Great Lakes Symposium on VLSI*, 2019, pp. 259–262.

[10] J. Chen, Z. Zhu, W. Zhu and Y.-W. Chang, "A Robust Modulus-Based Matrix Splitting Iteration Method for Mixed-Cell-Height Circuit Legalization," *ACM Trans. on Design Automation Electronic Systems* 26(2) (2021), pp. 1–28.

[11] R. Netto, S. Fabre, T. A. Fontana, V. Livramento, L. L. Pilla, L. Behjat and J. L. Guntzel, "Algorithm Selection Framework for Legalization Using Deep Convolutional Neural Networks and Transfer Learning," *IEEE Trans. on CAD* 41(5) (2021), pp. 1481–1494.

[12] H. Ren, S. Godil, B. Khailany, R. Kirby, H. Lia, S. Nath, J. Raiman and R. Roy, "Optimizing VLSI Implementation with Reinforcement Learning – ICCAD Special Session Paper," *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2021, pp. 1–6.

[13] Y.-C. Lu, S. Nath, V. Khandelwal and S. K. Lim, "RL-Sizer: VLSI Gate Sizing for Timing Optimization using Deep Reinforcement Learning," *Proc. IEEE/ACM Design Automation Conference*, 2021, pp. 733–738.

[14] H. Liao, W. Zhang, X. Dong, B. Poczos, K. Shimada and L. B. Kara, "A Deep Reinforcement Learning Approach for Global Routing." *ASME Journal of Mechanical Design* 142(6) (2020), p. 061701.

[15] Y. Lin, T. Qu, Z. Lu, Y. Su and Y. Wei, "Asynchronous Reinforcement Learning Framework and Knowledge Transfer for Net Order Exploration in Detailed Routing," *IEEE Trans. on CAD* 41(9) (2021), pp. 3132–3142.

[16] H. Liao, Q. Dong, X. Dong, W. Zhang, W. Zhang, W. Qi, E. Fallon and L. B. Kara, "Attention Routing: Track-Assignment Detailed Routing using Attention-based Reinforcement Learning," *Proc. ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2020, p. V11AT11A002.

[17] X. Ju, K. Zhu, Y. Lin and L. Zhang, "Asynchronous Multi-Nets Detailed Routing in VLSI using Multi-Agent Reinforcement Learning," *Proc. IEEE International Conference on Network Intelligence and Digital Content*, 2021, pp. 250–254.

[18] A. Agnesina, K. Chang and S. K. Lim, "VLSI Placement Parameter Optimization using Deep Reinforcement Learning," *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2020, pp. 1–9.

[19] R. Kirby, K. Nottingham, R. Roy, S. Godil and B. Catanzaro, "Guiding Global Placement With Reinforcement Learning," *arXiv preprint arXiv:2109.02631* (2021).

[20] U. Mallappa, S. Pratty and D. Brown, "RLPlace: Deep RL Guided Heuristics for Detailed Placement Optimization," *Proc. IEEE Design, Automation and Test in Europe Conference*, 2022, pp. 120–123.

[21] A. B. Kahng and Q. Wang, "Implementation and Extensibility of an Analytic Placer," *Proc. ACM International Symposium on Physical Design*, 2004, pp. 18–25.

[22] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver and K. Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning," *Proc. PMLR International Conference on Machine Learning*, 2016, pp. 1928–1937.

[23] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv preprint arXiv:1412.6980* (2014).

[24] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," *Proc. Advances in Neural Information Processing Systems*, 2017, pp. 4765–4774.

[25] A. Guttman, "R-trees: A Dynamic Index Structure for Spatial Searching", *Proc. ACM SIGMOD International Conference on Management of Data*, 1984, pp. 47–57.

[26] "OpenDP: Open Source Detailed Placement Engine version 0.1.0," *https://github.com/The-OpenROAD-Project/OpenDP/tree* [Online].

[27] M.-C. Ruben, "Bayesopt: A bayesian optimization library for nonlinear optimization, experimental design and bandits," Journal of Machine Learning 15(1) (2014), pp. 3735–3739.

[28] "Boost Geometry Library," *https://www.boost.org/doc/libs/1_65_1/libs/geometry/doc/html/index.html* [Online].

[29] "Ctypes – A Foreign Function Library for Python," *https://docs.python.org/3/library/ctypes.html* [Online].

[30] "RL-Legalizer: Reinforcement Learning-based Mixed-Height Standard Cell Legalization," *https://github.com/syunlee/RL-Legalizer* [Online].

[31] NanGate, Inc., "Nangate 45nm Open Cell Library," *www.nangate.com/openlibrary* [Online].

[32] "OpenCores: Open Source IP-Cores," *http://www.opencores.org* [Online].