AlphaPlacer: Analog Placement Enhanced by Monte Carlo Tree Search

Yapeng Li[1†], Peng Xu[2†], Mingzi Wang[2], Tinghuan Chen[1*]
[1]The Chinese University of Hong Kong, Shenzhen [2]The Chinese University of Hong Kong

ABSTRACT

Analog layout design remains heavily dependent on manual expertise, with placement being the most critical stage requiring extensive time and domain knowledge. Current automated placement techniques face challenges in capturing expert design practices and fall short of practical deployment. To address this limitation, we present AlphaPlacer, a novel MCTS-based analog placement framework that learns from historical layouts to provide expert knowledge-enhanced placement optimization. Our approach formulates analog placement as a hierarchical sequence pair search with MCTS, incorporating a bi-level search tree that decouples floorplan topology from device assignment and a similarity-driven reward mechanism. A pretrained learning framework is also integrated to capture sequence pair distributions from expert layouts. These components collectively enable AlphaPlacer to efficiently explore toward expert-like placements with improved convergence. Experimental results demonstrate that our method significantly outperforms baselines across key metrics including area, wirelength, and post-layout performance.

1 INTRODUCTION

Analog placement is a crucial step in generating analog layouts, which plays a major role in achieving high quality and performance in analog integrated circuits (ICs). It involves determining the physical locations of analog building blocks, each composed of one or more devices, before the routing procedure. A reliable evaluation of analog layout quality in the physical domain, which directly influences electrical circuit behavior, requires minimizing both the overall chip area and the estimated interconnect wire length.

As described in [1], decades of research on analog placement can be grouped into three main classes: (i) template-based methods, (ii) linear or nonlinear optimization, and (iii) simulated annealing (SA). Template-based approaches such as BAG [2] and LAYGO [3, 4] use user-defined layout patterns or floorplans, but their reliance on predefined templates and lack of an end-to-end flow limit flexibility. Mathematical formulations based on mixed-integer linear programming (MILP) [5–7] and nonlinear programming (NLP) [8–10] directly optimize metrics such as area and wirelength. MILP often suffers from poor scalability, while NLP leverages differentiable objectives and matrix-based solvers for faster convergence and better scalability. From an algorithmic perspective, simulated annealing remains the classical analog placement paradigm, with representative works such as [7, 11]. These methods use compact spatial representations, including slice trees [12], B* trees [13], and especially sequence pairs [14, 15], which are the most widely adopted.

[†]Equal contribution. [*]Corresponding author.

This work is partially supported by the National Key Research and Development Program of China (No. 2023YFB4402900) and the National Natural Science Foundation of China (No. 92573108, 62304197).

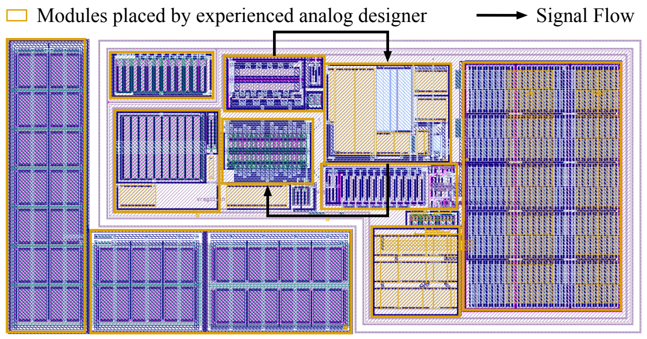


Figure 1: Illustration of an expert-crafted analog layout.

Due to the special requirements of analog placement on post-simulation performance, the placement optimization process must pay particular attention to satisfying various constraints. This optimization is performed while satisfying essential analog placement constraints, including symmetry constraint [16–18], proximity [19–21], and regularity [22, 23], etc. The common-centroid constraint [24, 25] is generally addressed during the layout generation of analog building blocks or primitive analog cells. As shown in Figure 1, designers cluster similar devices along the signal path and group devices with similar aspect ratios to reduce well usage and white space, implicitly satisfying multiple nested constraints. Because many decisions rely on experience and multi objective trade-off rather than explicit rules, this expertise is hard to encode with traditional methods. Hence, both tools and design flows would benefit from methods that implicitly capture design knowledge and automatically extract constraints from high-quality layouts.

Machine learning can uncover latent structures in analog placement data; however, existing models rarely seek to explicitly imitate the placement data from human designers. Liu et al. [26]developed a carefully designed convolutional neural network (CNN) model to capture the complex correlation between layout and final circuit performance. Li et al. [27]similarly proposed a customized graph neural network (GNN) model to estimate the impact of placement on post-layout performance. In addition, Chang et al. [28]leveraged Neural Architecture Search (NAS) to automatically construct tailored neural architectures for different analog circuit topologies and performance metrics. Recently, Lin et al. [29] conducted an initial study using a basic reinforcement learning (RL) framework for analog placement. However, few of them consider how to automatically learn effective placement patterns from historical layout data in a way that directly improves placement quality.

To address this gap, we propose a similarity-driven, learning-enhanced MCTS framework for analog placement. In our framework, analog placement is formulated as a bi-level sequence-pair search problem, and a customized reward function is devised to quantify the

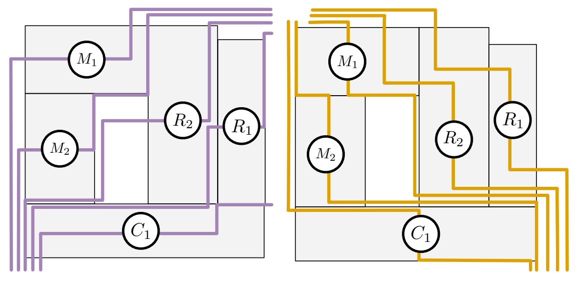


Figure 2: Placement and its corresponding sequence pair: (a) $\Gamma_+ = (M_1, M_2, R_2, R_1, C_1)$ and (b) $\Gamma_- = (C_1, M_2, M_1, R_2, R_1)$.

similarity between the current layout and a set of historical layouts. In this manner, the framework implicitly captures designer expertise and automatically infers constraints from high-quality reference layouts, thereby enabling post-layout optimal designs. The main contributions of this work are summarized as follows:

- We propose **AlphaPlacer**, an analog placement framework that formulates placement as a hierarchy search space solved by a customized MCTS method, separating high level floorplan topology from device-level assignment.
- We design a customized bi-level search tree and a similarity-driven reward in sequence pair space, enabling MCTS to exploit expert-like spatial patterns directly from historical layouts.
- We augment MCTS with a pretrained value network that learns from expert layouts, enabling the search tree to start with similarity-aware priors and converge toward high-quality placements under multiple objectives.
- On multiple benchmark circuits, our proposed AlphaPlacer framework consistently outperforms state-of-the-art analog placers in area, HPWL, and post-layout performance metrics.

2 PRELIMINARIES

2.1 Problem Formulation

Problem 1 (Similarity-driven Analog Placement). Given a netlist of an analog circuit and a library of expert layout designs, determine a position for each device to minimize area, wire length, and symmetry offset while preserving expert-like spatial patterns and style consistency with the expert layouts.

2.2 Sequence Pair

The *Sequence Pair (SP)* scheme offers a concise way to encode non-overlapping placements [30]. The SP representation defines a compact yet expressive search space, where each pair of permutations corresponds to a valid geometric topology. Concretely, it represents spatial relationships among blocks using two permutations of all device indices denoted by two sequences: positive sequence Γ_+ that encodes up-right and down-left step lines representing ascending spatial relationships between sequence orderings, and negative sequence Γ_- that encodes left-up and right-down step lines representing descending spatial relationships between sequence orderings. An example of a placement and its sequence pair is illustrated in Figure 2. The relative position of any device pair (I_a, I_b) is encoded by the order in these two sequences:

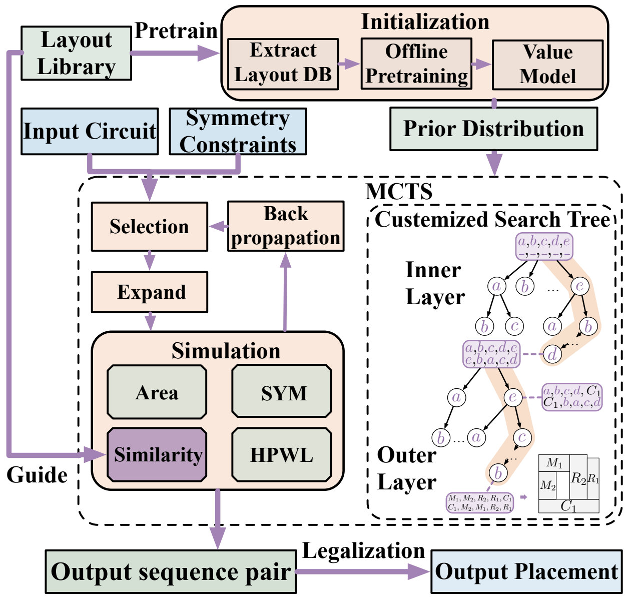


Figure 3: Our overall flow.

- **Horizontal (H) constraint:** if I_a appears before I_b in both Γ_+ and Γ_-, then I_a is to the *left* of I_b.
- **Vertical (V) constraint:** if I_a appears before I_b in Γ_+ but after I_b in Γ_-, then I_b is *below* I_a.

Based on these pairwise ordering relations, device coordinates can be computed through various algorithms such as longest path evaluation or linear programming (LP) based compaction methods, ensuring a non-overlapping placement [11].

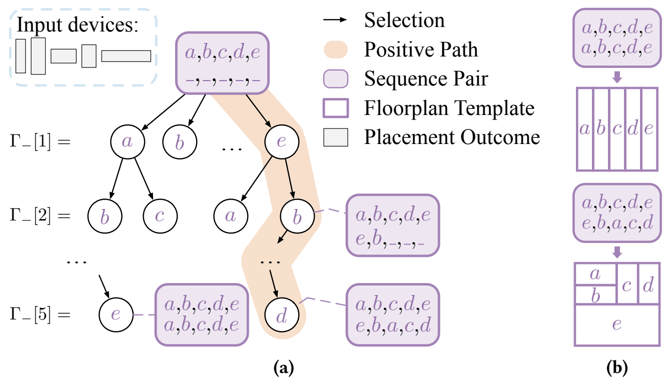
2.3 Monte Carlo Tree Search

The Monte Carlo Tree Search (MCTS) algorithm [31] uses random sampling and statistics from past actions to guide future decisions, and has shown strong performance in black-box and combinatorial optimization problems, such as Go [32, 33]. In MCTS, a search tree is built where each node represents a problem state and is assigned an Upper Confidence Bound for Trees (UCT) value that measures its suitability for selection.

3 METHOD

3.1 Overview

Given a library of expert layout designs, an input circuit and its symmetry constraints, AlphaPlacer formulates analog placement as a sequential decision process on sequence pair representations and solves it with MCTS guided by expert layout patterns. As shown in Figure 3, the framework operates through the following key components - Initialization Phase: We extract a layout database from the expert library and perform offline pretraining of a value model that captures expert design knowledge. This pretrained model is then integrated into MCTS to provide similarity-aware prior distributions, replacing uninformed random initialization. MCTS Search Process: The search iterates through four standard phases - selection, expansion, simulation, and backpropagation - using a customized bi-level search tree. The outer level constructs global floorplan topology while the inner level handles concrete device assignment. These



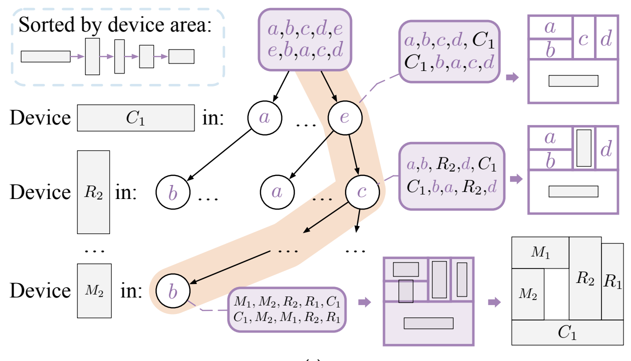


Figure 4: Search tree: (a) outer level search tree construction, (b) example floorplan templates from outer stage leaf nodes, and (c) inner level search tree construction.

dual mechanisms bias the MCTS search toward expert-like design patterns, enabling more promising exploration and convergence. Upon convergence, the framework outputs an optimal sequence pair, which after legalization yields the final placement.

To solve Problem 1, we define our optimization objective to minimize an augmented cost function for a sequence pair $s = (\Gamma+, \Gamma-)$:

$$cost(s) = \text{area}(s) + \alpha\ \text{HPWL}(s) + \beta\ \text{SYM}(s) - \gamma\ \text{SIM}(s), \quad (1)$$

where area$(\cdot)$, HPWL$(\cdot)$, and SYM$(\cdot)$ are evaluated on the placement derived from the sequence pair, representing the total packing area, half perimeter wire length, and symmetry offset, respectively. SIM$(\cdot)$ denotes our customized similarity-driven reward term that measures how closely the candidate placement matches reference expert layouts or desired topological patterns. Higher SIM$(\cdot)$ values lead to lower cost through the negative sign, encouraging expert like placements. The weighting coefficients α, β, and γ control the relative importance of different objectives. The reward function is defined as the negative of the cost function: The reward function for MCTS is defined as the negative of the cost function:

$$reward(s) = -cost(s). \quad (2)$$

3.2 Customized Bi-level Search Tree

As illustrated in Figure 4, We designed a customized bi-level search tree for sequence pair based analog placement, decomposing the search into two hierarchical stages. The *outer level* explores high level floorplan topology by constructing a sequence pair template over position identifiers, while the *inner level* instantiates this template by assigning concrete devices to these positions. This coarse to fine strategy yields a compact search space, cleanly separating global topology from device level decisions and remaining compatible with longest path evaluation.

Outer Level for Topological Structure. The outer level determines the high level spatial relationships among placement positions by incrementally constructing the sequence pair. Given n devices, we fix the positive sequence as $\Gamma_+ = (a, b, c, \ldots)$ of length n, where each element denotes a position identifier. The negative sequence Γ_- is left empty at the root and is built step by step during tree traversal. The outer search tree has n layers. At layer l, the algorithm chooses which remaining position identifier occupies entry $\Gamma_-[l]$, and each child node corresponds to a specific choice.

For example, in Figure 2(a), with $n = 5$ input devices, we fix $\Gamma_+ = (a, b, c, d, e)$, and the root corresponds to the incomplete pair $(a, b, c, d, e)/(\_, \_, \_, \_, \_)$. At the first layer, if position e is selected for $\Gamma_-[1]$, we obtain a child node representing $(a, b, c, d, e)/(e, \_, \_, \_, \_)$. This process continues until reached depth n, where all entries of Γ_- are filled with position identifier, and a complete topological configuration such as $(a, b, c, d, e)/(e, c, a, d, b)$ is formed. Each leaf node at the outer level, therefore, encodes a distinct topological floorplan template: it fixes the relative ordering of positions in both sequences, and thus the spatial relationships among positions, while remaining agnostic to which specific devices will eventually occupy those positions. Two representative templates are shown in Figure 4(b).

Inner Level for Device Assignment. The inner level instantiates a chosen topological template by mapping concrete devices to the position identifiers defined at the outer level. Starting from a leaf node of the outer level, which fixes (Γ_+, Γ_-) over the set of position identifiers, the inner search tree assigns devices to these positions level by level. Following common layout practice, where larger devices are usually placed first to reduce fragmentation and improve routability, we sort all input devices in descending order of area.

For the example in Figure 4(c) with devices R_1, R_2, M_1, M_2, C_1 and outer level leaf $(a, b, c, d, e)/(e, c, a, d, b)$, sorting yields the order C_1, R_2, R_1, M_2, M_1, which determines the assignment order across the n layers of the inner tree. At layer l, the algorithm selects an unused position for the l-th device in this list; each child at depth l fixes that device's placement. For example, if the first two layers assign C_1 to e and R_2 to c, we obtain the partial sequence pair $s = (a, b, R_2, d, C_1)/(C_1, R_2, a, d, b)$. After all n layers, each inner leaf corresponds to a fully specified sequence pair, to which we apply the longest path algorithm to compute device coordinates, yielding the final placement in the bottom-right of Figure 4(c).

3.3 Similarity-Driven Reward

Analog layout experts typically reason in terms of *relative* placement patterns rather than absolute coordinates, e.g., "device I_a is above I_b and to the left of I_c". Our bi-level search naturally exposes such relations in the sequence pair domain: the outer level fixes the relative order of *positions* in Γ_+ and Γ_-, and the inner level binds devices to these positions. To guide MCTS toward layouts that resemble expert-designed examples, we define a sequence pair similarity score that operates directly on the sequence pair representation.

Given a sequence pair $\boldsymbol{s} = (\Gamma_+, \Gamma_-)$ and its corresponding device set $\mathcal{D}$, for any pair of devices (I_a, I_b) from $\mathcal{D}^i$, we define a relation function $R^{(s)}(I_a, I_b) \in \{above, below, left, right\}$ that determines spatial relation of I_a and I_b based on relative orders in $\boldsymbol{s}_i$. We then consider two sequence pairs: a sequence pair $\boldsymbol{s}_i = (\Gamma^i_+, \Gamma^i_-)$ produced by the search of input circuit C_i and a *reference* sequence pair $\boldsymbol{s}_j = (\Gamma^j_+, \Gamma^j_-)$ from a expert designed layout of reference circuit C_j ; and we assume they have the same device set: $\mathcal{D}^i = \mathcal{D}^j$.

Our sequence pair similarity score is then a Kendall-tau style [34] ranking score computed purely in sequence pair space:

$$r(\boldsymbol{s}_i, \boldsymbol{s}_j) = \frac{\left|\{(I_a, I_b) : R^{(s_i)}(I_a, I_b) = R^{(s_j)}(I_a, I_b)\}\right|}{|\{(I_a, I_b) : (I_a, I_b) \in \mathcal{D}^i\}|} \in [0, 1], \quad (3)$$

where the numerator counts device pairs with consistent spatial relations between the two sequence pairs, and the denominator represents the total number of device pairs. The similarity $r(\cdot)$ reaches 1 when all comparable pairwise relations induced by the two sequence pairs agree. The similarity directly measures how well the resulting device level sequence pair matches the designer's pairwise ordering patterns. In this way, the sequence pair similarity encourages the search to preserve expert-like pairwise relations. As illustrated in Figure 5, the sequence pair similarity effectively captures spatial placement characteristics.

Extension to Partial Similarity Assessment. In practice, the input circuit rarely matches any library layout circuit exactly (with different device set: $\mathcal{D}^i \neq \mathcal{D}^j$), so direct sequence pair similarity is not sufficient. We therefore use Schematic Versus Schematic (SVS) matching to extract partial correspondences between the input schematic and all library schematics[35, 36]. Given input circuit C_i with device set $\mathcal{D}_i$, for each library circuit C_j, SVS evaluates their schematics and returns a subset of matched devices $\mathcal{D}_{ij} \subseteq \mathcal{D}_i$. The coverage of the match of C_j is defined as

$$\text{coverage}_{ij} = \frac{|\mathcal{D}_{ij}|}{|\mathcal{D}_i|}. \quad (4)$$

From the complete sequence pair $\boldsymbol{s}_i = (\Gamma^i_+, \Gamma^i_-)$ of the input circuit, we derive a sub-sequence pair $\boldsymbol{s}_{ij} = (\Gamma^{ij}_+, \Gamma^{ij}_-)$ by retaining only devices in $\mathcal{D}_{ij}$ and preserving their relative order. Using Equation (3), we then compute a similarity score $r \in [0, 1]$ between $\boldsymbol{s}_{ij}$ and the corresponding subsequence pair $\boldsymbol{s}_{ji}$ from the layout of circuit C_j in the library. To balance quality and diversity of references, we select the top-M matches with the largest coverage_{ij}. The similarity-driven reward $\text{SIM}(\cdot)$ aggregates these individual scores using a coverage-weighted average:

$$\text{SIM}(\boldsymbol{s}_i) = \frac{\sum_j^M \text{coverage}_{ij} \cdot r(\boldsymbol{s}_{ij}, \boldsymbol{s}_{ji})}{\sum_j^M \text{coverage}_{ij}}. \quad (5)$$

This formulation ensures that templates with higher device overlap have a larger influence on the reward, while still allowing the search to benefit from multiple partial references when no single layout provides a complete match.

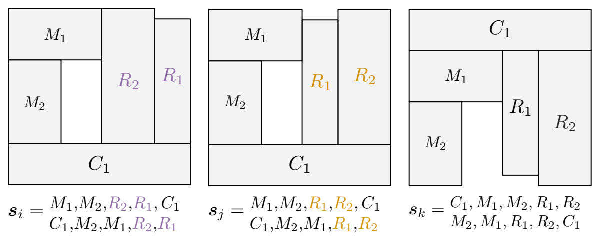

M_1
M_2
R_2
R_1
C_1
M_1
M_2
R_1
R_2
C_1
C_1
M_1
R_1
R_2
M_2
$\boldsymbol{s}_i = M_1, M_2, R_2, R_1, C_1$
C_1, M_2, M_1, R_2, R_1
$\boldsymbol{s}_j = M_1, M_2, R_1, R_2, C_1$
C_1, M_2, M_1, R_1, R_2
$\boldsymbol{s}_k = C_1, M_1, M_2, R_1, R_2$
M_2, M_1, R_1, R_2, C_1

Figure 5: Different placements with sequence pair similarity score $r(\boldsymbol{s}_i, \boldsymbol{s}_j) = 0.9$, and $r(\boldsymbol{s}_i, \boldsymbol{s}_k) = 0.3$.

3.4 Domain-specific MCTS

In our framework, MCTS is equipped with learning-oriented strategies to progressively identify promising sequence pair patterns and emulate expert analog layout styles. We apply these strategies across all four MCTS phases, customized for a bi-level sequence pair search space. For an analog circuit with n modules, a placement is encoded as a sequence pair (Γ_+, Γ_-) of length $2n$, where Γ_+ and Γ_- are permutations of the same device set. The outer level of the search fixes a topological template by constructing one sequence, while the inner level assigns specific devices to sequence positions, yielding a total of $(n!)^2$ possible placements. Rather than exploring this space uniformly, MCTS constructs a search tree whose nodes represent partial sequence pairs and whose edges represent incremental choices of the next sequence element.

Initialization of MCTS. To exploit historical layouts and stabilize early stage search, AlphaPlacer performs an offline pretraining of the MCTS value model using our sequence pair similarity score Equation (3). We first extract a layout database from the library of expert layout designs and encode each expert placement as a sequence pair. Running AlphaPlacer in an offline mode over these references produces trajectories $(\boldsymbol{s}_t, r_t)$, where $\boldsymbol{s}_t$ is a (partial or complete) sequence pair state at current node t and $r_t \in [0, 1]$ is the similarity score to the corresponding expert layout obtained by the similarity evaluator. These trajectories supervise a lightweight value network V_θ attached to MCTS. For each visited sequence pair state $\boldsymbol{s}_t$ with score r_t, we minimize the regression loss:

$$\mathcal{L}_{\text{pre}} = \left(r_t - V_\theta(\boldsymbol{s}_t)\right)^2, \quad (6)$$

so that V_θ learns to predict how expert-like each partial placement is. After pretraining, its output serves as a data driven prior for the initial search tree, so online MCTS starts from similarity-aware values rather than uninformed statistics.

Selection & Expansion of MCTS. During online search, AlphaPlacer adopts a bi-level selection strategy that is tightly coupled with the similarity-driven value estimates and the bi-level search space. At the outer level, each node corresponds to a coarse floorplan topology represented by a partial sequence pair. From a parent node $\boldsymbol{s}$, MCTS selects the next child $\boldsymbol{s}^*$ using a UCT score as folows:

$$\boldsymbol{s}^* = \underset{\boldsymbol{s}'_t \in \text{Child}(\boldsymbol{s}_t)}{\text{argmax}} \left\{ V_\theta(\boldsymbol{s}_t) + c_{\text{ucb}} \sqrt{\frac{\ln N(\boldsymbol{s}_t)}{1 + N(\boldsymbol{s}_t, \boldsymbol{s}'_t)}} \right\}, \quad (7)$$

where $V_\theta(\boldsymbol{s})$ is the pretrained value model hat predicts similarity to expert-like placements, which is the empirical mean return of the chile node $\boldsymbol{s}'$ within the child node set of state $\boldsymbol{s}$, $N(\boldsymbol{s})$ and $N(\boldsymbol{s}, \boldsymbol{s}')$ are the visit counts of the node and edge, and c_{ucb} balances

Table 1: Placement Performance Comparison.

Circuit	NLP-based (MAGICAL) [10, 37]			SA-based (Align) [7]			RL-based [29]			**AlphaPlacer (ours)**		
	Area (μm^2)↓	HPWL (μm)↓	Runtime (s)↓	Area (μm^2)↓	HPWL (μm)↓	Runtime (s)↓	Area (μm^2)↓	HPWL (μm)↓	Runtime (s)↓	Area (μm^2)↓	HPWL (μm)↓	Runtime (s)↓
OTA1-A	2720.64	609.32	41.73	1453.84	406.57	20.78	1387.24	398.48	18.92	**1175.03**	**270.42**	**14.19**
OTA1-B	2934.98	612.02	41.85	1406.50	602.28	**8.19**	1419.07	589.31	19.34	**1185.69**	**278.27**	69.30
OTA1-C	3161.78	552.86	40.62	1637.37	541.94	58.93	1597.87	531.75	**32.41**	**1394.45**	**280.88**	55.01
OTA1-D	3113.88	613.36	36.28	1824.52	644.13	17.45	1743.93	338.97	15.67	**1330.35**	**378.41**	**16.84**
OTA2-A	3084.60	558.52	**32.91**	1307.10	460.28	35.46	1298.30	**245.88**	**31.28**	**1175.92**	278.04	80.44
OTA2-B	3559.04	686.48	37.64	1339.39	647.75	**8.92**	1352.66	659.4	10.15	**1235.97**	**298.72**	40.80
OTA2-C	3225.38	582.37	45.81	1412.15	580.94	26.67	1379.40	571.30	**23.84**	**1221.22**	**321.60**	67.46
OTA2-D	3225.38	623.31	50.49	1393.77	499.32	**9.43**	1418.90	492.60	58.76	**1288.23**	**343.44**	114.71
OTA3-A	3377.58	728.78	**18.35**	660.65	643.07	59.96	681.30	625.40	54.73	**440.03**	**482.77**	58.34
OTA3-B	3653.24	763.78	**12.17**	640.91	702.98	94.62	628.70	694.50	87.29	**477.52**	**484.65**	68.36
OTA4-A	3595.21	740.58	**10.89**	718.97	631.98	99.85	697.40	618.70	41.47	**577.01**	**445.60**	89.70
OTA4-B	3595.21	756.98	**11.56**	750.04	637.01	118.69	762.80	621.90	109.30	**507.99**	**456.60**	112.35
Average Ratio	1.00	1.00	**0.62**	0.38	0.89	0.70	0.38	0.81	**0.62**	**0.31**	**0.54**	1.00

exploration and exploitation. At the inner level, once a promising floorplan node is reached, we perform device assignment within that topology. Candidate assignments are selected by the UCT score. Thus, similarity guides both high level floorplan exploration and low level device placement, driving the search toward compact, low HPWL, expert-like layouts.

Simulation of MCTS. The simulation evaluates the newly expanded leaf using a batch-based approach. Given a partial sequence pair at node $\boldsymbol{s}_T$, AlphaPlacer samples multiple complete sequence pairs by randomly assigning the remaining positions and devices. Each candidate is scored with the multi-objective reward in Equation (2), combining area, wirelength, symmetry offset and similarity reward, and the simulation returns the best reward in the batch. A global archive caches evaluated sequence pairs and their rewards, allowing reuse of past results and turning expensive full placement evaluations into a reusable repository that accelerates future tree exploration.

MCTS is considered converged when the simulation phase fails to discover any improved solution within a fixed number of consecutive iterations. Upon convergence, we extract the final placement by selecting the child of the root with the highest average reward and recursively following the locally best child at each level, obtaining a complete sequence pair $(\Gamma_+^\star, \Gamma_-^\star)$, which is then legalized to be a placement as the final output. If convergence is not reached, the search continues with backpropagation.

Backpropagation of MCTS. Once the reward value $reward(\boldsymbol{s}_T)$ is obtained via Equation (2), it is propagated backward from $\boldsymbol{s}_T$ to the root $\boldsymbol{s}_0$. For each node $\boldsymbol{s}_t$ on this path, update its visit count and average reward:

$$N_t \leftarrow N_t + 1, \quad \bar{x}_t \leftarrow \frac{\bar{x}_t(N_t - 1) + reward(\boldsymbol{s}_T)}{N_t}. \tag{8}$$

These updates make each node's value estimate converge to the expected reward of completing its partial sequence pair, so that promising prefixes accumulate higher $\bar{x}_t$ and are selected more often.

Legalization. During MCTS optimization, symmetry is a soft constraint via a differentiable loss that penalizes asymmetric orderings between mirrored module pairs in the precedence matrices. For left–right symmetric pairs (i, i') and (j, j'):

$$\mathrm{SYM}(\boldsymbol{s}) = \sum_{(i,i'),(j,j')\in\mathcal{M}_{\mathrm{lr}}} \left\| R^{(s)}(I_i, I_j) - R^{(s)}[I_{i'}, I_{j'}] \right\|_2^2. \tag{9}$$

This term is added as a penalty in the reward, biasing the policy toward symmetry-aware placements. After optimization, a MILP-based legalization enforces hard geometric symmetry and legality. For each left–right pair (m_{l_i}, m_{r_i}) and top–bottom pair (m_{t_i}, m_{b_i}):

$$x_{l_i} + w_{l_i} + x_{r_i} = W, \qquad y_{l_i} = y_{r_i}, \tag{10}$$

$$y_{t_i} + h_{t_i} + y_{b_i} = H, \qquad x_{t_i} = x_{b_i}, \tag{11}$$

mirroring modules about the layout center lines with alignment and non-overlap. The MILP acts as an environment level feasibility projection, mapping imperfect actions to legal, symmetry-satisfying layouts. Together, the soft symmetry loss and hard MILP constraints yield symmetry-aware learning with guaranteed symmetric final placements.

4 EXPERIMENTAL RESULTS

4.1 Experimental Setting

We implement the proposed performance driven analog layout framework in Python 3.7.5 and C++. All experiments, including both our method and the baselines, are conducted on the same machine equipped with an Intel Xeon Platinum 8368@2.40GHz CPU and 64 GB Micron 3200 MHz RAM. The differentiable global placer is implemented in PyTorch 2.0.0, the legalization engine is implemented in C++, and all mixed-integer linear programs are solved using Gurobi [38]. For routing, we build upon a modified version of the C++ MAGICAL framework [10].

Benchmarks. We evaluate methods on ten operational transconductance amplifier (OTA) designs, labeled OTA1(A-D), OTA2(A-D), OTA3(A-B), and OTA4(A-B). These benchmarks differ in circuit topology and function, and were created by experienced designers in TSMC 40 nm technology. OTA1s and OTA2s are distinct two-stage Miller compensated OTA, while OTA3s and OTA4s are two-stage telescopic OTA with more complex topologies. All designs are implemented in TSMC 40 nm, parasitics are extracted with Calibre PEX, and post-layout simulations are performed in Cadence Spectre.

Compared Methods. We compare our AlphaPlacer framework against three state-of-the-art analog placement methods: (i) MAGICAL (NLP)[37], which uses analytical algorithms for fully automated layout but may underutilize area due to relaxed compactness; (ii) ALIGN (SA)[7], which employs a compact sequence-pair representation and simulated annealing but does not directly optimize specific

Table 2: Post-layout Performance Comparison.

Circuits	NLP-based (MAGICAL) [10, 37]					SA-based (ALIGN) [7]					RL-based [29]					**AlphaPlacer (Ours)**				
	OV (μV)↓	CMRR (dB)↑	BW (MHz)↑	Gain (dB)↑	Noise (μV_{rms})↓	OV (μV)↓	CMRR (dB)↑	BW (MHz)↑	Gain (dB)↑	Noise (μV_{rms})↓	OV (μV)↓	CMRR (dB)↑	BW (MHz)↑	Gain (dB)↑	Noise (μV_{rms})↓	OV (μV)↓	CMRR (dB)↑	BW (MHz)↑	Gain (dB)↑	Noise (μV_{rms})↓
OTA-1-A	1186	**104.5**	51.66	36.96	224.6	4289	34.89	50.97	33.18	234.7	1856	52.34	48.92	34.85	229.2	**332.6**	75.12	**54.82**	**37.57**	**222.4**
OTA-1-B	1277	67.73	53.77	36.86	225.5	49700	23.20	25.53	14.62	277.0	5420	41.25	31.87	19.34	251.8	**89.57**	**70.62**	**57.91**	**38.07**	**222.2**
OTA-1-C	1389	24.89	53.73	**39.81**	247.5	30270	25.32	13.91	18.32	243.0	8947	34.18	25.64	24.71	239.4	**163.2**	**64.26**	**59.52**	37.05	**227.6**
OTA-1-D	**964.4**	**79.20**	47.37	36.22	226.9	17900	37.98	43.33	27.65	230.5	4325	59.12	46.18	31.92	233.1	1583	65.50	**51.01**	**36.29**	**226.0**
OTA-2-A	6086	**59.31**	**38.75**	35.70	258.0	8165	32.04	37.01	30.31	299.9	5623	29.87	39.45	32.17	287.3	**4895**	41.03	37.83	**37.34**	**254.2**
OTA-2-B	55580	**27.22**	3.390	13.12	256.1	14130	**38.38**	29.62	23.49	**247.6**	**11270**	22.96	33.84	26.73	378.9	14910	15.42	**34.69**	**28.20**	424.7
OTA-2-C	35660	11.04	33.48	19.31	575.5	203400	22.46	6.928	8.243	219.4	47890	24.33	14.25	12.68	235.7	**7866**	**25.21**	**36.19**	**33.73**	**274.7**
OTA-2-D	**5731**	**36.48**	**37.69**	**36.08**	**259.2**	13810	12.48	11.04	29.55	557.7	11920	**47.83**	18.47	29.12	486.2	13100	18.24	34.95	29.44	352.8
OTA-3-A	8.800	113.3	263.2	6.670	2359	12.38	99.52	164.9	6.223	2400	9.845	112.7	198.3	6.441	2376	**8.720**	**127.1**	**274.5**	**6.690**	**2343**
OTA-3-B	**3.660**	90.03	270.3	6.670	2543	12.56	90.72	**271.7**	4.894	3897	10.94	94.18	235.6	7.892	3124	11.17	**96.42**	214.9	**44.62**	**2248**
OTA-4-A	9.340	111.1	262.8	6.670	2374	13.33	93.89	164.1	6.209	2398	11.47	108.2	203.4	6.382	2371	**8.802**	**113.3**	**274.5**	6.670	**2346**
OTA-4-B	**3.860**	**90.24**	270.0	6.670	2560	14.34	86.53	40.73	6.103	2397	19.73	88.41	89.34	7.215	2187	42.18	85.58	**297.7**	**10.86**	**1963**
Average Ratio	**0.43**	**1.00**	0.90	0.83	0.95	1.00	0.76	0.61	0.66	1.00	0.58	0.85	0.71	0.74	0.98	0.50	0.98	**1.00**	**1.00**	**0.91**

Table 3: Schematic Performance.

Circuits	OTA	1-A	1-B	1-C	1-D	2-A	2-B	2-C	2-D	3-A	3-B	4-A	4-B
CMRR	(dB)↑	153.3	150.2	143.3	115.3	155.3	155.0	154.7	154.4	126.6	136.1	88.31	91.04
BW	(MHz)↑	112.1	115.7	118.8	121.0	108.1	108.5	108.7	108.7	589.7	595.1	607.3	595.4
Gain	(dB)↑	38.26	38.50	38.61	38.43	37.92	38.54	39.05	39.48	47.48	47.60	22.20	22.49
Noise	(μV_{rms})↓	214.5	215.8	217.8	221.8	213.4	214.5	215.5	216.4	3024	3040	2428	2484

design objectives; (iii) RL-based method [29], which uses GraphSAGE and BSG with hierarchical clustering and reward-based learning to handle proximity and symmetry constraints, but does not incorporate customized similarity-aware objectives for analog-specific layout requirements.

Evaluation Metrics. We evaluate the placement quality using multiple metrics. For layout efficiency, we measure runtime (RT), placement area (in μm^2), and half perimeter wirelength (HPWL in μm). For circuit performance evaluation, we conduct post layout simulations to extract key analog metrics including offset voltage (OV in μV), common mode rejection ratio (CMRR in dB), bandwidth (BW in MHz), DC gain (in dB), and input-referred noise (in μV_{rms}) for OTA circuits.

4.2 Overall Comparison

Table 1 compares AlphaPlacer with MAGICAL, ALIGN, and the RL-based approach in terms of placement quality. Across all OTA benchmarks, AlphaPlacer consistently achieves the smallest area and HPWL while maintaining practical runtime. On average, AlphaPlacer demonstrates substantial improvements in compactness and wirelength efficiency compared to all baseline methods, with the most significant gains observed against MAGICAL and competitive advantages over ALIGN and the RL-based approach. While AlphaPlacer requires slightly longer runtime than some baselines in some cases, this overhead is fully justified by the substantial performance improvements achieved. These results indicate that the similarity-guided bi-level MCTS search effectively explores more compact and wirelength efficient placements than existing approaches.

Table 2 and Table 3 report post-layout circuit performance and the corresponding pre-layout schematic performance. AlphaPlacer consistently outperforms all baseline methods in offset voltage reduction, achieving the best robustness across different circuit variants. For small-signal metrics, AlphaPlacer maintains excellent bandwidth and gain preservation while effectively controlling noise levels. Notably, AlphaPlacer shows superior performance stability compared to ALIGN and the RL-based method, which often suffer from significant degradation in gain and bandwidth. The runtime performance remains competitive across all comparisons.

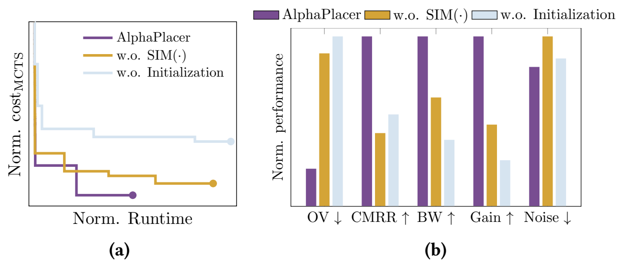


Figure 6: Component ablation study on OTA-2: (a) optimization convergence and (b) post-layout circuit performance.

Overall, AlphaPlacer generates layouts that are more compact and have shorter interconnects, and these physical improvements translate into superior post-layout electrical performance. The gains in area, HPWL, and key analog figures of merit, relative to all baseline approaches, validate the effectiveness of our similarity-guided bi-level MCTS framework in capturing expert placement patterns and optimizing practical analog designs.

4.3 Ablation Study

We evaluate our similarity-driven reward and initialization via ablation on OTA-2(A-D), comparing: (i) full AlphaPlacer, (ii) without similarity reward SIM(·) (Equation (5)), and (iii) without initialization (Section 3.4). Results are averaged and normalized across all test circuits. Figure 6(a) shows that full AlphaPlacer converges fastest to the lowest cost. Removing SIM(·) gives a comparable final cost but slower early convergence, while removing initialization severely hurts convergence and raises the cost. Figure 6(b) further shows that only the full AlphaPlacer attains the best post-layout metrics; omitting either component degrades circuit performance. Overall, SIM(·) steers MCTS toward expert-like patterns, and initialization supplies stable, informed value estimates in early search, making both crucial for high-quality analog layouts.

5 CONCLUSION

We presented AlphaPlacer, an analog placement framework that models placement as a hierarchical sequence pair search solved with a customized MCTS algorithm. Through a bi-level search tree that decouples floorplan topology from device assignment and a similarity-driven reward mechanism, AlphaPlacer can exploit historical layouts to mimic expert placement patterns without relying on exhaustive manual constraints. Additionally, the integration of a pre-trained value network that learns from expert layouts significantly enhances search efficiency and convergence quality. Experiments show that AlphaPlacer consistently achieves smaller area and HPWL than baselines, while delivering equal or better post-layout performance, demonstrating that AlphaPlacer is a practical and effective approach for analog placement.

REFERENCES

[1] P. Xu, J. Li, T.-Y. Ho, B. Yu, and K. Zhu, “Performance-driven analog layout automation: Current status and future directions,” in *Proc. ASPDAC*. IEEE, 2024, pp. 679–685.

[2] J. Crossley, A. Puggelli, H.-P. Le, B. Yang, R. Nancollas, K. Jung, L. Kong, N. Narevsky, Y. Lu, N. Sutardja, E. J. An, A. L. Sangiovanni-Vincentelli, and E. Alon, “BAG: A designer-oriented integrated framework for the development of ams circuit generators,” in *Proc. ICCAD*, 2013.

[3] J. Han, W. Bae, E. Chang, Z. Wang, B. Nikolić, and E. Alon, “LAYGO: A template-and-grid-based layout generation engine for advanced CMOS technologies,” *IEEE TCAS I*, vol. 68, no. 3, pp. 1012–1022, 2021.

[4] T. Shin, D. Lee, D. Kim, G. Sung, W. Shin, Y. Jo, H. Park, and J. Han, “LAYGO2: A custom layout generation engine based on dynamic templates and grids for advanced CMOS technologies,” *IEEE TCAD*, 2023.

[5] B. Xu, B. Basaran, M. Su, and D. Z. Pan, “Analog placement constraint extraction and exploration with the application to layout retargeting,” in *Proc. ISPD*, 2018, pp. 98–105.

[6] B. Xu, S. Li, X. Xu, N. Sun, and D. Z. Pan, “Hierarchical and analytical placement techniques for high-performance analog circuits,” in *Proc. ISPD*, 2017, pp. 55–62.

[7] K. Kunal, M. Madhusudan, A. K. Sharma, W. Xu, S. M. Burns, R. Harjani, J. Hu, D. A. Kirkpatrick, and S. S. Sapatnekar, “Align: Open-source analog layout automation from the ground up,” in *Proc. DAC*, 2019, pp. 1–4.

[8] H.-C. Ou, K.-H. Tseng, J.-Y. Liu, I.-P. Wu, and Y.-W. Chang, “Layout-dependent-effects-aware analytical analog placement,” in *Proc. DAC*, 2015, pp. 1–6.

[9] B. Xu, S. Li, C.-W. Pui, D. Liu, L. Shen, Y. Lin, N. Sun, and D. Z. Pan, “Device layer-aware analytical placement for analog circuits,” in *Proc. ISPD*, 2019, pp. 19–26.

[10] H. Chen, M. Liu, X. Tang, K. Zhu, A. Mukherjee, N. Sun, and D. Z. Pan, “MAGICAL 1.0: An open-source fully-automated ams layout synthesis framework verified with a 40-nm 1GS/s $\Delta\sum$ ADC,” in *Proc. CICC*, 2021, pp. 1–2.

[11] F. Balasa and K. Lampaert, “Module placement for analog layout using the sequence-pair representation,” in *Proc. DAC*, 1999, pp. 274–279.

[12] P.-C. Pan, C.-Y. Chin, H.-M. Chen, T.-C. Chen, C.-C. Lee, and J.-C. Lin, “A fast prototyping framework for analog layout migration with planar preservation,” *IEEE TCAD*, vol. 34, no. 9, p. 1373–1386, Sep. 2015.

[13] T.-C. Chen and Y.-W. Chang, “Modern floorplanning based on b/sup */-tree and fast simulated annealing,” *IEEE TCAD*, vol. 25, no. 4, pp. 637–650, 2006.

[14] X. Tang and D. F. Wong, “Floorplanning with alignment and performance constraints,” in *Proc. DAC*, 2002, p. 848–853.

[15] L. Xiao, E. F. Young, X. He, and K.-P. Pun, “Practical placement and routing techniques for analog circuit designs,” in *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2010, pp. 675–679.

[16] F. Balasa, S. C. Maruvada, and K. Krishnamoorthy, “Efficient solution space exploration based on segment trees in analog placement with symmetry constraints,” in *Proc. ICCAD*, 2002, pp. 497–502.

[17] E. F. Y. Tam Yiu-Cheong and C. Chu., “Analog placement with symmetry and other placement constraints,” in *Proc. ICCAD*, 2006, pp. 349–354.

[18] K. Zhu, H. Chen, M. Liu, X. Tang, N. Sun, and D. Z. Pan, “Effective analog/mixed-signal circuit placement considering system signal flow,” in *Proc. ICCAD*, 2020, pp. 1–9.

[19] H.-C. C. Chien, H.-C. Ou, T.-C. Chen, T.-Y. Kuan, and Y.-W. Chang, “Double patterning lithography-aware analog placement,” in *Proc. DAC*, 2013, pp. 1–6.

[20] P.-H. Lin and S.-C. Lin, “Analog placement based on hierarchical module clustering,” in *Proc. DAC*, 2008, pp. 50–55.

[21] Q. Ma, L. Xiao, Y.-C. Tam, and E. F. Young, “Simultaneous handling of symmetry, common centroid, and general placement constraints,” *IEEE TCAD*, vol. 30, no. 1, pp. 85–95, 2010.

[22] P.-Y. Chou, H.-C. Ou, and Y.-W. Chang, “Heterogeneous b*-trees for analog placement with symmetry and regularity considerations,” in *Proc. ICCAD*, 2011, pp. 512–516.

[23] S. Nakatake, M. Kawakita, T. Ito, M. Kojima, M. Kojima, K. Izumi, and T. Habasaki, “Regularity-oriented analog placement with diffusion sharing and well island generation,” in *Proc. ASPDAC*, 2010, pp. 305–311.

[24] P.-H. Lin, H. Zhang, M. D. Wong, and Y.-W. Chang, “Thermal-driven analog placement considering device matching,” in *Proc. DAC*, 2009, pp. 593–598.

[25] Q. Ma, E. F. Y. Young, and K. P. Pun, “Analog placement with common centroid constraints,” in *Proc. ICCAD*, 2007, pp. 579–585.

[26] M. Liu, K. Zhu, J. Gu, L. Shen, X. Tang, N. Sun, and D. Z. Pan, “Towards decrypting the art of analog layout: Placement quality prediction via transfer learning,” in *Proc. DATE*, 2020, pp. 496–501.

[27] Y. Li, Y. Lin, M. Madhusudan, A. Sharma, W. Xu, S. S. Sapatnekar, R. Harjani, and J. Hu, “A customized graph neural network model for guiding analog IC placement,” in *Proc. ICCAD*, 2020, pp. 1–9.

[28] C.-C. Chang, J. Pan, Z. Xie, Y. Li, Y. Lin, J. Hu, and Y. Chen, “Fully automated machine learning model development for analog placement quality prediction,” in *Proc. ASPDAC*, 2023, pp. 58–63.

[29] M. P.-H. Lin, C.-C. Lee, and Y.-C. Hsieh, “Reinforcement learning or simulated annealing for analog placement? a study based on bounded-sliceline grids,” in *Proc. ISPD*, 2024, pp. 143–150.

[30] X. Tang and D. Wong, “Fast-sp: A fast algorithm for block placement based on sequence pair,” in *Proc. ASPDAC*, 2001, pp. 521–526.

[31] L. Wang, R. Fonseca, and Y. Tian, “Learning search space partition for black-box optimization using monte carlo tree search,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 19 511–19 522, 2020.

[32] L. Song, K. Xue, X. Huang, and C. Qian, “Monte carlo tree search based variable selection for high dimensional bayesian optimization,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 28 488–28 501, 2022.

[33] S. Gelly and D. Silver, “Monte-carlo tree search and rapid action value estimation in computer go,” *Artificial Intelligence*, vol. 175, no. 11, pp. 1856–1875, 2011.

[34] P. K. Sen, “Estimates of the regression coefficient based on kendall’s tau,” *Journal of the American Statistical Association*, vol. 63, no. 324, pp. 1379–1389, 1968.

[35] E. Barke, “A network comparison algorithm for layout verification of integrated circuits,” *IEEE TCAD*, vol. 3, no. 2, pp. 135–141, 1984.

[36] G. Pelz and U. Roettcher, “Pattern matching and refinement hybrid approach to circuit comparison,” *IEEE TCAD*, vol. 13, no. 2, pp. 264–276, 1994.

[37] B. Xu, K. Zhu, M. Liu, Y. Lin, S. Li, X. Tang, N. Sun, and D. Z. Pan, “Magical: Toward fully automated analog ic layout leveraging human and machine intelligence: Invited paper,” in *Proc. ICCAD*, 2019, pp. 1–8.

[38] “Gurobi,” https://www.gurobi.com/.