PAPlace: Performance-Driven Differentiable Analog Placement

PENG XU, Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, Hong Kong
YUAN PU, The Chinese University of Hong Kong, Hong Kong, Hong Kong
KEREN ZHU, Fudan University, Shanghai, China
TINGHUAN CHEN, The Chinese University of Hong Kong - Shenzhen, Shenzhen, China
TSUNGYI HO, The Chinese University of Hong Kong, Hong Kong, Hong Kong
BEI YU, CSE, The Chinese University of Hong Kong, Hong Kong, Hong Kong

Analog circuit placement is crucial for optimal performance, but achieving a decent layout demands expertise and time. Recent advances in machine learning techniques have shown promising results in modeling analog layout performance. PAPlace further extends these methods and integrates them into the core analog placement engine, allowing direct optimization of the post-layout performance effectively. Our approach proposes a differentiable prediction model that combines layout and wiring information into a non-linear analog placement engine. We then incorporate the differentiable performance model into a gradient-descent-based global placement engine. A multi-objective optimization method is further proposed to find the common gradient descent direction for different metrics. The experimental results on benchmarks under the TSMC 40nm technology node demonstrate the superiority of the proposed framework compared with the cutting-edge works, with up to 2163.00μV, 73.95dB, 62.25MHz, 57.84dB improvement in Offset Voltage, CMRR, BandWidth, DC Gain metrics.

CCS Concepts: • **Hardware → Placement;**

Additional Key Words and Phrases: Electronic design automation, analog placement, multi-objective optimization, performance prediction

ACM Reference Format:
Peng Xu, Yuan Pu, Keren Zhu, Tinghuan Chen, Tsungyi Ho, and Bei Yu. 2026. PAPlace: Performance-Driven Differentiable Analog Placement. *ACM Trans. Des. Autom. Electron. Syst.* 31, 5, Article 112 (April 2026), 23 pages. https://doi.org/10.1145/3799979

This work is supported by The Research Grants Council of Hong Kong SAR (No. CUHK14211824 and No. CUHK14201624), and partially supported by the National Natural Science Foundation of China under Grants (62504054, 92573108 and 62304197).
Authors' Contact Information: Peng Xu, Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, Hong Kong; e-mail: 1155187705@link.cuhk.edu.hk; Yuan Pu, The Chinese University of Hong Kong, Hong Kong, Hong Kong; e-mail: ypu@cse.cuhk.edu.hk; Keren Zhu, Fudan University, Shanghai, China; e-mail: krzhu@fudan.edu.cn; Tinghuan Chen, The Chinese University of Hong Kong - Shenzhen, Shenzhen, Guangdong, China; e-mail: chentinghuan@cuhk.edu.cn; Tsungyi Ho, The Chinese University of Hong Kong, Hong Kong, Hong Kong; e-mail: tyho@cse.cuhk.edu.hk; Bei Yu (corresponding author), CSE, The Chinese University of Hong Kong, Hong Kong, Hong Kong; e-mail: byu@cse.cuhk.edu.hk.



This work is licensed under a Creative Commons Attribution 4.0 International License.

© 2026 Copyright held by the owner/author(s).
ACM 1084-4309/2026/04-ART112
https://doi.org/10.1145/3799979

1 Introduction

Analog circuit placement is critical to optimal performance, but obtaining a decent circuit layout requires significant time and expertise. Unlike digital circuits, analog circuits are sensitive to layout parasitics and coupling, which can complicate the relationship between performance and layout. Furthermore, multiple performance characteristics are affected by numerous RC elements in an intertwined manner. There is a practical way to produce generalized performance models for layout implementation [1]. Consequently, the current analog placement process does not account for post-layout performance, potentially leading to suboptimal post-layout performance.

While attempting to improve placement performance, the existing analog layout placement methods are mainly focused on optimizing *proxy* objectives for performance [2–5]. Specific geometric constraints, such as symmetry and common centroid, have been considered in [2] to improve the tolerance of layouts to parasitic effects caused by mismatching. However, these methods with indirect objectives lack a vehicle for performance optimization. Another line of research incorporates partial performance metrics directly into layout optimization. For example, Ref. [3] uses a linear approximation model to simulate performance during layout optimization, which limits its ability to handle the nonlinear behavior of many analog circuits. Refs. [4] and [5] consider the signal flow and pole positions as constraints to optimize some performance indicators. These metrics are specific to design and hard to generalize for different analog circuits. In summary, conventional analog placement methods using proxy objectives lack a principal method in performance optimization: the proxy can be inaccurate or not generalizable.

Machine learning can **directly** model placement performance, but existing models have not improved analog placers due to the existing gap. Machine learning techniques have been used in modeling circuit performance, such as the impact of device sizing [6, 7]. Li et al. proposed a circuit performance model based on a customized **graph neural network** (**GNN**) and two optimization techniques to improve circuit performance [8]. In the field of analog layout modeling, there has been emerging research in model placement effects on the post-layout analog performance recently [7–10]. Liu et al. developed a well-designed **convolutional neural network** (**CNN**) model to capture the complex correlation between the layout and final performance [9]. Li et al. also created a customized graph neural network model to estimate the impact of placement on layout performance [7]. Additionally, Chang et al. leveraged **neural architecture search** (**NAS**) to automatically develop customized neural architectures for predicting different analog circuit designs and metrics [10]. However, these methods still do not fill the gap between performance models and analog placers. A key reason is that the relationship between input layout images and module coordinates is not differentiable. This work further extends these methods and integrates them into the core analog placement engine, allowing direct optimization of the circuit performance.

In this work, we propose a novel analog placement framework, **Performance-driven Analog Placement** (**PAPlace**), bridging the recent placement performance modeling and an SOTA nonlinear analog placer.

Specifically, we integrate a neural network model into the placement objective for explicit performance optimization. The main contributions of this paper are listed as follows:

- We develop a differentiable performance-driven analog placement framework that can effectively simulate post-layout performance for layouts. The framework incorporates gradients for different modules into the placement process to achieve better performance.
- To more accurately capture the impact of parasitic effects on post-layout performance, we propose module density (MUDY) and pin density (PINDY) features that combine the layout and wiring information.

— We propose a multi-objective gradient-based optimization method to solve the potential conflicts of different metrics in analog placement.
Two different modes of performance model construction are proposed to overcome the data challenges.
— Experimental results show that our framework outperforms the MAGICAL placement method [4] assisted by random search and multi-objective Bayesian optimization [11] in multiple post-layout simulation metrics.

2 Background and Preliminaries

This section provides an overview of the analog placement algorithm (Section 2.1) and performance prediction (Section 2.2), followed by the formulation of the performance-driven analog placement problem (Section 2.5).

2.1 Analog Placement Algorithm

The problem of analog placement has been extensively studied with various solving methods [1]. Analytical analog placement methods, especially **non-linear programming** (**NLP**), have recently become popular [12, 13]. NLP has two important advantages: it can efficiently solve the target and is flexible in handling multiple optimization objectives [14]. Typical NLP-based analog placer [4] involves global placement and legalization stages: hard constraints are first relaxed in the NLP-based global placement and are legalized in the second stage, as described in [4].

Researchers used solving algorithms to directly optimize the analog placement objectives with NLP [12, 13], as it is suitable to optimize non-linear objective functions and it has advantages in scalability theoretically. Typically, NLP-based analog placer needs to relax the constraints and therefore requires two stages: Global Placement and Legalization.

In the widely-used MAGICAL placer, global placement aims to spread the modules in the layout using gradient descent-based optimization to optimize placement objectives and the constraint penalties [15].

$$\min_{\boldsymbol{x},\boldsymbol{y}} \quad f_{\text{NLP}}(\boldsymbol{x},\boldsymbol{y}) = f_{WL} + a \cdot f_{OL} + b \cdot f_{BND} + c \cdot f_{SYM}, \tag{1}$$

where the objective function consists of four terms: wirelength f_{WL}, overlap penalty f_{OL}, boundary penalty f_{BND}, and symmetry penalty f_{SYM}. Coefficients a, b, and c are used to balance the different objective terms.

The wirelength objective f_{WL} is the total **half-perimeter wirelength** (**HPWL**). The overlap penalty f_{OL} is modeled as an area overlap function, which is modeled as an area overlap function similar to [16]. f_{BND} is the penalty for violating the boundary constraints. A desirable placement bounding box for the design is derived from controlling the circuit area, white space ratio, and aspect ratio.

The last term f_{SYM} penalizes the violation of the symmetry constraint, which requires: (1) each symmetric device pair within the same group to be symmetric with respect to the same axis; (2) the self-symmetric devices to be self-symmetric with respect to the same axis of the group. After the global placement stage, **linear-programming** (**LP**)-based legalization is used to correct small violations of overlapping and asymmetry, as described in [4].

2.2 Placement Features for Performance Prediction

To focus on the differences in performance metrics and layout design space for different analog functional building blocks, we have limited our study to **operational transconductance amplifiers** (**OTA**) designs.

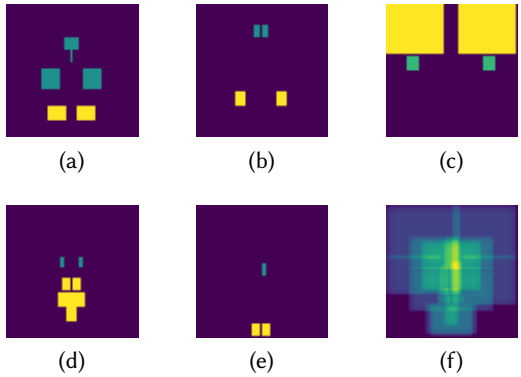


Fig. 1. (a) First Stage devices; (b) Other Stages devices; (c) Feedback devices; (d) Common-mode feedback devices; (e) Bias Deives; and (f) Routing Demand.

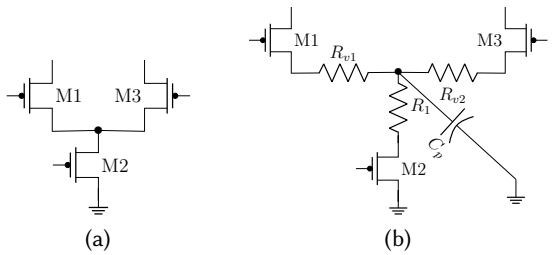


Fig. 2. Modeling of interconnect parasitics: (a) Differential pair without parasitic; (b) Differential pair with parasitic capacitor and resistors.

In our performance-driven analog placement framework, we use a machine learning-based performance prediction model as a vehicle to enable direct optimization over the performance metrics. As analog circuit layout can be naturally viewed as images, we represent intermediate layout placement results as 2D images similar to Ref. [9].

As shown in Figure 1, we divide the circuit into the following subcircuits based on functionality for OTA circuits:

— **First Stage** Devices in the first stage. This includes the differential input, load, and tail transistors.
— **Other Stages** Devices in the other amplifier stages.
— **Feedback** Passive devices in the compensation feedback loop, such as Miller capacitance.
— **Common-mode feedback** Devices in Common-mode feedback circuits.
— **Bias** Devices in the current mirrors.
— **Routing Demand** The aggregated pin boundary box for each net.

2.3 Theoretical Analysis with the Relationship between Layout and Performance

Layout affects circuit performance primarily through device mismatch and interconnect parasitics. Mismatch-induced degradation follows Pelgrom's model [17], where sensitivity-weighted variations depend on the device separation distance in placement $D(\boldsymbol{x}, \boldsymbol{y})$. As shown in Figure 2, the interconnect parasitics are modeled using the star-network formulation of Lampaert [18], with

parasitic capacitances and resistances estimated from wirelength and weighted by their corresponding performance sensitivities. The total performance degradation is the sum of the mismatch and interconnect contributions, which can be expressed as $\Delta P_t = \Delta P_{t,\text{module}} + \Delta P_{t,\text{net}}$.

Because analytical sensitivity calculation is computationally expensive, we train a neural network to predict the NLP degradation ΔP_t directly from placement coordinates. However, directly using regression to predict the absolute value of the performance degradation term will be inaccurate. We convert this problem into a binary classification problem. Based on the relative magnitude of the post-layout performance, we divide the top 50% of the layouts into class 0, and the remaining 50% of the layouts into class 1.

$$y_i(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 0, & \text{if} \quad P_t \geq P_{t,50\%}; \\ 1, & \text{else.} \end{cases}$$

To incorporate the influence of device mismatches and interconnect parasitic effects on the post-layout performance during the prediction process, we proposed two types of features, MUDY and PINDY, in Section 3.1.

2.4 Comparison with Fully-Synthesizable Analog Design Approaches

Recent research into fully synthesizable analog blocks, such as fully digital or standard-cell-based OTAs, represents a parallel trajectory to our work [19–21]. While these flows offer significant advantages in digital tool compatibility, voltage scalability, and IP reuse, they often necessitate a compromise in raw analog performance. In contrast, PAPlace is tailored for performance-critical custom analog blocks, where mitigating layout-dependent effects at the transistor level is essential. Consequently, rather than competing with fully synthesizable analogs, our method augments the design ecosystem. PAPlace can be used to generate high-performance sub-modules, serving as foundational building blocks, which can then be integrated into larger systems alongside fully synthesizable components.

2.5 Problem Formulation

To enhance analog layout performance, optimizing placement parameters towards different objectives is crucial. A common approach is using a user-defined **figure-of-merit** (**FOM**) representation, a weighted sum of post-layout simulation metrics [22]. However, performance-driven analog placement requires modeling tradeoffs between conflicting metrics beyond linear combinations. An alternative approach is finding conflict-free optimal solutions, known as multi-objective optimization. This problem involves finding such solutions given multiple criteria. In performance-driven analog placement, we specify the multi-objective optimization using a vector-valued loss $\boldsymbol{L}$:

$$\min_{\boldsymbol{x},\boldsymbol{y}} \boldsymbol{L}\left(\boldsymbol{x}, \boldsymbol{y}\right) = \min_{\boldsymbol{x},\boldsymbol{y}} \left(L_1\left(\boldsymbol{x}, \boldsymbol{y}\right), \ldots, L_T\left(\boldsymbol{x}, \boldsymbol{y}\right)\right)^\top. \tag{2}$$

We define conflict-free optimality for performance-driven analog placement in the following:

Definition 1 (Performance-driven Analog Placement). (1) A solution $(\boldsymbol{x}, \boldsymbol{y})$ dominates a solution if for all metrics t $\boldsymbol{L}(\boldsymbol{x}, \boldsymbol{y}) \leq \boldsymbol{L}(\boldsymbol{x}', \boldsymbol{y}')$ and $\boldsymbol{L}(\boldsymbol{x}, \boldsymbol{y}) \neq \boldsymbol{L}(\boldsymbol{x}', \boldsymbol{y}')$.
(2) A solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is called conflict-free optimal is there exists no solution $(\boldsymbol{x}, \boldsymbol{y})$ dominates $(\boldsymbol{x}^*, \boldsymbol{y}^*)$.

With the definition of conflict-free optimality, we include an additional penalty term in our objective function and formulate a new optimization problem to integrate performance prediction information into the analog placement framework, as shown below:

$$\min_{\boldsymbol{x},\boldsymbol{y}} \quad f_{\text{PAPlace}}(\boldsymbol{x}, \boldsymbol{y}) = f_{\text{NLP}}(\boldsymbol{x}, \boldsymbol{y}) + \rho \cdot L(\boldsymbol{x}, \boldsymbol{y}), \tag{3}$$

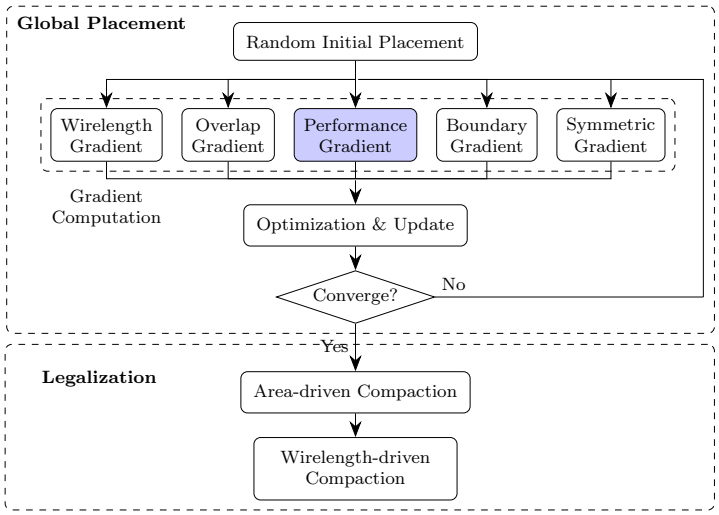


Fig. 3. The overall flow of our framework.

where the objective $L(\boldsymbol{x}, \boldsymbol{y})$ indicates the metrics of post-layout performance. The performance objective $L(\boldsymbol{x}, \boldsymbol{y})$ can be expressed as a weighted sum of multiple post-layout performance metrics, but here we aim at obtaining a conflict-free solution in this form:

$$\begin{aligned} L(\boldsymbol{x}, \boldsymbol{y}) = {} & c_1 \cdot L_{\text{OV}} + c_2 \cdot L_{\text{CMRR}} \\ & + c_3 \cdot L_{\text{UGB}} + c_4 \cdot L_{\text{Gain}} + c_5 \cdot L_{\text{Noise}}, \end{aligned} \tag{4}$$

where the coefficients are dynamically determined using a gradient-based multi-objective optimization method.

3 Performance-Driven Analog Placement

This section presents the proposed PAPlace framework flow and details the algorithms. Figure 3 shows the overall flow of our placement framework. Our placement algorithm differs from traditional analog methods that rely on optimizing the proxy targets. Instead, we explicitly incorporate model-based performance gradients into the optimization process and use these gradients to guide placement.

The gradient descent-based analog placement with neural network-based performance prediction allows the automatic analog placement to directly perform optimization over the post-layout performance. However, there are several major algorithmic challenges in enabling this performance-driven analog placement methodology.

- **Fully differentiable performance model:** The existing performance prediction models usually use the extracted layout images as input features [9]. The relation between the placed module coordinates and the extracted images is not differentiable. In PAPlace, we propose a custom performance prediction method to allow the model to be differentiable from the module coordinates to the predicted performance metrics.
- **Multi-objective optimization:** The analog circuit designs naturally incorporate different types of performance metrics. For example, the performance specification of a typical OTA usually covers different metrics, such as Gain and CMRR. Many of those metrics can conflict with each other, resulting in trade-offs between different metrics. In PAPlace, we further

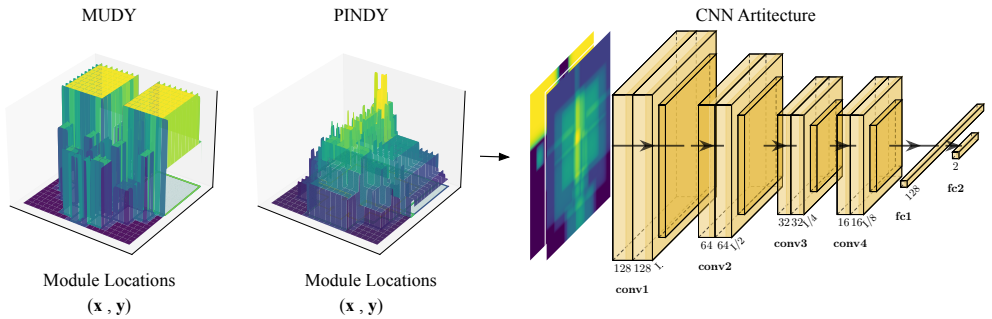


Fig. 4. Differentiable MUDY and PINDY features used for performance prediction.

develop a multi-objective optimization algorithm to enable the efficient exploration of multiple performance objectives.

— **Different modes of performance model construction:** Constructing accurate models for analog placement is challenging due to the difficulty in obtaining sufficient high-quality data. To address this, we propose two modes of obtaining performance models: (1) PAPlace-S (slow mode), which trains from scratch with balanced sampling data; PAPlace-F (fast mode), which transfers from the pre-trained model.

In the rest of this section, we introduce our fully differentiable performance prediction models (Section 3.1), the performance-driven analog global placement (Section 3.2), the multi-objective optimization algorithm (Section 3.3), and different modes of obtaining performance model (Section 3.4).

3.1 Performance Prediction Based on the Layout and Wiring Information

Placement decides the layout implementation's overall floorplan and wiring topology. Therefore, placement solutions usually critically impact post-layout parasitic effects and circuit performance. Our intuition is that parasitic effects arise from both interactions between different modules and interactions between wires caused by routing. To better capture the impact of parasitic effects on post-simulation performance, we will simultaneously incorporate both of these features into our predictions.

To accurately predict the impact of parasitic effects on post-simulation performance, PAPlace uses two types of features to effectively represent the placement solutions: module density (MUDY) and PIN routing demand (PINDY). By introducing these two different features, we attempt to use them to predict the impact of mismatches and interconnect parasitic effects on the final circuit performance. Then, we use a CNN structure to fit the relative influence of the post-simulation performance value as mentioned in Section 2.3, which can be represented as

$$\begin{aligned} f_P(\Delta P_t) &= f_P(\Delta P_{t,module}(\boldsymbol{x},\boldsymbol{y}) + \Delta P_{t,net}(\boldsymbol{x},\boldsymbol{y})) \\ &\approx f_P(\text{CNN}(\text{MUDY}(\boldsymbol{x},\boldsymbol{y}), \text{PINDY}(\boldsymbol{x},\boldsymbol{y}))), \end{aligned} \tag{5}$$

where CNN is the used neural network encoder, and f_p represents the linear classifier layers as shown in Figure 4.

MUDY and PINDY are constructed in a differentiable process, allowing PAPlace to perform performance optimization directly using the prediction models. We will describe these two features in detail next.

Module Density Features. We propose a differentiable feature extraction, module density features (MUDY), to represent the module placements. Intuitively, module density indicates whether a module covers an area. However, as in the previous approach [23], directly defining module density based on coverage is not differentiable.

To calculate MUDY, we first get the bounding box of each module,

$$x_i^l = x_i,\ x_i^h = x_i + W_i,\ y_i^l = y_i,\ y_i^h = y_i + H_i, \tag{6}$$

where m_i denotes the i^{th} module in the set of modules specified in the circuit netlist $\mathbb{M}$, whose coordinate of the bottom-left corner is (x_i, y_i). W_i and H_i are the width and height of m_i respectively. In the region $x \in [x_i^l, x_i^h], y \in [y_i^l, y_i^h]$, we simply define the MUDY for nodule m_i as

$$\text{MUDY}_{m_i}(x, y) = f_{\Phi_x}(x_i^l, x, x_i^u) \cdot f_{\Phi_y}(y_i^l, y, y_i^u), \tag{7}$$

where f_{Φ_x} and f_{Φ_y} are density functions defined in the vertical and horizontal directions using the sigmoid function, respectively. If $x \notin [x_{m_i}^l, x_{m_i}^h]$ or $y \notin [y_e^l, y_e^h]$, we have $\text{MUDY}_{m_i}(\boldsymbol{x}, \boldsymbol{y}) = 0$.

To make the density function differentiable, we can use the sigmoid function to smooth it [24]. Specifically, we can define f_{Φ_x} and f_{Φ_y} as

$$\begin{aligned} f_{\Phi_x}(x_i^l, x, x_i^u) &= p(x - x_i^l)p(x_i^u - p_x), \\ f_{\Phi_y}(y_i^l, y, y_i^u) &= p(y - y_i^l)p(y_i^u - p_y), \\ p(t) &= \frac{1}{1 + e^{-\beta t}}, \end{aligned} \tag{8}$$

where $p(t)$ is the sigmoid function, which approximates exact 0-1 logic function when hyperparameter $\beta \to 0$.

In practice, the MUDY features are divided into $M \times N$ grid modules. The MUDY of a grid-cell $b_{k,l}$ is calculated by summing up the MUDY values of the modules in it:

$$\text{MUDY}(k, l) = \sum_{m_i \in b_{k,l}} \text{MUDY}_{m_i}(k, l)\,\text{Type}(m_i), \tag{9}$$

where $\text{Type}(m_i)$ is defined by the functionality of m_i as in Section 2.2.

Pin Density Features. To calculate routing demand features, we introduce the Pin Density (PINDY), which is the pin density map inspired by rectangular uniform wire density (RUDY) [25] and PINRUDY [23]. Here, we use the unnormalized version of PINRUDY in anticipation that it can reflect the specific routing requirements.

To compute the PINDY, we first obtain the bounding box of a net,

$$x_e^h = \max_{p_e} x_{p_e},\ x_e^l = \min_{p_e} x_{p_e},\ y_e^h = \max_{p_e} y_{p_e},\ y_e^l = \min_{p_e} y_{p_e}, \tag{10}$$

where p_e denotes a pin in the net e, whose location is (x_{p_e}, y_{p_e}). Then, we split the layout into an $M \times N$ grid and estimate the pin density of each grid-cell $b_{k,l}$. The PINDY of a pin is calculated with,

$$\text{PINDY}_{p_e}(k, l) = f_{\Phi_x}(x_e^l, k, x_e^h) \cdot f_{\Phi_y}(y_e^l, l, y_e^h), \tag{11}$$

where $(x_{p_e}, y_{p_e}) \in b_{k,l}$. Finally, the PINDY of the grid-cell $b_{k,l}$ is defined as

$$\text{PINDY}(k, l) = \sum_{p_e \in b_{k,l}} \text{PINDY}_{p_e}(k, l). \tag{12}$$

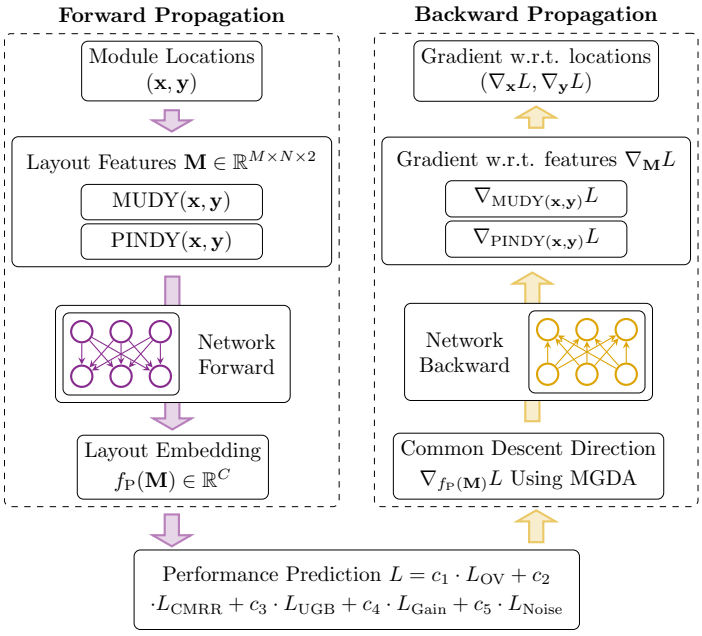


Fig. 5. The optimization flow for **performance gradient** updating.

3.2 Performance-Driven Analog Global Placement

The process flow illustrated in Figure 5 explains how the gradients of the performance prediction concerning module locations are calculated. We used MUDY and PINDY to obtain the differentiable relationship between module coordinates and the post-layout performance, and then we used the two differentiable functions to get the gradient values for those coordinates. Subsequently, we passed these gradients to the global placement engine as the gradients of performance concerning the modules.

First, the MUDY and PINDY features are extracted from the module locations $(\boldsymbol{x}, \boldsymbol{y})$, representing the layout and wiring information. These two features are combined into a two-channel feature map $\mathbf{M} \in R^{M\times N\times 2}$. Then, A pre-trained model takes this feature map $\mathbf{M}$ as input and outputs a prediction result $L(\boldsymbol{x}, \boldsymbol{y})$ for the current layout. Finally, we find the common gradient descent direction of different performance metrics as the performance guidance gradient for the module, and the details of common gradient descent will be illustrated in Section 3.3.

Since gradients are only defined on vector fields, we use the notation $\nabla_{\boldsymbol{A}}$ to represent taking a derivative concerning a matrix $\boldsymbol{A}$ of size $M \times N$ as the gradient concerning its vectorized representation, for simplicity.

$$\begin{aligned} \nabla_{\boldsymbol{A}} f &:= \nabla_{\overrightarrow{\boldsymbol{A}}} f, \\ \overrightarrow{\boldsymbol{A}} &:= \left[a_{1,1}, \ldots, a_{M,1}, \ldots, a_{1,N}, \ldots, a_{M,N}\right]^{\top}. \end{aligned} \tag{13}$$

To perform gradient-based optimization, we need to calculate the gradients of the performance penalty function L, as defined in Equation (3), w.r.t module locations. We use the chain rule to compute the gradients, which have three consecutive parts as illustrated in Figure 5.

(1) Compute gradient w.r.t layout embedding: $\nabla_{f_{\mathrm{P}(\mathbf{M})}} L$;
(2) Back-propagate $\nabla_{f_{\mathrm{P}(\mathbf{M})}} L$ through the trained neural network model to obtain the gradient w.r.t stacked features: $\nabla_{\mathbf{M}} L$.

(3) Extract different channels of $\nabla_{\mathbf{M}} L$ as gradients w.r.t two feature maps, compute the gradient w.r.t module locations $\nabla_{\boldsymbol{x},\boldsymbol{y}} L$ and sum them together.

Since we can use the back-propagation to obtain the gradient w.r.t the input features, $\nabla_{\mathbf{M}} L$, with the auto-grad mechanism [26], we can focus on computing the gradient w.r.t module locations. The gradient w.r.t the input features $\mathbf{M}$ has two channels, which are $\nabla_{\mathrm{MUDY}(\boldsymbol{x},\boldsymbol{y})} L$ and $\nabla_{\mathrm{PINDY}(\boldsymbol{x},\boldsymbol{y})} L$. Therefore, the gradient with respect to cell location $\boldsymbol{x}, \boldsymbol{y}$ can be rewritten as follows:

$$\begin{aligned} \nabla_{\boldsymbol{x},\boldsymbol{y}} L = &\mathbf{J}_{\boldsymbol{x},\boldsymbol{y}}(\mathrm{MUDY}(\boldsymbol{x},\boldsymbol{y}))^{\top} \cdot \nabla_{\mathrm{MUDY}(\boldsymbol{x},\boldsymbol{y})} L \\ &+ \mathbf{J}_{\boldsymbol{x},\boldsymbol{y}}(\mathrm{PINDY}(\boldsymbol{x},\boldsymbol{y}))^{\top} \cdot \nabla_{\mathrm{PINDY}(\boldsymbol{x},\boldsymbol{y})} L. \end{aligned} \tag{14}$$

Extract MUDY **Gradient**. From Equation (7) of the MUDY features, we can explicitly find the Jacobian matrices on the right-hand side of Equation (14). The Jacobian matrix $\mathbf{J}_{\boldsymbol{x},\boldsymbol{y}}(\mathrm{MUDY}(\boldsymbol{x},\boldsymbol{y}))$ can be calculated as:

$$\mathbf{J}_{\boldsymbol{x},\boldsymbol{y}}(\mathrm{MUDY}(\boldsymbol{x},\boldsymbol{y})) = \sum_{m_i \in \mathcal{M}} \mathbf{J}_{\boldsymbol{x},\boldsymbol{y}} \left(\mathrm{MUDY}_{m_i}(\boldsymbol{x},\boldsymbol{y})\right). \tag{15}$$

The entry at jth row and kth column of $\mathbf{J}_{\boldsymbol{x},\boldsymbol{y}} \left(\mathrm{MUDY}_{m_i}(\boldsymbol{x},\boldsymbol{y})\right)$ w.r.t horizontal location x_k is

$$\nabla_{x_k} \overrightarrow{\mathrm{MUDY}}_{m_j}(\boldsymbol{x},\boldsymbol{y})_{j} \propto \nabla_{x_k} f_{\Phi_x}(x_k, j^{'}, x_k + W_k), \tag{16}$$

where $j^{'} \coloneqq k \bmod M$ and $k^{'} \coloneqq \mathrm{floor}(k/M)$ represents the corresponding bin location of i-th entry of vectorized $\mathrm{MUDY}_i(\boldsymbol{x},\boldsymbol{y})$ map. We cancel out f_{Φ_y} as it is not related to horizontal locations.

As the density function is the product of two functions w.r.t the bottom-left corner (x_k, y_k) and the above-right corner $(x_k + W_k, y_k + H_k)$, the derivative can be calculated as

$$\begin{aligned} \frac{\partial f_{\Phi_x}(x_k, j^{'}, x_k + W_k)}{\partial x_k} = &\frac{\partial p(j^{'} - x_k)}{\partial x_k} \cdot p(x_k + W_k - j^{'}) \\ &+ p(j^{'} - x_k) \cdot \frac{\partial p(x_k + W_k - j^{'})}{\partial x_k}. \end{aligned} \tag{17}$$

Because the derivative of the sigmoid function $p(t) = \frac{1}{1+e^{-\beta t}}$ is $\frac{\partial p(t)}{\partial t} = p(t)(1 - p(t))$, we can obtain the gradient for the righthand side of Equation (17) as

$$\begin{aligned} \frac{\partial f_{\Phi_x}(x_e^l, j^{'}, x_e^h)}{\partial x_k} = &p(t_1) \cdot (1 - p(t_1)) \cdot p(t_2) \\ &+ p(t_1) \cdot p(t_2) \cdot (1 - p(t_2)), \end{aligned} \tag{18}$$

where $t_1 = j^{'} - x_k$ and $t_2 = x_k + W_k - j^{'}$.

By completing this step, we have completed the gradient-solving process for the MUDY feature.

Extract PINDY **Gradient**. Since the PINDY functions are not differentiable everywhere, we replace gradients with subgradients as in [23]. Similar to calculate gradient of the MUDY feature, the entry at i-th row and j-th column of $\mathbf{J}_{\boldsymbol{x},\boldsymbol{y}}$ $(\mathrm{PINDY}_e(\boldsymbol{x},\boldsymbol{y}))$ w.r.t horizontal location x_j is

$$\nabla_{x_j} \overrightarrow{\mathrm{PINDY}}_e(\boldsymbol{x},\boldsymbol{y})_i \propto \nabla_{x_j} f_{\Phi_x}(x_e^l, i^{'}, x_e^h), \tag{19}$$

where $i^{'} \coloneqq i \bmod M$ and $j^{'} \coloneqq \mathrm{floor}(i/M)$ are corresponding to the bin location of ith entry of vectorized $\mathrm{PINDY}_e(\boldsymbol{x},\boldsymbol{y})$ map.

Suppose that the jth module at (x_j, y_j) has a pin at $(\widetilde{x_j}, \widetilde{y_j})$ connected to net e, the derivative can be set as

$$\frac{\partial x_e^l}{\partial x_j} = \delta_{jl}, \frac{\partial x_e^h}{\partial x_j} = \delta_{jk}, \tag{20}$$

where $k \coloneqq \text{argmax}_{(\tilde{x_t},\tilde{y_t})\in e}\, \tilde{x_t}$ and $l \coloneqq \text{argmin}_{(\tilde{x_t},\tilde{y_t})\in e}\, \tilde{x_t}$ are the indices of pins that have a maximal and minimal horizontal coordinate in this net respectively. The notation δ_{jk} is the Kronecker delta function [27]. The Kronecker delta function is typically expressed in the following form :

$$\delta_{ij} = \begin{cases} 1, & \text{if } j = k \\ 0, & \text{if } j \neq k \end{cases} \tag{21}$$

With the help of the Kronecker delta function, we can derive the gradient w.r.t x_j of the jth module as

$$\begin{aligned} \frac{\partial f_{\Phi_x}(x_e^l, i^{'}, x_e^h)}{\partial x_j} =& \delta_{il} \cdot p(t_1) \cdot (1 - p(t_1)) \cdot p(t_2) \\ &+ \delta_{il} \cdot p(t_1) \cdot p(t_2) \cdot (1 - p(t_2)), \end{aligned} \tag{22}$$

where $t_1 = i^{'} - x_j$ and $t_2 = x_j + W_j - i^{'}$.

By completing this step, we have completed the gradient-solving process for the PINDY feature. We backpropagate the gradient w.r.t MUDY features and PINDY features to Equation (14), then propagate the gradient and calculate $\nabla_{\boldsymbol{x},\boldsymbol{y}} L$.

Let $f_{\text{PAPlace}}(\boldsymbol{x},\boldsymbol{y})$ be the objective function that is required to be minimized in Equation (3).

$$\nabla_{\boldsymbol{x},\boldsymbol{y}} f_{\text{PAPlace}}(\boldsymbol{x},\boldsymbol{y}) = \nabla_{\boldsymbol{x},\boldsymbol{y}} f_{\text{NLP}}(\boldsymbol{x},\boldsymbol{y}) + \rho \cdot \nabla_{\boldsymbol{x},\boldsymbol{y}} L(\boldsymbol{x},\boldsymbol{y}) \tag{23}$$

Our proposed performance-driven analog placer uses the performance gradients $(\nabla_{\boldsymbol{x}} f_{\text{PAPlace}}, \nabla_{\boldsymbol{y}} f_{\text{PAPlace}})$ to adjust all movable modules and optimize the objective function.

3.3 Multi-Objective Optimization Algorithm for PAPlace

Similar to single-objective optimization, multi-objective optimization can be solved to local optimality using gradient descent. One approach, known as the **multiple gradient descent algorithm (MGDA)** [28], is based on the **Karush–Kuhn–Tucker (KKT)** conditions, which are necessary for optimality [28].

We now state the KKT conditions for placement-related parameters as follows:

Definition 2 (KKT conditions for Performance-driven Analog Placement). There exist $\alpha^1, \ldots, \alpha^T \geq 0$ such that $\sum_{t=1}^{T} \alpha^t = 1$ and $\sum_{t=1}^{T} \alpha^t \nabla_{\boldsymbol{x},\boldsymbol{y}} L_t\,(\boldsymbol{x},\boldsymbol{y}) = 0$.

In the context of performance-driven analog placement with conflicting objectives, a conflict-free point is a solution that satisfies all requirements. We can use this definition to approach the following optimization problem:

$$\min_{\alpha^1,\ldots,\alpha^T} \left\{ \left\| \sum_{t=1}^{\top} \alpha^t \nabla_{\boldsymbol{x},\boldsymbol{y}} L_t\,(\boldsymbol{x},\boldsymbol{y}) \right\|_2^2 \mid \sum_{t=1}^{T} \alpha^t = 1, \alpha^t \geq 0 \quad \forall t \right\} \tag{24}$$

According to Ref. [28], the solution to the optimization problem Equation (24) is either 0 and satisfies the KKT conditions, or it provides a descent direction that improves all tasks. Therefore, the resulting placement algorithm would be gradient descent on the placement-related parameters, followed by solving Equation (24), and applying the solution $(\sum_{t=1}^{\top} \alpha^t \nabla_{\boldsymbol{x},\boldsymbol{y}} L_t\,(\boldsymbol{x},\boldsymbol{y}))$ as a gradient update to module locations.

In fact, solving the optimization problem defined in Equation (24) is equivalent to finding a minimum-norm point in the convex hull of the input point set. Ozan Sener and Vladlen Koltun [29] propose using a convex optimization approach to solve the convex quadratic problem with linear constraints. In the case of two objectives, the optimization problem can be transformed into:

$$\min_{\gamma\in[0,1]} \left\| \gamma \nabla_{\boldsymbol{x},\boldsymbol{y}} L_1\,(\boldsymbol{x},\boldsymbol{y}) + (1-\gamma) \nabla_{\boldsymbol{x},\boldsymbol{y}} L_2\,(\boldsymbol{x},\boldsymbol{y}) \right\|_2^2, \tag{25}$$

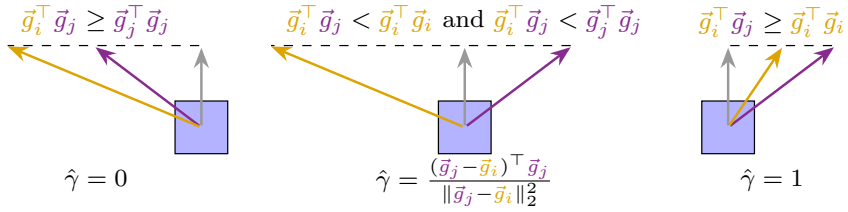


Fig. 6. The visualization shows the process of finding the common gradient descent direction with respect to module m_i for a two-objective case, $\hat{\gamma} = \min_{\gamma \in [0,1]} \left\| \gamma \boldsymbol{g}_i + (1-\gamma)\boldsymbol{g}_j \right\|_2^2$. The solution is either an edge case or a perpendicular vector, as suggested by the geometry.

ALGORITHM 1: Common Gradient Descent for PAPlace

Input: Physical locations $\boldsymbol{x}, \boldsymbol{y}$ of modules specified in the circuit netlist $\mathcal{M}$.

for all module m_i specified in the circuit netlist $\mathcal{M}$ **do**
  $(x^i, y^i) \leftarrow$ Coordinates of module m_i;
  $g^1, \ldots, g^T \leftarrow \nabla_{(x^i, y^i)} L_i(x^i, y^i)$;
  $\alpha^1, \ldots, \alpha^T \leftarrow$ FRANKWOLFESOLVER$(g^1, \ldots, g^T)$; ▷ Solve Equation (24) to find a common descent direction
  $x^i \leftarrow x^i - \eta \sum_{t=1}^T \alpha^t \nabla_{x^i} L_t(x^i, y^i)$;
  $y^i \leftarrow y^i - \eta \sum_{t=1}^T \alpha^t \nabla_{y^i} L_t(x^i, y^i)$;
end for
function FRANKWOLFESOLVER$(g^1, \ldots, g^T)$
  Initialize $\boldsymbol{\alpha} = (\alpha^1, \ldots, \alpha^T) = (\frac{1}{T}, \ldots, \frac{1}{T})$;
  Precompute $\boldsymbol{G}$ st. $\boldsymbol{G}_{i,j} =< g_i, g_j >$;
  repeat
    $\hat{t} \leftarrow \arg\min_r \sum_t \alpha^t \boldsymbol{G}_{rt}$;
    $\hat{\gamma} \leftarrow \arg\min_\gamma \left((1-\gamma)\boldsymbol{\alpha} + \gamma \boldsymbol{e}_{\hat{t}}\right)^\top \boldsymbol{G} \left((1-\gamma)\boldsymbol{\alpha} + \gamma \boldsymbol{e}_{\hat{t}}\right)$; ▷ Using Equation (26)
    $\boldsymbol{\alpha} \leftarrow (1-\hat{\gamma})\boldsymbol{\alpha} + \hat{\gamma}\boldsymbol{e}_{\hat{t}}$;
  until $\hat{\gamma} \sim 0$ or Number of Iterations Limit
  return $\alpha^1, \ldots, \alpha^T$.
end function

and the analytical solution for this one-dimensional problem can be expressed as

$$\hat{\gamma} = \left[\frac{\left(\nabla_{\boldsymbol{x},\boldsymbol{y}} L_2(\boldsymbol{x},\boldsymbol{y}) - \nabla_{\boldsymbol{x},\boldsymbol{y}} L_1(\boldsymbol{x},\boldsymbol{y})\right)^\top \nabla_{\boldsymbol{x},\boldsymbol{y}} L_2(\boldsymbol{x},\boldsymbol{y})}{\left\| \nabla_{\boldsymbol{x},\boldsymbol{y}} L_1(\boldsymbol{x},\boldsymbol{y}) - \nabla_{\boldsymbol{x},\boldsymbol{y}} L_2(\boldsymbol{x},\boldsymbol{y}) \right\|_2^2} \right]_{+,\frac{1}{T}}, \tag{26}$$

where $[\cdot]_{+,\frac{1}{T}}$ represents clipping to $[0,1]$ as $[a]_{+,\frac{1}{T}} = \max(\min(a,1),0)$. We visualize the analytical solution for the one-dimensional problem in Figure 6. However, the analytical solution presented in Figure 6 applies only to the two-objective case. For multiple objectives, there is no direct analytical solution available from the geometric method. Therefore, we need to employ an effective optimization strategy to extend the geometric solution to handle multiple objectives.

To generate the solution to multiple objectives, we can adapt the Frank–Wolfe algorithm [30] to solve the problem efficiently. The Frank–Wolfe algorithm [30] is an iterative first-order optimization algorithm for constrained convex optimization. In each iteration, the Frank–Wolfe algorithm considers a linear approximation of the objective function and moves towards a minimizer of this linear function. Similarly, we can use the analytical solutions for two optimization objectives as the linear approximation in each iteration. This process is repeated until the maximum number of iterations is reached.

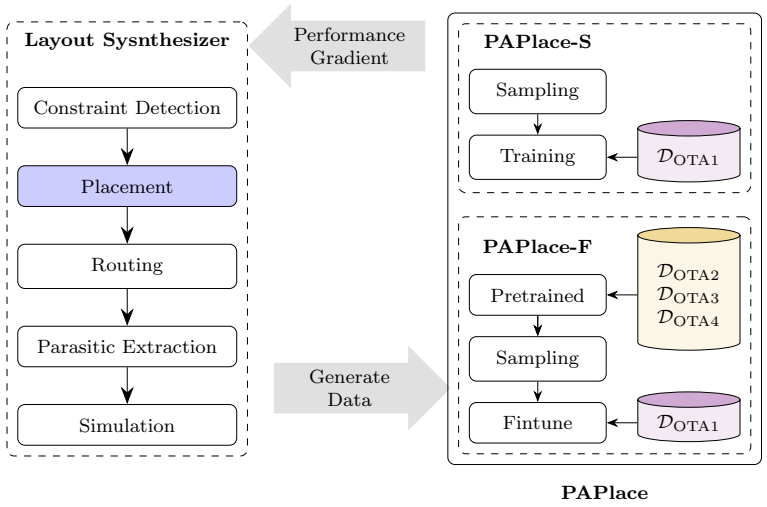


Fig. 7. The different modes of obtaining a performance prediction model for PAPlace.

We give the updating procedure for using the Frank-Wolfe solver to optimize the performance-driven analog placement problem in Algorithm 1.

3.4 Different Modes of Performance Model Construction

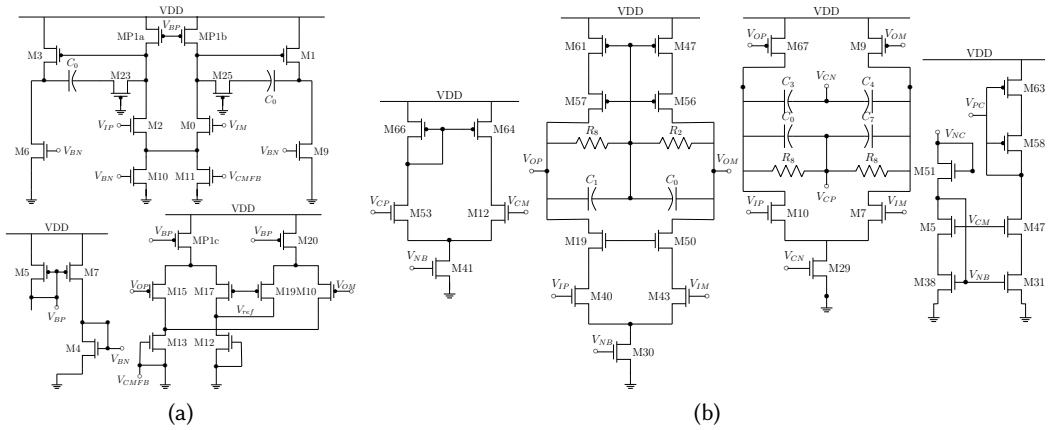
Accurate model predictions are crucial for our framework to guide placements effectively. However, data quantity and quality greatly affect model accuracy, which is challenging in complex and time-consuming analog post-layout simulation. This is particularly challenging as the analog post-layout simulation is often complex and time-consuming. The challenge lies in obtaining a high-quality model with limited data from the target design. To address this, we propose two modes for performance model construction in the PAPlace framework: PAPlace-S and PAPlace-F.

PAPlace-S. If no layout data is available for the current design, we propose collecting a small amount of sampling data. We then model the prediction problem as a binary classification problem and use balanced sampling to enable the model to achieve accurate predictions with a small amount of data, as in Ref. [9]. The quality labels are binary classification labels assigned separately for each performance metric. Since this mode requires a sampling phase that takes more time, we call this the PAPlace slow mode (PAPlace-S).

PAPlace-F. Another way is to leverage the transferability of the pre-trained model obtained from other designs. In this way, we can obtain a relatively accurate model with a few samples through fine-tuning. PAPlace-F uses only 20 samples for fine-tuning, which represents a practical trade-off between accuracy and data collection cost. This choice is justified by our experimental results showing that even with limited samples, reasonable accuracy can be achieved, and PAPlace-F still outperforms baseline methods despite using limited samples. Since this mode requires less time and is more practically useful, we call this the PAPlace fast mode (PAPlace-F) and emphasize it as the recommended approach for practical deployment.

4 Experimental Results

We implement the proposed performance-driven analog placement framework with Python 3.7.5 and C++. The main part of the analog placement and routing algorithms is implemented in C++ based on the MAGICAL framework [15]. The gradient extraction and feature generation parts



(a) (b)

Fig. 8. The topology description of two designs with two sizings: (a) Miller-compensated operational transconductance amplifiers (OTA1 and OTA2); (b) Telescopic operational transconductance amplifiers (OTA3 and OTA4).

are implemented with torch-1.13.1 [26]. All designs are in TSMC 40nm technology, extracted for parasitics with Calibre PEX, and simulated with Cadence Spectre. We implement the proposed performance-driven analog placement framework with Python 3.7.5 and C++. The main part of the analog placement and routing algorithms is implemented in C++ based on the MAGICAL framework [15]. The gradient extraction and feature generation parts are implemented with torch-1.13.1 [26]. All designs are in TSMC 40nm technology, extracted for parasitics with Calibre PEX, and simulated with Cadence Spectre. The implementation of PAPlace is available at https://github.com/shipxu123/PAPlace.

4.1 Experimental Setting

Benchmarks. Due to significant differences in performance metrics and layout design space of different analog functional building modules, we restrict our research scope to the design of **operational transconductance amplifiers** (**OTAs**). The statistics of our benchmarks are shown in Table 1. OTAs are classic analog circuit designs that have a wide range of applications. As one of the most classic analog circuits, the optimization of OTAs (Operational Transconductance Amplifier) post-layout performance is a typical problem, which can provide reference suggestions for other circuits. However, we have to admit that the performance model migration problem across different types of analog circuits remains unresolved. Especially for different types of circuits, the performance metrics they focus on may inherently vary.

We conducted experiments on four OTAs: OTA1, OTA2, OTA3, and OTA4. OTA1 and OTA2 are 2-stage Miller-compensated amplifier OTAs, as shown in Figure 8(a). The Miller-compensated amplifier OTA is an amplifier structure consisting of a differential input stage and a common-source output stage. It achieves stability and compensation through a Miller compensation capacitor, which improves the gain-bandwidth product. This structure has relatively high gain, moderate input capacitance, and reasonably wide bandwidth, making it suitable for applications requiring medium gain and bandwidth. The two designs consist of 6 PMOS transistors, 8 NMOS transistors, 2 capacitors, and a total of 25 modules that need to be placed. OTA3 and OTA4 are two 2-stage telescopic OTAs, and have a more complex topology, as depicted in Figure 8(b). The telescopic OTA is an amplifier structure that adopts a telescopic circuit topology. The first stage uses a telescopic

Table 1. Benchmark Circuits Information of the Number of PMOS, Number of NMOS, Number of Capacitors, Number of Resistors, Number of Standard Cells, Number of Total Modules

Benchmark	#PMOS	#NMOS	#Cap	#Res	#Total
OTA1	6	8	2	0	25
OTA2	6	8	2	0	25
OTA3	16	10	6	4	36
OTA4	16	10	6	4	36
COMP	8	9	0	0	17

differential pair, and the second uses a common-source amplifier. This structure has relatively low power consumption, but a lower gain than the Miller-compensated OTA. Compared with the Miller-compensated OTA, it has a smaller input capacitance and a simpler compensation circuit. The two designs comprise 16 PMOS transistors, 10 NMOS transistors, 6 capacitors, 4 resistors, and a total of 36 modules that need to be placed. To assess the generalization capability of our framework, we further introduced a comparator (COMP) design benchmark.

The selection of these circuit benchmarks allows us to explore the challenges of optimizing circuit layouts for circuits with different post-layout performance requirements. OTA1 and OTA2 have the same circuit topology but different sizing, and the same goes for OTA3 and OTA4. All generated layouts are LVS (**Layout Versus Schematic** (**LVS**) clean.

Compared Methods. We compare our PAPlace framework with the following baselines:

(1) MAGICAL$^+$: MAGICAL placement method [4] assisted by random Search [31];
(2) DAC'20: The same placement method equipped with multi-objective Bayesian optimization [11].

In those two approaches, the placement-related parameters, i.e., net weight parameters, are wrapped into random search and multi-objective Bayesian optimization frameworks.

Evaluation Metrics. The proposed approaches aim at boosting the ability to obtain analog circuit layouts with optimal performance. The well-known post-layout metrics (Offset Voltage, CMRR, DC Gain, Bandwidth, and Noise) and the runtime (i.e., wall-clock time) are used to measure performance and efficiency, respectively. To assess layout density, routability, and design-rule correctness, we report: final layout area, **half-perimeter wirelength** (**HPWL**), **routed wirelength** (**RWL**), via usage, and post-routing **design rule violations** (**DRV**). These metrics allow a direct comparison of our methods and other methods in terms of physical implementation quality. We set the runtime of PAPlace-S and PAPlace-F as the time they take to train/finetune the performance learning model and obtain the post-layout simulation results of the final placement solutions. The runtime of Magical$^+$ [4] and DAC'20 [11] is fixed to the time of 20 times initial sampling and 200 times trial-and-error processes. Unlike MAGICAL$^+$ and DAC'20, PAPlace samples and simulates in parallel, so we use 5 hosts to collect data simultaneously. We use 1,000 samples on the target design to train PAPlace-S and 20 samples to finetune PAPlace-F.

Simulation Settings. All post-layout simulations are performed using industry-standard tools. We use Calibre PEX with full R+C+CC extraction for accurate parasitic modeling. Performance metrics are evaluated in Cadence Spectre using the appropriate analyses for each metric: STB analysis for loop gain, DC loop gain, unity-gain bandwidth, and phase margin; AC analysis for CMRR; DC operating-point analysis for output DC levels and input-referred offset voltage; and noise analysis for noise measurement. These settings match standard industrial post-layout verification flows.

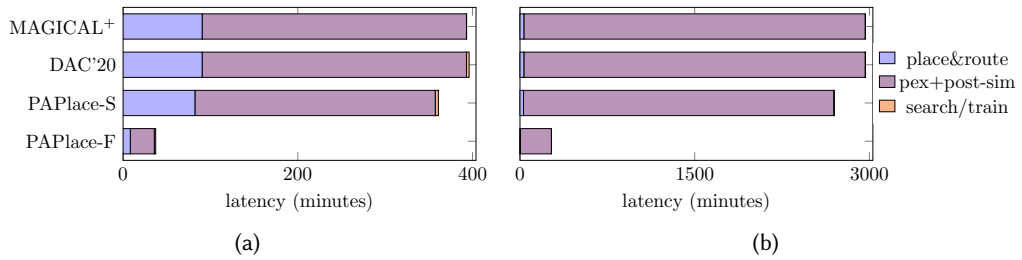


Fig. 9. The runtime breakdown of different methods: (a) OTA1; (b) OTA3.

Table 2. Placement Performance Comparison (The Best Results are in **Bold**)

Placement	Metrics	MAGICAL$^+$ [4]	DAC'20 [11]	Paplace-S	Paplace-F
OTA1	Area (μm^2)	2149.28	2039.04	1911.60	1991.04
	HPWL (μm)	443.37	434.65	436.00	424.45
	RWL (μm)	1112.80	1082.00	1441.20	1062.80
	VIA	38	40	37	38
	DRV	0	0	0	0
OTA2	Area (μm^2)	1982.08	2081.52	1896.96	2081.52
	HPWL (μm)	381.20	421.20	439.10	437.65
	RWL (μm)	1259.60	1331.60	1098.80	1533.60
	VIA	54	57	42	39
	DRV	0	0	0	0
OTA3	Area (μm^2)	1049.12	1056.16	1245.28	1398.72
	HPWL (μm)	433.00	435.68	453.25	435.68
	RWL (μm)	1061.60	1220.80	1168.00	1214.40
	VIA	71	79	60	70
	DRV	0	0	0	0
OTA4	Area (μm^2)	977.44	954.32	1075.84	1236.00
	HPWL (μm)	432.90	438.32	500.93	416.33
	RWL (μm)	1120.40	1036.00	1252.40	1088.40
	VIA	69	68	68	74
	DRV	0	0	0	0

The relatively low per-sample simulation time is enabled by running all spectral analyses in parallel across five compute hosts.

4.2 Experimental Results Comparison and Analysis

We first compare the layout qualities of our method compared with the cutting-edge methods. To verify the layout qualities, we perform post-layout simulations. We use Calibre PEX to extract parasitic resistance, parasitic capacitor, and coupling capacitance (R+C+CC). The performance is evaluated using Cadence Spectre. Then, we compare the runtime performance of different methods.

Layout Metrics Comparion. Table 2 compares standard layout metrics, including area, HPWL, routed wirelength, via count, and post-routing DRV, for all four OTA benchmarks. All reported layouts are fully legalized and DRC-clean (DRV = 0). For OTA1 and OTA2, PAPlace-S achieves the smallest or close-to-smallest area while keeping HPWL and RWL comparable to MAGICAL$^+$ and DAC'20. PAPlace-F attains similar wirelength and via counts to DAC'20. For OTA3 and OTA4,

Table 3. Post-Layout Performance Comparison

Circuits		Schematic	MAGICAL$^+$ [4]	DAC'20 [11]	PAPlace-S	PAPlace-F
OTA1	Offset Voltage(μV) ↓	-	1466.00	873.00	**507.00**	773.00
	CMRR(dB) ↑	155.30	89.08	79.06	**135.40**	91.60
	BandWidth(MHz) ↑	108.10	50.97	51.41	**52.15**	51.82
	DC Gain(dB) ↑	37.19	36.79	37.13	**37.28**	37.06
	Noise(μV_{rms}) ↓	213.40	225.00	225.00	**223.00**	224.00
OTA2	Offset Voltage(μV) ↓	-	11370.00	5750.00	**3587.00**	5679.00
	CMRR(dB) ↑	37.72	34.36	35.42	**44.94**	34.96
	BandWidth(MHz) ↑	87.54	36.56	39.05	**40.94**	39.12
	DC Gain(dB) ↑	34.26	30.58	32.48	**35.05**	32.56
	Noise(μV_{rms}) ↓	213.60	**265.00**	279.00	275.00	281.00
OTA3	Offset Voltage(μV) ↓	-	206.00	18.32	**13.46**	40.22
	CMRR(dB) ↑	126.60	46.96	47.25	**121.20**	117.60
	BandWidth(MHz) ↑	590.00	34.39	49.07	**54.40**	54.05
	DC Gain(dB) ↑	47.50	6.87	43.24	47.78	**47.83**
	Noise(μV_{rms}) ↓	3061.00	**1570.00**	2020.00	2249.00	2330.00
OTA4	Offset Voltage(μV) ↓	-	90.00	29.00	**7.00**	167.00
	CMRR(dB) ↑	110.50	74.50	89.78	85.75	**112.20**
	BandWidth(MHz) ↑	802.10	25.61	24.63	**86.88**	39.02
	DC Gain(dB) ↑	44.20	6.08	6.08	**63.92**	44.34
	Noise(μV_{rms}) ↓	3430.00	2560.00	2434.00	**877.00**	2518.00
COMP	Delay↓(ps)	103.00	409.80	291.00	**100.50**	108.10
	Power↓(μW)	13.45	21.53	21.53	**18.52**	19.52

PAPlace-S and PAPlace-F incur a modest area increase due to more performance-oriented device spreading, but HPWL, RWL, and via counts remain in the same range as the two baselines.

Post-layout Performance Comparison. Results in Table 3 and Figure 9 demonstrate that our proposed PAPlace-S obtains layout results that are superior to baseline methods in terms of post-layout performance, with a similar or less runtime. Our proposed PAPlace-F method achieves inferior post-layout performance compared with PAPlace-S, but with significantly less runtime, and outperforms all baseline methods, as shown in Figure 9 and Table 5. Specifically, we observe the following:

— In terms of post-layout performance, our PAPlace method significantly improves four metrics: Offset Voltage, CMRR, Bandwidth, and DC Gain, while maintaining similar performance in the Noise metric, shown in Table 3. PAPlace-S achieves up to 2163.00μV, 73.95dB, 62.25MHz, and 57.84dB improvement compared with a best baseline method (DAC'20). Similarly, PAPlace-F achieves up to 100.00μV, 70.35dB, 14.39MHz, and 38.26dB improvement in these four metrics compared with the DAC'20 method, demonstrating the effectiveness of our differentiable optimization approach.

— On average, PAPlace-S obtains 69.26%, 32.26%, 30.07%, 47.53%, 8.48% improvement compared with the MAGICAL$^+$ method on Offset Voltage, CMRR, BandWidth, and DC Gain, Noise, respectively. PAPlace-F obtains 32.92%, 23.77%, 14.86%, and 37.97% improvement compared with the MAGICAL$^+$ method on Offset Voltage, CMRR, BandWidth, and DC Gain, respectively. These experiments validate the effectiveness of our PAPlace method, demonstrating its capability to enhance placement qualities on multiple post-layout metrics. We further visualize the placement results by different methods in Figure 11.

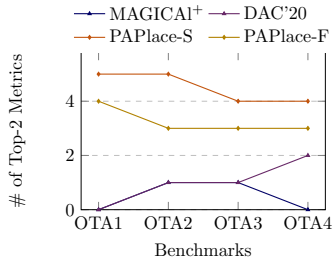


Fig. 10. The number of top-2 metrics for different methods.

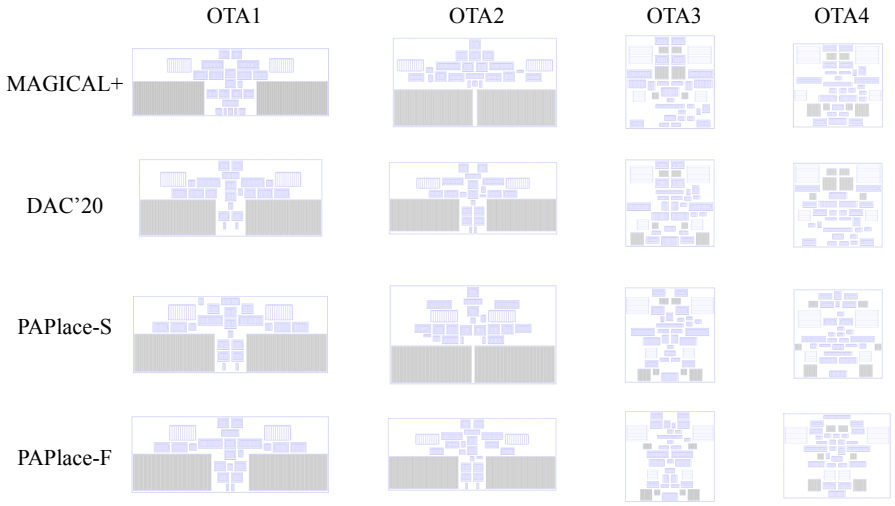


Fig. 11. The visualization of placement results by different methods.

— In contrast, MAGICAL$^+$ and DAC'20 methods utilize random sampling for estimation. This results in unreliable search quality due to the complexity of the analog placement. For example, the performance of these methods is far below the results obtained by the PAPlace methods in terms of the Bandwidth and Gain metrics in OTA4. This indicates the unreliability of those methods under a limited budget.

Evaluation on Comparator Design. The generalization capability of our framework on the COMP design is detailed in Table 3 and Figure 12(a). While MAGICAL$^+$ and DAC'20 exhibit higher normalized delay and power, our approaches (PAPlace-S and PAPlace-F) consistently improve delay while keeping power competitive. Notably, PAPlace-S achieves the lowest delay with reduced power compared with both baselines, demonstrating the efficiency of the proposed framework on a non-OTA design.

Analysis on multi-objective learning. As shown in Figure 10, our PAPlace method outperforms all baseline methods in terms of the number of top-2 metrics achieved for the obtained layout. PAPlace-S and PAPlace-F achieve top-2 performance in almost all metrics in Offset Voltage, CMRR, Bandwidth, and DC Gain. The results confirm that our multi-objective optimization method yields conflict-free optimal layout solutions.

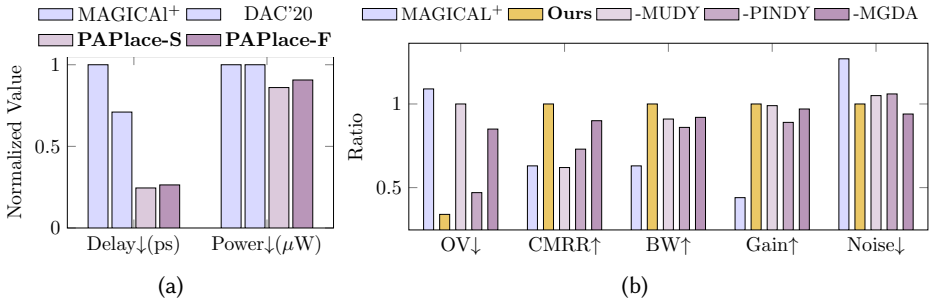


Fig. 12. (a) Normalized delay and power comparison on the COMP benchmark; (b) Ablation of MUDY, PINDY, and MGDA, with the MAGICAL$^+$ (NLP without ML) baseline, across OTA benchmarks.

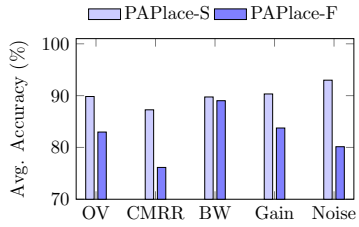


Fig. 13. Visualization of prediction performance comparison for PAPlace-S and PAPlace-F on different metrics.

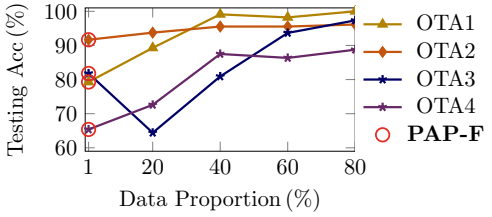


Fig. 14. The visualization of testing accuracy across different proportions of training samples on offset voltage. The red circles (PAP-F) highlight the performance achieved using only 1% of the training data with transfer learning.

4.3 Ablation Study

Ablation Analysis of Different Modules. Figure 12(b) details our ablation study, comparing the full PAPlace framework against the MAGICAL$^+$ baseline (NLP placement without ML) and three variants removing specific components: −MUDY, −PINDY, and −MGDA. The results show progressive improvements from the baseline to the full framework. First, introducing the ML model significantly boosts Gain and Bandwidth compared with MAGICAL$^+$, validating the benefit of performance prediction over vanilla NLP placement. Second, specific modules prove critical for distinct metrics: removing MUDY drastically degrades Offset Voltage, while removing PINDY primarily harms CMRR. Third, omitting MGDA disrupts the multi-objective balance, leading to suboptimal trade-offs compared with the full framework. Ultimately, the complete PAPlace integration achieves optimal performance across all metrics, confirming that MUDY, PINDY, and MGDA are all essential for maximizing post-layout circuit performance.

Analysis on Accuracy for Transfer Learning. We evaluated the accuracy of PAPlace-S and PAPlace-F on the same test set for each design to compare the two proposed methods, as shown in Table 4 and Figure 13. The results show that PAPlace-S achieves reliable performance prediction accuracy for all designs. Overall, the accuracy of the transferred models is reliable. Compared with training from scratch, which requires more data collection time, the accuracy reduction ranges from 3% to 22%. However, there are still cases where the model performance deviates. This is because the transferability of the models varies under different designs and metrics. Transfer between different

Table 4. Placement Prediction Accuracy with Collected Data for PAPlace-S and Transfer Learning Results for PAPlace-F

Design	Prediction Accuracy	PAPlace-S	PAPlace-F
OTA1	Offset Voltage(%)	**99.11**	79.29
	CMRR(%)	**88.59**	70.24
	BandWidth(%)	**98.21**	90.18
	DC Gain(%)	**94.64**	90.18
	Noise(%)	**95.31**	76.04
OTA2	Offset Voltage(%)	**95.54**	91.67
	CMRR(%)	**91.96**	77.68
	BandWidth(%z)	**96.43**	95.54
	DC Gain(%)	**93.62**	88.01
	Noise(%)	**91.96**	79.14
OTA3	Offset Voltage(%)	**83.33**	81.79
	CMRR(%)	**86.17**	68.95
	BandWidth(%)	**83.59**	80.36
	DC Gain(%)	**91.67**	85.71
	Noise(%)	**95.83**	68.49
OTA4	Offset Voltage(%)	**81.35**	65.39
	CMRR(%)	**82.33**	62.02
	BandWidth(%)	**80.71**	72.14
	DC Gain(%)	**81.35**	59.50
	Noise(%)	**88.80**	69.29

topologies and sizing settings results in different performance distributions in the layout, which can cause the transfer to fail. The limited number of sample data for finetuning makes it difficult for the PAPlace-F to fit the current design's overall data distribution. Despite this, PAPlace-F still provides relatively reliable performance estimation. The performance improvement of PAPlace-F, compared with the DAC'20 method, demonstrates the robustness of performance-based gradient optimization methods.

Analysis on Runtime Comparison. In terms of accumulated runtime, our proposed PAPlace method has a similar or less runtime than the compared methods, as shown in Table 5 and Figure 9. Specifically, PAPlace-S requires 91.06% of the time compared with the DAC'20 method, resulting in significant performance improvement and a noticeable reduction in runtime. Due to requiring a much smaller amount of data, PAPlace-F only takes 9.28% of the time compared with the DAC'19 method, leading to enhanced performance and a substantial reduction in runtime. Even though the inference time of our proposed approach is slower than the plain placement process, our method effectively reduces the accumulated runtime by exploiting the sampled data.

Analysis on Training Samples. To justify the choice of sample size for fine-tuning, we quantitatively evaluated how prediction accuracy varies with the proportion of training data. Figure 14 illustrates the testing accuracy as the data proportion increases from 1% to 80%. The results indicate a general positive correlation between sample count and accuracy; however, they also reveal that the marginal utility of additional data diminishes after the initial few samples. The red circles in Figure 14 mark the performance of **PAP-F** using our selected budget of approximately 20 samples corresponding to the 1% data point. Evidence of sufficiency is observed in the high accuracy achieved at this low-data regime: PAP-F reaches over 90% accuracy for OTA2 and approximately

Table 5. The Runtime Comparison of Different Methods

Design	Time (m)	MAGICAL$^+$ [4]	DAC'20 [11]	PAPlace-S	PAPlace-F
OTA1	P&R	90.47	90.47	83.16	9.15
	PEX+Sim	302.60	302.60	275.09	27.51
	Search/Train	0.21	3.04	3.58	1.58
	Total	393.28	396.11	361.84	38.24
OTA2	P&R	79.91	79.91	75.15	9.77
	PEX+Sim	305.61	305.61	277.83	27.78
	Search/Train	0.21	3.04	3.58	1.58
	Total	385.74	388.56	356.56	39.13
OTA3	P&R	33.42	33.42	30.66	3.32
	PEX+Sim	2929.84	2929.84	2663.49	266.35
	Search/Train	0.22	3.49	3.83	1.33
	Total	2963.48	2966.75	2697.98	271.00
OTA4	P&R	71.11	71.11	64.90	6.72
	PEX+Sim	738.07	738.07	670.97	67.10
	Search/Train	0.22	3.49	3.83	1.33
	Total	809.39	812.66	739.70	75.15
ratio		0.997	1.000	**0.911**	**0.093**

80% for OTA1 and OTA3 using only 1% of the data. While OTA4 is more challenging, the method still secures reasonable generalization. This confirms that 20 samples represent an optimal practical trade-off, providing sufficient information for the model to generalize effectively while keeping the data collection cost minimal.

5 Conclusion and Future Directions

In this work, we presented Performance-driven analog placement (PAPlace), a novel framework that integrates recent machine learning advances with the state-of-the-art analog placer to optimize for post-layout performance. Our framework combines a differentiable performance model, a gradient-descent-based global placement, and a multi-objective optimization method to optimize multiple post-layout metrics effectively. Experimental results demonstrate that PAPlace outperforms conventional methods in multiple post-layout metrics, with up to 2163.00μV, 73.95dB, 62.25MHz, 57.84dB improvement in Offset Voltage, CMRR, BandWidth, and DC Gain metrics.

Our proposed framework offers a promising method for optimizing analog circuit placement for post-layout performance while reducing the required time and expertise. Besides, several promising directions can be considered. One important direction is to migrate the performance models across different types of analog circuits, thus we can improve the generalizability of the PAPlace methodology to a wider range of circuit types. Another potential improvement of PAPlace is to incorporate the routing knowledge into the early stage of the analog placement process to improve the routability. Additionally, incorporating simulation-in-the-loop feedback during optimization, rather than using offline-trained models, could further improve optimization quality, though it would increase computational cost.

References

[1] Hao Chen, Mingjie Liu, Xiyuan Tang, Keren Zhu, Nan Sun, and David Z. Pan. 1993. Challenges and opportunities toward fully automated analog layout design. *The Journal of Supercomputing* 41, 11 (1993), 1674–4926.

[2] Qiang Ma, Evangeline F. Y. Young, and K. P. Pun. 2007. Analog placement with common centroid constraints. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 579–585.

[3] U. Choudhury and A. Sangiovanni-Vincentelli. 1995. A performance-driven placement tool for analog integrated circuits. *IEEE Journal Solid-State Circuits* 30, 7 (1995), 773–780.
[4] Keren Zhu, Hao Chen, Mingjie Liu, Xiyuan Tang, Nan Sun, and David Z. Pan. 2020. Effective analog/mixed-signal circuit placement considering system signal flow. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1–9.
[5] Abhishek Patyal, Hung-Ming Chen, Mark Po-Hung Lin, Guan-Qi Fang, and Simon Yi-Hung Chen. 2022. Pole-aware analog layout synthesis considering monotonic current flows and wire crossings. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 42, 1 (2022), 266–279.
[6] Wenlong Lyu, Fan Yang, Changhao Yan, Dian Zhou, and Xuan Zeng. 2018. Multi-objective bayesian optimization for analog/RF circuit synthesis. In *ACM/IEEE Design Automation Conference (DAC)*. 1–6.
[7] Yaguang Li, Yishuang Lin, Meghna Madhusudan, Arvind Sharma, Wenbin Xu, Sachin S. Sapatnekar, Ramesh Harjani, and Jiang Hu. 2020. A customized graph neural network model for guiding analog IC placement. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1–9.
[8] Yaguang Li, Yishuang Lin, Meghna Madhusudan, Arvind Sharma, Sachin Sapatnekar, Ramesh Harjani, and Jiang Hu. 2023. Performance-driven wire sizing for analog integrated circuits. *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 28, 2 (2023), 1–23.
[9] Mingjie Liu, Keren Zhu, Jiaqi Gu, Linxiao Shen, Xiyuan Tang, Nan Sun, and David Z. Pan. 2020. Towards decrypting the art of analog layout: Placement quality prediction via transfer learning. In *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*. 496–501.
[10] Chen-Chia Chang, Jingyu Pan, Zhiyao Xie, Yaguang Li, Yishuang Lin, Jiang Hu, and Yiran Chen. 2023. Fully automated machine learning model development for analog placement quality prediction. In *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*. 58–63.
[11] Mingjie Liu, Keren Zhu, Jiaqi Gu, Linxiao Shen, Xiyuan Tang, Nan Sun, and David Z. Pan. 2020. Closing the design loop: Bayesian optimization assisted hierarchical analog layout synthesis. In *ACM/IEEE Design Automation Conference (DAC)*. 496–501.
[12] Hung-Chih Ou, Kai-Han Tseng, Jhao-Yan Liu, I.-Peng Wu, and Yao-Wen Chang. 2015. Layout-dependent-effects-aware analytical analog placement. In *ACM/IEEE Design Automation Conference (DAC)*. 1–6.
[13] Biying Xu, Shaolan Li, Chak-Wa Pui, Derong Liu, Linxiao Shen, Yibo Lin, Nan Sun, and David Z. Pan. 2019. Device layer-aware analytical placement for analog circuits. In *ACM International Symposium on Physical Design (ISPD)*. 19–26.
[14] Yishuang Lin, Yaguang Li, Donghao Fang, Meghna Madhusudan, Sachin S. Sapatnekar, Ramesh Harjani, and Jiang Hu. 2022. Are analytical techniques worthwhile for analog IC placement?. In *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*. 154–159.
[15] Hao Chen, Mingjie Liu, Xiyuan Tang, Keren Zhu, Abhishek Mukherjee, Nan Sun, and David Z. Pan. 2021. MAGICAL 1.0: An open-source fully-automated AMS layout synthesis framework verified with a 40-nm 1GS/s $\Delta\sum$ ADC. In *IEEE Custom Integrated Circuits Conference (CICC)*. 1–2.
[16] Syota Kuwabara, Yukihide Kohira, and Yasuhiro Takashima. 2013. An effective overlap removable objective for analytical placement. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 96, 6 (2013), 1348–1356.
[17] Marcel J. M. Pelgrom, Aad C. J. Duinmaijer, and Anton P. G. Welbers. 1989. Matching properties of MOS transistors. *IEEE Journal Solid-State Circuits* 24, 5 (1989), 1433–1439.
[18] Koen Lampaert, Georges Gielen, and Willy M. Sansen. 1995. A performance-driven placement tool for analog integrated circuits. *IEEE Journal Solid-State Circuits* 30, 7 (1995), 773–780.
[19] Gaetano Palumbo and Giuseppe Scotti. 2021. A novel standard-cell-based implementation of the digital OTA suitable for automatic place and route. *Journal of Low Power Electronics and Applications* 11, 4 (2021), 42.
[20] Mark Po-Hung Lin, Hao-Yu Chi, Abhishek Patyal, Zheng-Yao Liu, Jun-Jie Zhao, Chien-Nan Jimmy Liu, and Hung-Ming Chen. 2020. Achieving analog layout integrity through learning and migration. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1–8.
[21] Pedro Toledo, Paolo Crovetti, Orazio Aiello, and Massimo Alioto. 2021. Design of digital OTAs with operation down to 0.3 V and nW power for direct harvesting. *IEEE Transactions on Circuits and Systems I: Regular Papers* 68, 9 (2021), 3693–3706.
[22] Kalyanmoy Deb, Karthik Sindhya, and Jussi Hakanen. 2016. Multi-objective optimization. In *Decision Sciences*. CRC Press, 161–200.
[23] Siting Liu, Qi Sun, Peiyu Liao, Yibo Lin, and Bei Yu. 2021. Global placement with deep learning-enabled explicit routability optimization. In *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*.
[24] Meng-Kai Hsu, Yi-Fang Chen, Chau-Chin Huang, Sheng Chou, Tzu-Hen Lin, Tung-Chieh Chen, and Yao-Wen Chang. 2014. NTUplace4h: A novel routability-driven placement algorithm for hierarchical mixed-size circuit designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)* 33, 12 (2014), 1914–1927.

[25] Peter Spindler and Frank M. Johannes. 2007. Fast and accurate routing demand estimation for efficient routability-driven placement. In *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*.

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems* 32 (2019), 8024–8035.

[27] Adem Kilicman and Zeyad Abdel Aziz Al Zhour. 2007. Kronecker operational matrices for fractional calculus and some applications. *Appl. Math. Comput.* 187, 1 (2007), 250–265.

[28] Jean-Antoine Désidéri. 2012. Multiple-gradient descent algorithm (MGDA) for multiobjective optimization. *Comptes Rendus Mathematique* 350, 5-6 (2012), 313–318.

[29] Ozan Sener and Vladlen Koltun. 2018. Multi-task learning as multi-objective optimization. *Annual Conference on Neural Information Processing Systems (NIPS)* 31 (2018).

[30] Martin Jaggi. 2013. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *International Conference on Machine Learning (ICML)*. 427–435.

[31] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research* 13, 10 (2012), 281–305.

Received 19 February 2025; revised 5 February 2026; accepted 14 February 2026