RSPlace: Rotation Sensing Macro Placement via Bidirectional Tree Expansion

Tianyi Liu, Yaxin Xu, Lin Geng, Ningzhong Liu*, Han Sun, Yu Wang

College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics
{tianyi.liu, yaxin.xu, sunhan, wangyucs}@nuaa.edu.cn, beargl@163.com, lnz_nuaa@163.com

Abstract

Macro placement is a crucial subproblem of chip design, focusing on determining the locations of numerous macros while minimizing multiple metrics. In recent years, reinforcement learning (RL) has gained traction as a favorable technique to improve placement performance. However, existing RL-based placers ignore the orientation of macros, resulting in the state space constrained to two-dimensional discrete coordinates and greatly restricting the exploration opportunities. To address this issue, we propose a novel macro placement method, RSPlace, which guides the bidirectional expansion of the global search tree to offer the RL agent more exploration opportunities, incorporating rotation into the RL-based macro placement solution for the first time. RSPlace intelligently determines the optimal rotation angle to maximize placement benefits by leveraging rotation sensing and placement perturbations. Extensive experiments demonstrate that taking the macro orientation into account substantially broadens the feasible locations and effectively reduces the half-perimeter wirelength (HPWL), thus ensuring that our approach significantly improves the optimization effect compared to the state-of-the-art method.

Extended version —
https://github.com/liutianyi123/RSPlace

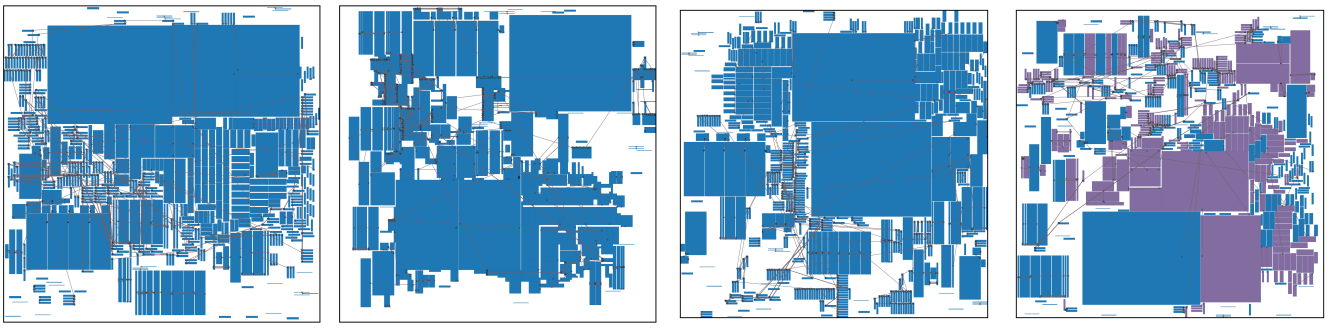
Introduction

The rapid advancement of very large-scale integration (VLSI) technology presents significant challenges for leveraging existing Electronic Design Automation (EDA) tools in chip design (Markov, Hu, and Kim 2012), particularly during the placement phase, where numerous circuit modules are arranged on a constrained layout area. Macro placement, as a critical subproblem, involves optimally positioning large circuit modules to minimize multiple metrics, including half-perimeter wirelength (HPWL), density, and congestion.

Existing methods for macro placement can be categorized into two paradigms: optimization-based methods and reinforcement learning (RL)-based methods. Optimization-based methods formulate macro placement as a combinatorial optimization problem and employ classical techniques such as simulated annealing (SA), quadratic programming (QP), and evolutionary algorithm (EA). In contrast, RL-based methods formulate macro placement as a Markov Decision Process (MDP) (Sutton and Barto 2018), where a RL agent learns optimal placement strategies through continuous interaction with the environment. In Graph Placement (Mirhoseini et al. 2021) and DeepPR (Cheng and Yan 2021), the policy network is guided by a hypergraph, while MaskPlace (Lai, Mu, and Luo 2022) views macro placement as a visual representation problem, utilizing a wire mask to capture HPWL increments and a position mask to identify feasible locations. In ChiPFormer (Lai et al. 2023), macro placement is based on offline RL, which can efficiently generalize to unseen placement tasks. A recent breakthrough is EfficientPlace (Geng et al. 2024), which employs a search tree to explore placement solutions and a RL agent to execute local policy learning, demonstrating remarkable placement quality. However, the methods mentioned above only construct the state space from two-dimensional discrete coordinates, which is not sufficient and restricts the exploration opportunities. In other words, when the solution space is expanded through macro rotation, the solutions generated by previous methods are inherently sub-optimal due to their limitations in action representation.

To address this issue, we propose a novel macro placement method called RSPlace, which directs the bidirectional expansion process of the global search tree for offering RL agent more exploration opportunities. Unlike previous methods, the state space of RSPlace includes both discrete coordinates and rotation angles on a two-dimensional canvas. On one hand, given a macro to be placed, RSPlace generates its position mask and wire mask under different rotation angles, then identifies the mask combination that results in the minimal HPWL increment, thereby determining the optimal macro rotation strategy. On the other hand, RSPlace perturbs the placement solution by adjusting the orientation of macros after each loop (multiple RL rollouts). For each macro in the new configuration, adjustments are made to preserve the non-overlapping property. These improvements substantially increases the number of feasible positions and significantly reduces HPWL increments. Figure 1 shows the visualization of macro placement results on the *adaptec2* circuit benchmark. RSPlace effectively reduces the HPWL while maintaining 0% overlap. Compared to previous RL-

*Corresponding author.



(a) MaskPlace (b) WireMask-EA (c) EfficientPlace (d) RSPlace

Figure 1: Visualization of macro placement result on the circuit benchmark *adaptec2*. The macros are visualized by blue or purple rectangles, where purple rectangles in (d) mean that macros have been rotated. The wires are represented by brown lines. We compare the proposed approach RSPlace with three representative methods, including (a) MaskPlace (Lai, Mu, and Luo 2022) (HPWL = 79.98×10^5, Overlap = 0%), (b) WireMask-EA (Shi et al. 2023) (HPWL = 55.32×10^5, Overlap = 0%), (c) EfficientPlace (Geng et al. 2024) (HPWL = 43.35×10^5, Overlap = 0%), (d) RSPlace (ours) (HPWL = 38.00×10^5, Overlap = 0%). The smaller the value of HPWL, the better.

based methods, this paper has three main **contributions**:

- **The New Dimension.** We incorporate rotation sensing and placement perturbations into the expansion process of the global search tree to select the optimal rotation angle, integrating the macro orientation into the RL-based macro placement problem for the first time.
- **Comprehensive Analysis.** The potential positive impacts of rotation on macro placement are comprehensively analyzed, including generating more feasible positions and fewer HPWL increments.
- **Outstanding Performance.** Extensive experiments demonstrate that our approach achieves better performance compared to existing work, even outperforming the recent state-of-the-art method.

Background

In this section, we first introduce the key metrics for evaluating placement outcomes. Later, we will briefly review existing macro placement methods, including optimization-based methods and RL-based methods.

Macro Placement

Optimizing the power, performance, and area (PPA) metrics is the ultimate goal of chip design, with macro placement being a critical sub-stage. Specifically, the macro placement problem involves taking a netlist $H = (V, E)$ as input. Here, $V = \{M_j\}_{j=1}^{k}$ represents the set of macros, while E consists of hyper-edges. Each hyper-edge $e_i \in E$ captures the connectivity between multiple circuit modules.

HPWL, an approximate method for wirelength estimation, is extensively utilized due to its high accuracy and computational efficiency (Rabaey, Chandrakasan, and Nikolic 2002; Chen et al. 2006). Given a hyper-edge with serval pins, its HPWL is equal to the half perimeter of the smallest rectangle (bounding box) enclosing these pins. Further, for a netlist H, the HPWL under the placement solution s is defined as:

$$\text{HPWL}(s, H) = \sum_{e_i \in E} (w_i + h_i) \tag{1}$$

where $w_i = \max_{M_j \in e_i} x_j - \min_{M_j \in e_i} x_j$ and $h_i = \max_{M_j \in e_i} y_j - \min_{M_j \in e_i} y_j$ denote the width and height of the bounding box enclosed by the hyper-edge e_i. The bottom-left and top-right corners of the bounding box are represented by the coordinates $(\min_{M_j \in e_i} x_j, \min_{M_j \in e_i} y_j)$ and $(\max_{M_j \in e_i} x_j, \max_{M_j \in e_i} y_j)$. To improve readability, this metric will be simplified to HPWL(s) in the following sections.

Congestion arises from overcrowding during placement, which can result in decreased chip performance and signal transmission obstruction. This paper employs the RUDY congestion estimation method (Spindler and Johannes 2007). In this approach, every grid accumulates the inverses of the height and width of all the bounding boxes covering it, then extracts the average of the p maximum values across all grids. The congestion of grid $g_{m,n}$ is formulated as:

$$\text{Congestion}(g_{m,n}) = \sum_{e_i \in E(g_{m,n})} (\frac{1}{w_i} + \frac{1}{h_i}) \tag{2}$$

where $E(g_{m,n})$ denotes bounding boxes overlaid on grid $g_{m,n}$.

Optimization-based Methods

Over the past few decades, optimization-based methods have remained the predominant in macro placement, including partitioning-based methods, black-box-optimization (BBO) methods, and analytical methods.

Partitioning-based methods (Khatkhate et al. 2004; Roy et al. 2006) employ an iterative approach that divides circuits and placement areas into multiple clusters and sub-regions, aiming to minimize inter-cluster connections while assigning each cluster to its corresponding sub-region. Although

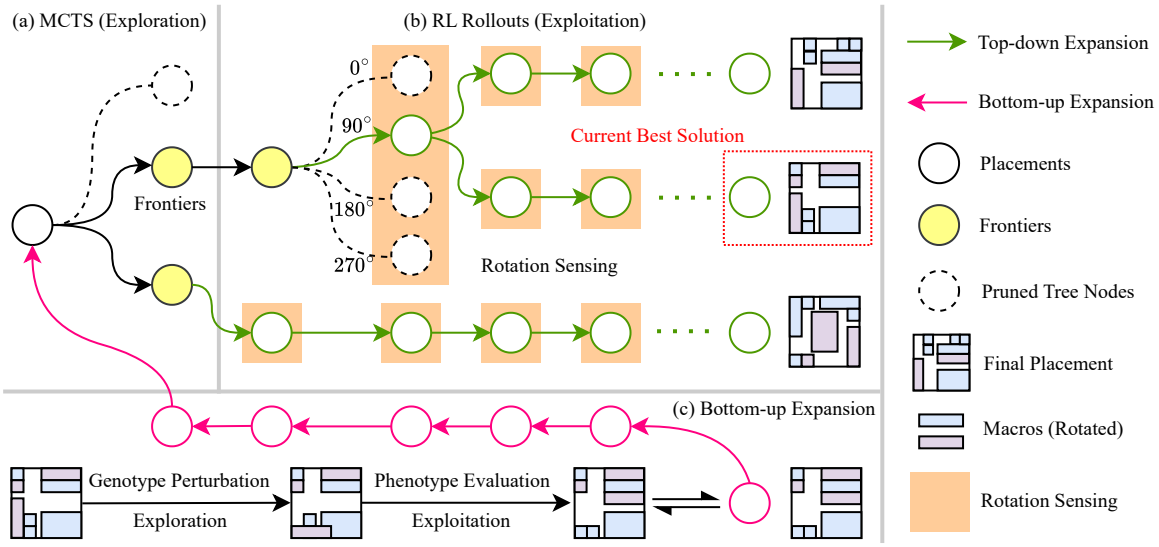


Figure 2: Overview of RSPlace. (a) A global search tree, guided by MCTS. (b) Top-down tree expansion driven by multiple RL rollouts. (c) Bottom-up tree expansion, incorporating genotype perturbation and phenotype evaluation.

advanced hypergraph partitioning techniques (Karypis et al. 1997) have proven effective in optimizing placement within individual partitions, the intricate topological structure of netlists poses significant challenges in dividing them into relatively independent parts.

BBO methods typically adopt SA to optimize macro placement solutions by applying random perturbations to their data structures, such as MP-tree (Chen et al. 2007), MDP-tree (Liu et al. 2019), and extension of corner sequence (ECS) (Chiou et al. 2016). A key advantage of these methods lies in their ability to handle problems where the objective metric lacks an analytical formulation or is non-differentiable. However, due to the nature of random perturbations, these approaches suffer from low sample efficiency, resulting in high computational costs as the problem scale increases.

Analytical methods optimize the placement problem by modeling it as an analytical function of module coordinates. Quadratic methods (Viswanathan et al. 2007; Kim and Markov 2012) formulate it as a convex quadratic problem, while nonlinear methods like NTUPlace3 (Chen et al. 2008), RePlace (Cheng et al. 2018), and DREAMPlace (Lin et al. 2020) approximate HPWL using differentiable functions, such as the log-sum-exp wirelength (Naylor, Donelly, and Sha 2001). Although these methods are efficient, they often lead to overlaps due to treating overlap avoidance as a soft constraint.

RL-based Methods

In recent years, researchers are also endeavoring to model macro placement as a sequential decision problem, proposing numerous methods based on RL. Graph Placement (Mirhoseini et al. 2021), published in *Nature*, stands out as the pioneering end-to-end macro placement learning approach. It segments the canvas into discrete grids, where the RL agent sequentially determines the location of each macro using discrete coordinates on the two-dimensional canvas. Note that rewards are sparse in Graph Placement, implying that only the macro placed in the last step receives a reward. Consequently, training the model becomes exceptionally challenging. Similar to Graph Placement (Mirhoseini et al. 2021), DeepPR (Cheng and Yan 2021) also represents a netlist as a hypergraph, but incorporates intrinsic rewards to encourage exploration. However, the macro placement solutions generated by DeepPR still face the issue with overlap. To address this problem, MaskPlace (Lai, Mu, and Luo 2022) views macro placement as a pixel-level visual representation problem. It defines the reward function as a dense reward about HPWL and ensures the non-overlapping property through position masks. Additionally, some researchers have developed a macro placement architecture based on offline RL, which can efficiently generalize to unseen placement tasks (Lai et al. 2023). A recent breakthrough of RL-based methods is EfficientPlace (Geng et al. 2024), which employs the bi-level framework for macro placement, including a global search tree to explore placement solution and a RL agent to execute local policy learning, demonstrating remarkable placement quality.

Approach

This section is organized as follows. We first introduce the key components of our framework, and then analyzes the potential impact of the macro orientation, including generating more feasible positions and fewer HPWL increments.

RSPlace Framework

Overall Architecture. Our framework is illustrated in Figure 2, and the details of the algorithm are presented in Algorithm 1 and Appendix A.1. RSPlace consists of three main components: (a) A high-level global search tree that guides and maintains a potential set of nodes (frontiers) through Monte Carlo Tree Search (MCTS). (b) Top-down local policy learning driven by RL agent, leveraging promising nodes along with rotation sensing to efficiently explore placement strategies. (c) Bottom-up tree expansion, which integrates genotype perturbation and phenotype evaluation for the joint optimization of the existing placement solution. The three components work in tandem to achieve a dynamic balance between exploration and exploitation, leading to superior placement performance.

Solution Representation & Global Search Tree. For a netlist containing n macros, previous RL-based placers define the placement solution as a set of coordinate sequences on a two-dimensional canvas, i.e., $s_n = \{(x_1, y_1), ..., (x_n, y_n)\}$, where (x_i, y_i) denotes the location of the ith macro. In this paper, we incorporate rotation into the placement process, representing the placement of the ith macro as a triple (x_i, y_i, θ_i). This flexibility enhances the opportunities for the agent to identify superior placement solutions. The detailed comparison between RSPlace and existing methods can be found in Appendix A.2. RSPlace has a larger state space, allowing the RL agent to explore a wider range of possibilities.

Since each node in the global search tree $\mathcal{T}$ represents an intermediate placement state, we define the empty canvas as $s_0 = \{\}$, which serves as the root node of tree $\mathcal{T}$. The final placement is denoted as $s_n = \{(x_1, y_1, \theta_1), ..., (x_n, y_n, \theta_n)\}$, corresponding to a leaf node. Further, we define $\mathcal{V}(s)$ as the set of leaf nodes for a given tree node s. Hence, the macro placement in this paper is formulated as:

$$\min_{s \in \mathcal{X}} \text{HPWL}(s) \tag{3}$$

where $\mathcal{X} = \mathcal{V}(s_0) \cap \mathcal{P}$ and $\mathcal{P}$ represents the set of non-overlapping placement solutions.

Node Selection. The concept of frontiers is introduced by EfficientPlace (Geng et al. 2024) to mitigate policy learning bias. It refers to a set of the most promising intermediate placement states, denoted as $\mathcal{F}$. Under this constraint, a node $s \in \mathcal{F}$ is randomly selected during each RL rollout, and $\mathcal{F}$ is updated after a predefined number of the expansion. The update of frontiers is based on UCT (Browne et al. 2012) algorithm:

$$S(s) = Q(s) + \alpha \cdot \frac{1}{\sqrt{N(s)}} \tag{4}$$

where $S(s)$ is the score of a tree node s and $Q(s) = \max_{s_n \in \mathcal{V}(s)} -\text{HPWL}(s_n)$. Additionally, $N(s)$ indicates the visit time of a node s, and α is the coefficient to balance exploration and exploitation.

RSPlace maintains a frontier set $\mathcal{F}$ while introducing an important enhancement that keeps both the locations and orientations of macros fixed along the path from the root to any selected frontier node s. This constraint enables more focused local policy learning while preserving the benefits of rotational flexibility in subsequent placement decisions.

Algorithm 1: RSPlace

Input: Netlist, search tree $\mathcal{T}$, RL agent
Parameters: Number of loops K_{TS}, K_{RL}, K_P
Output: Best placement solution

Initialize $\mathcal{T}$ with s_0, and $\mathcal{F} \leftarrow s_0$
for $i = 1$ **to** K_{TS} **do**
  // Top-down tree expansion
  for $j = 1$ **to** K_{RL} **do**
    Select a frontier node $s \in \mathcal{F}$
    Execute a RL rollout from s by rotation sensing
    Expand $\mathcal{T}$ with new nodes in the rollout
    Backpropagate the rollout results
    Update the RL agent
  end for
  // Bottom-up tree expansion
  for $k = 1$ **to** K_P **do**
    Select a leaf node by Eq. 7
    Execute the genotype perturbation by Eq. 8
    Execute the phenotype evaluation
    Expand $\mathcal{T}$ with the new leaf node
    Backpropagate the phenotype evaluation result
  end for
  Update the frontiers $\mathcal{F}$
end for
Return: The best placement solution in $\mathcal{T}$

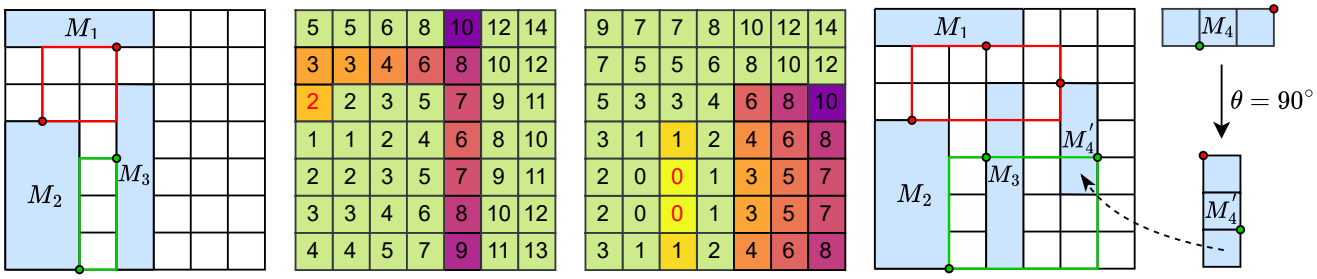
Top-down Tree Expansion & Rotation Sensing. Each top-down tree expansion starts from a selected frontier node and terminates at a leaf node. This process is modeled as a MDP in RSPlace, where the RL agent continuously interacts with the environment to determine the placement solution for each macro. The agent is updated using proximal policy optimization (PPO) (Schulman et al. 2017).

In RSPlace, the direction of top-down tree expansion is guided by the rotation sensing module. The purpose of this module is to determine the macro orientation that minimizes the HPWL increment and identifies the best mask combination. Since the calculation of wire mask depends on the pin's relative position (Δ_x, Δ_y), adjustments must be made accordingly. The new position coordinates of the pin within the macro are calculated using the following formula:

$$\begin{bmatrix} \Delta'_x \\ \Delta'_y \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \Delta_x \\ \Delta_y \end{bmatrix} \tag{5}$$

where θ is the rotation angle, and (Δ'_x, Δ'_y) represent the new position coordinates of the pin. The two-dimensional canvas is composed of discrete grids, so there are only four accessible rotation angles. The overall architecture of the RL agent is presented in Appendix A.3, which is integrated into the top-down tree expansion process.

Given a canvas of size $N \times N$ and time step t, we denote the position mask as $P(\theta_t) \in \{0, 1\}^{N \times N}$ and the normalized wire mask as $W(\theta_t) \in [0, 1]^{N \times N}$. Then, the optimal



(a) Placement example (b) Masks for $\theta = 0°$ (c) Masks for $\theta = 90°$ (d) Calculation of HPWL

Figure 3: Illustration of rotation benefits. Considering macro orientation significantly increases the number of feasible positions while reducing HPWL increments.

rotation angle θ_t^* can be defined as:

$$\theta_t^* = \arg\min_{\theta_t} \min(P(\theta_t) + W(\theta_t)) \tag{6}$$

where $\min(\cdot)$ represents the minimum element in a matrix, $\theta_t \in \{0°, 90°, 180°, 270°\}$, and value "0" in the position mask indicates a feasible position.

Bottom-up Tree Expansion & Placement Perturbation. During the RL rollout process, the macro orientation is determined by Eq. 6. However, maximizing the current placement benefit does not necessarily yield the optimal placement solution. To address this, RSPlace employs a bottom-up tree expansion approach. Specifically, after completing several RL rollouts, a new set of leaf nodes is generated. RSPlace selects the node with the highest Q value as the placement to be perturbed, which is denoted as s_n^*. This selection process can be formalized as follows:

$$s_n^* = \arg\max_{s_n \in \mathcal{V}(s_0)} Q(s_n) \tag{7}$$

For the selected node s_n^*, we construct a rotation candidate set $\Omega \subseteq V$ through random sampling of macros. The genotype perturbation for each $M_t \in \Omega$ is defined by Eq. 8:

$$\theta_t \leftarrow (\theta_t + \Delta\theta) \mod 360° \tag{8}$$

where $\Delta\theta \in \{90°, 180°, 270°\}$.

It is obvious that overlapping between macros may occur after the genotype perturbation, causing the original placement solution to likely no longer be optimal for achieving the minimum HPWL increment. To preserve the non-overlapping property and reduce the HPWL value, it is essential to adjust these locations. Assume that the locations of $M_1, \ldots, M_{i-1}$ have already been updated. When updating the location of M_i, the position mask and wire mask are recalculated based on this configuration. Then, M_i is moved to the nearest feasible grid that results in the least HPWL increment. Once all macros have been adjusted, a new leaf node is generated, which represents the phenotype evaluation of perturbed s_n^*. In fact, this new leaf node can be expanded back to the root node, which can be interpreted as a bottom-up tree expansion. Next, we will provide a detailed analysis of the potential impact of the macro orientation.

Analysis of Macro Orientation

Rotation provides more feasible positions. Figures 3(b) and 3(c) demonstrate the placement masks for orientations $\theta \in \{0°, 90°\}$ at time step $t = 4$, where three pre-placed macros are indicated by the blue grids in Figure 3(a). The masks for $\theta \in \{180°, 270°\}$ can be derived analogously. For the case of $\theta = 0°$, macro M_4 is to be placed. By using its bottom-left corner grid as the reference position and excluding unfeasible positions (marked in green) that either violate boundary constraints or cause overlap, we obtain 12 valid positions in Figure 3(b). In contrast, when rotated to $\theta = 90°$, the transformed macro M_4' owns 19 feasible positions in Figure 3(c). This clearly reveals that rotational flexibility opens up more possibilities for macro placement.

Rotation produces fewer HPWL increments. Figures 3(b) and 3(c) illustrate the HPWL increment for each grid, where the bottom-left corner of the canvas is defined as coordinate $(0, 0)$. The minimum HPWL increment is 2, achieved when the bottom-left corner of M_4 is placed at $(0, 4)$. However, if rotation is performed, placing the rotated macro M_4' at $(2, 1)$ and $(2, 2)$ reduces the HPWL increment to 0. For clarity, Figure 3(d) depicts the case where M_4' is placed at $(5, 2)$, resulting in HPWL increments of 2 and 3 for two hyper-edges, due to the inclusion of pins in their bounding boxes. The total HPWL increment in this scenario is 5. This example highlights that minimizing the HPWL increment requires positioning pins of the same hyper-edge in close proximity. Since rotation expands the set of possible placements, there is a greater advantage in finding positions that minimize HPWL increments.

Experiments

We compare RSPlace with current representative macro placement approaches. SA (Cheng et al. 2023) and WireMask-EA (Shi et al. 2023) are optimization-based methods. Graph Placement (Mirhoseini et al. 2021), DeepPR (Cheng and Yan 2021), MaskPlace (Lai, Mu, and Luo 2022), ChiPFormer (Lai et al. 2023) and EfficientPlace (Geng et al. 2024) are RL-based methods. Following the previous work (Geng et al. 2024), the ISPD2005 (Nam et al. 2005) dataset is selected as our benchmark. Further details of ISPD2005 and our experimental configuration are available in Appendix A.4 and Appendix A.5.

Method	adaptec1	adaptec2	adaptec3	adaptec4	bigblue1	bigblue3
Graph Placement (50k)	30.01 ± 2.98	351.71 ± 38.20	358.18 ± 13.95	151.42 ± 9.72	10.58 ± 1.29	357.48 ± 47.83
DeepPR (3k)	19.91 ± 2.13	203.51 ± 6.27	347.16 ± 4.32	311.86 ± 56.74	23.33 ± 3.65	430.48 ± 12.18
MaskPlace (3k)	7.62 ± 0.67	75.16 ± 4.97	100.24 ± 13.54	87.99 ± 3.25	3.04 ± 0.06	90.04 ± 4.83
ChiPFormer (2k)	6.62 ± 0.05	67.10 ± 5.46	76.70 ± 1.15	68.80 ± 1.59	2.95 ± 0.04	72.92 ± 2.56
SA (1M)	19.24 ± 1.42	102.84 ± 5.83	140.34 ± 7.48	148.57 ± 23.08	5.92 ± 0.98	270.46 ± 43.76
WireMask-EA (1k)	5.97 ± 0.07	54.31 ± 3.25	60.33 ± 0.40	61.91 ± 1.60	2.16 ± 0.01	62.68 ± 4.97
EfficientPlace (1k)	5.90 ± 0.07	42.52 ± 0.76	55.75 ± 0.84	54.68 ± 0.82	2.18 ± 0.02	57.81 ± 1.62
RSPlace (1k)	$\mathbf{4.72 \pm 0.07}$	$\mathbf{39.79 \pm 1.16}$	$\mathbf{54.50 \pm 0.85}$	$\mathbf{51.87 \pm 1.18}$	$\mathbf{2.03 \pm 0.01}$	$\mathbf{56.44 \pm 3.53}$

Table 1: Comparisons of HPWL ($\times 10^5$) on six different circuit benchmarks. The mean and standard deviation (mean±std) of each result are obtained by running under five different random seeds. The run steps after method names are also provided. A smaller HPWL value indicates better performance, with the best results highlighted in bold and the second-best underlined.

Benchmark	MaskPlace (3k)	ChiPFormer (2k)	WireMask-EA (1k)	EfficientPlace (1k)	RSPlace (1k)
bigblue2	5.75 ± 0.11	5.44 ± 0.10	4.48 ± 0.07	4.19 ± 0.08	$\mathbf{3.99 \pm 0.05}$
bigblue4	103.26 ± 2.69	102.84 ± 0.15	86.88 ± 0.99	88.01 ± 1.45	$\mathbf{82.29 \pm 1.29}$

Table 2: Comparisons of HPWL ($\times 10^5$) on "bigblue2" and "bigblue4" circuits. The best results are highlighted in bold and the second-best underlined.

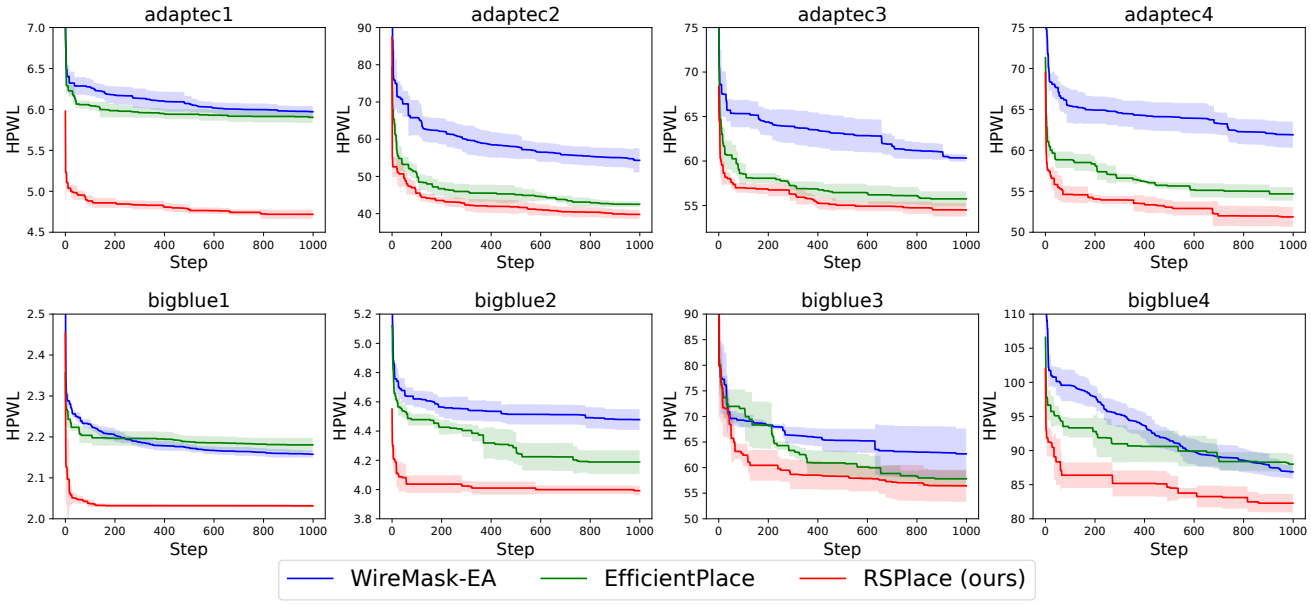


Figure 4: Results of HPWL ($\times 10^5$) vs. run steps. The shaded region represents the standard error.

Main Results. Table 1 presents the HPWL results of different methods on macro placement. To be consistent with the previous work (Geng et al. 2024), we conduct the experiment on 5 different random seeds and record the mean and standard deviation. The results of EfficientPlace and WireMask-EA are derived from by running their released code. The results of other baseline methods are from (Lai et al. 2023) and (Geng et al. 2024). We note that RSPlace attains the lowest HPWL in all circuit benchmarks. For the largest benchmarks, *bigblue2* and *bigblue4*, we place 256 and 1024 macros respectively, aligning with EfficientPlace. Results are provided in the Table 2. RSPlace demonstrates a performance improvement of 20% compared to EfficientPlace on the circuit benchmark *adaptec1*. For circuit benchmark *bigblue4*, RSPlace also has an improvement of 5% compared to WireMask-EA. Additionally, we further compare RSPlace with WireMask-EA, and the recent state-of-the-art approach EfficientPlace. The comparison illustrated in Figure 4 is based on HPWL over the number of run steps. For each circuit benchmark, the number of evaluations for both EfficientPlace and WireMask-EA is set to 1,000. Following (Geng et al. 2024), the best HPWL value reached by

each step is recorded for comparison. We find that RSPlace consistently achieves the lowest HPWL for each circuit benchmark, which further highlights the remarkable performance of our method.

Performance on Congestion Metric. The congestion of RSPlace is compared with MaskPlace, WireMask-EA and EfficientPlace by selecting the top 10% congested grids. The detailed results are presented in Appendix A.6. The results indicate that RSPlace can achieve low congestion levels while optimizing HPWL, consistent with WireMask-EA's finding that congestion strongly correlates with HPWL.

Mixed-Size Placement Results. The workflow of mixed-size placement of RSPlace includes three phases: (1) Return the macro placement result from RSPlace. (2) Fix macros placed by RSPlace, and utilize DREAMPlace to obtain a coarse placement result. (3) Allow all macros and standard cells movable, and use DREAMPlace for legalization and detailed placement. We report the mixed-size placement results in Appendix A.7, where RSPlace achieves the best performance in 7 of the 8 circuit benchmarks, showing that macro orientation also significantly impacts the mixed-size placement task.

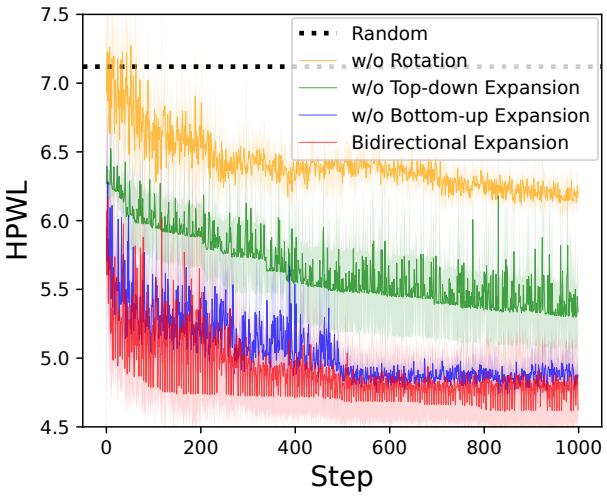


Figure 5: Compare HPWL curves for different components.

Ablation Study. The ablation study is conducted on circuit benchmark *adaptec1* to investigate the impact of different components in RSPlace. Results are shown in Figure 5 and the experimental configuration is as follows: (1) *Random:* This configuration indicates the orientation of each macro is determined by random rotation, indicating that introducing rotation without any guidance can seriously compromise performance; (2) *w/o Rotation:* The rotation is dropped in this configuration, where only a single orientation macro placement solution is considered in the tree expansion, greatly limiting the exploration space; (3) *w/o Top-down Expansion:* In this configuration, the expansion of the search tree only guided by random perturbation, leading the an insufficient exploration of the solution space; (4) *w/o Bottom-up Expansion:* We discontinue the bottom-up expansion of the search tree, resulting the search tree is trapped in the suboptimal placement for a long time; (5) *Bidirectional Expansion:* This is the standard configuration of RSPlace, achieving the best performance.

Analysis of Trajectory Exploration. To further investigate how macro orientation affects solution space exploration, we employ t-SNE to visualize the placement trajectories from RSPlace and EfficientPlace on the *adaptec1* circuit benchmark. Figure 6 presents 2,500 trajectory samples per method, with each independent run contributing the top 500 trajectories. We observe that the red and blue regions exhibit clear spatial separation, indicating different exploration patterns in high-quality solution spaces. Notably, RSPlace's top-5 trajectories occupy regions completely absent in EfficientPlace's exploration, demonstrating that ignoring macro rotation may render certain optimal solution spaces inaccessible. Another interesting finding is that, unlike the relatively concentrated distribution of EfficientPlace's top-5 trajectories, RSPlace's exhibit significant spatial dispersion. This not only validates the performance advantages of macro rotation but also reveals that RSPlace produces more diverse placement styles.

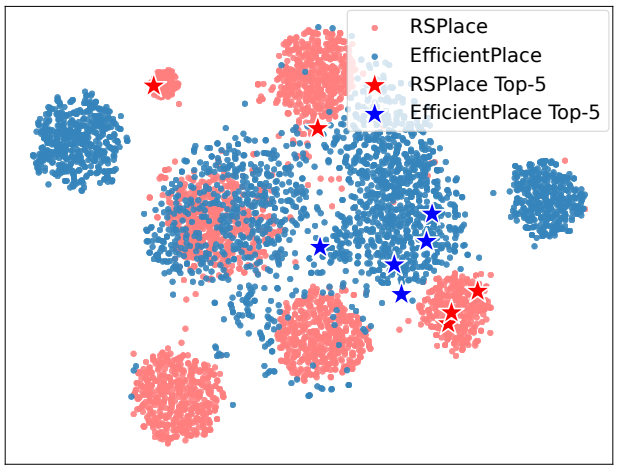


Figure 6: t-SNE visualization of the solution space explored by RSPlace and EfficientPlace, with stars marking the top-5 placements for each method.

Conclusion

This paper proposes a novel macro placer named RSPlace for offering RL agent more exploration opportunities. To incorporate the macro orientation into the RL-based macro placement solution, we design the rotation sensing module in the top-down tree expansion process, and introduce the placement perturbations into the bottom-up tree expansion phases. Though RSPlace shows significant potential, it also has some limitations. First, this work primarily focuses on HPWL and congestion as proxy metrics, due to its direct relationship with wirelength and routing quality. Second, analytical placers are required during mixed-size placement tasks. In the future, we hope this work can provide new insights for orientation optimization.

Acknowledgments

The authors gratefully acknowledge support from the NUAA Fundamental Research Funds for the Central Universities (No. NT2025006), the National Natural Science Foundation of China (No. U25A20533), the High Performance Computing Platform of Nanjing University of Aeronautics and Astronautics, and the Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

Browne, C. B.; Powley, E.; Whitehouse, D.; Lucas, S. M.; Cowling, P. I.; Rohlfshagen, P.; Tavener, S.; Perez, D.; Samothrakis, S.; and Colton, S. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1): 1–43.

Chen, T.-C.; Jiang, Z.-W.; Hsu, T.-C.; Chen, H.-C.; and Chang, Y.-W. 2006. A high-quality mixed-size analytical placer considering preplaced blocks and density constraints. In *Proceedings of the 2006 IEEE/ACM International Conference on Computer-Aided Design*, 187–192.

Chen, T.-C.; Jiang, Z.-W.; Hsu, T.-C.; Chen, H.-C.; and Chang, Y.-W. 2008. Ntuplace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(7): 1228–1240.

Chen, T.-C.; Yuh, P.-H.; Chang, Y.-W.; Huang, F.-J.; and Liu, D. 2007. MP-trees: A packing-based macro placement algorithm for mixed-size designs. In *Proceedings of the 44th annual Design Automation Conference*, 447–452.

Cheng, C.-K.; Kahng, A. B.; Kang, I.; and Wang, L. 2018. Replace: Advancing solution quality and routability validation in global placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 38(9): 1717–1730.

Cheng, C.-K.; Kahng, A. B.; Kundu, S.; Wang, Y.; and Wang, Z. 2023. Assessment of reinforcement learning for macro placement. In *Proceedings of the 2023 International Symposium on Physical Design*, 158–166.

Cheng, R.; and Yan, J. 2021. On joint learning for solving placement and routing in chip design. *Advances in Neural Information Processing Systems*, 34: 16508–16519.

Chiou, C.-H.; Chang, C.-H.; Chen, S.-T.; and Chang, Y.-W. 2016. Circular-contour-based obstacle-aware macro placement. In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, 172–177. IEEE.

Geng, Z.; Wang, J.; Liu, Z.; Xu, S.; Tang, Z.; Yuan, M.; Hao, J.; Zhang, Y.; and Wu, F. 2024. Reinforcement Learning within Tree Search for Fast Macro Placement. In *Forty-first International Conference on Machine Learning*, volume 235, 15402–15417.

Karypis, G.; Aggarwal, R.; Kumar, V.; and Shekhar, S. 1997. Multilevel hypergraph partitioning: Application in VLSI domain. In *Proceedings of the 34th Annual Design Automation Conference*, 526–529.

Khatkhate, A.; Li, C.; Agnihotri, A. R.; Yildiz, M. C.; Ono, S.; Koh, C.-K.; and Madden, P. H. 2004. Recursive bisection based mixed block placement. In *Proceedings of the 2004 International Symposium on Physical Design*, 84–89.

Kim, M.-C.; and Markov, I. L. 2012. Complx: A competitive primal-dual lagrange optimization for global placement. In *Proceedings of the 49th Annual Design Automation Conference*, 747–752.

Lai, Y.; Liu, J.; Tang, Z.; Wang, B.; Hao, J.; and Luo, P. 2023. ChiPFormer: Transferable chip placement via offline decision transformer. In *Proceedings of the 40th International Conference on Machine Learning*, 18346–18364.

Lai, Y.; Mu, Y.; and Luo, P. 2022. Maskplace: Fast chip placement via reinforced visual representation learning. *Advances in Neural Information Processing Systems*, 35: 24019–24030.

Lin, Y.; Jiang, Z.; Gu, J.; Li, W.; Dhar, S.; Ren, H.; Khailany, B.; and Pan, D. Z. 2020. DREAMPlace: Deep learning toolkit-enabled gpu acceleration for modern VLSI placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(4): 748–761.

Liu, Y.-C.; Chen, T.-C.; Chang, Y.-W.; and Kuo, S.-Y. 2019. MDP-trees: multi-domain macro placement for ultra large-scale mixed-size designs. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, 557–562.

Markov, I. L.; Hu, J.; and Kim, M.-C. 2012. Progress and challenges in VLSI placement research. In *Proceedings of the 25th International Conference on Computer-Aided Design*, 275–282.

Mirhoseini, A.; Goldie, A.; Yazgan, M.; Jiang, J. W.; Songhori, E.; Wang, S.; Lee, Y.-J.; Johnson, E.; Pathak, O.; Nazi, A.; et al. 2021. A graph placement methodology for fast chip design. *Nature*, 594(7862): 207–212.

Nam, G.-J.; Alpert, C. J.; Villarrubia, P.; Winter, B.; and Yildiz, M. 2005. The ISPD2005 placement contest and benchmark suite. In *Proceedings of the 2005 International Symposium on Physical Design*, 216–220.

Naylor, W. C.; Donelly, R.; and Sha, L. 2001. Non-linear optimization system and method for wire length and delay optimization for an automatic electric circuit placer. U.S. Patent 6 301 693.

Rabaey, J. M.; Chandrakasan, A.; and Nikolic, B. 2002. *Digital integrated circuits*, volume 2. Prentice hall Englewood Cliffs.

Roy, J. A.; Adya, S. N.; Papa, D. A.; and Markov, I. L. 2006. Min-cut floorplacement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(7): 1313–1326.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Shi, Y.; Xue, K.; Lei, S.; and Qian, C. 2023. Macro placement by wire-mask-guided black-box optimization. *Advances in Neural Information Processing Systems*, 36: 6825–6843.

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

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Viswanathan, N.; Nam, G.-J.; Alpert, C. J.; Villarrubia, P.; Ren, H.; and Chu, C. 2007. RQL: Global placement via relaxed quadratic spreading and linearization. In *Proceedings of the 44th Annual Design Automation Conference*, 453–458.