EVOPCB: EVOLVING PCB PHYSICAL DESIGN TO EXPERT ALIGNMENT VIA POLICY OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Despite substantial progress in automated printed circuit board (PCB) physical design, mainly including placement and routing (PnR), industrial physical design still relies heavily on human expert engineers. Existing frameworks can generate design-rule-compliant designs, but conventional objectives alone do not ensure satisfactory performance or compliance with industrial standards. Moreover, existing frameworks often cannot adapt to user specifications or domain-specific constraints, which also limits their applicability. To handle these problems, we propose EvoPCB[1], an end-to-end, LLM-based PCB physical design framework. EvoPCB introduces RePCB, a performance-aware supervised representation model that embeds PnR topology and measures similarity to expert layouts. A two-stage specialization, supervised fine-tuning (SFT) followed by customized decoupled clip and dynamic sampling policy optimization (DAPO), assists EvoPCB in generating high-quality initial designs. EvoPCB then refines the layouts through a multi-agent and multi-objective script evolution method, MM-Evolve, which coordinates geometric and multi-physics optimization. Experimental results demonstrate that EvoPCB achieves the best overall performance compared to existing state-of-the-art methods across all evaluated metrics.

1 INTRODUCTION

Printed circuit boards (PCBs) play a critical role in nearly all electronic systems. As electronic products become more capable and complex, modern PCBs contain increasing numbers of components, pins, and nets. Designing these PCBs requires simultaneous consideration of thermal and electromagnetic (EM) performance (Wu et al., 2010; Bogatin, 2010; Mircea, 2012), physical design rules, and user specifications. Consequently, industrial PCB physical design still relies heavily on human experts. Automatic PCB placement and routing (PnR) is quite challenging because of the large, heterogeneous solution space and the lack of reliable early-stage evaluation. Components may be rotated and placed on either side of the board, while board outlines and component shapes can be irregular. Moreover, inner routing layers further enlarge the solution space. Meanwhile, conventional placement metrics such as half-perimeter wirelength (HPWL) cannot fully predict downstream routability or multi-physics performance, so early evaluation of PCB PnR is usually invalid.

To address these challenges, various frameworks have been developed for automated PCB PnR design (Chen et al., 2023; Cheng et al., 2022; Tsou et al., 2024; Li et al., 2026; Chen et al., 2025; Han et al., 2026a;b), typically integrating multiple design constraints with optimization objectives. OpenROAD (OpenROAD, 2019a) open-sourced PCB-PR-App (OpenROAD, 2019b), composed of SA-PCB (OpenROAD, 2019d) and PcbRouter (OpenROAD, 2019c), to solve the PCB PnR problem. NS-Place (Cheng et al., 2022) minimized net congestion using a support vector machine-like formulation and performed legalization by solving a routability-aware Mixed-Integer Linear Programming (MILP). HiePlace (Li et al., 2026) is a heuristic algorithm to optimize the placement, thereby reducing manufacturing complexity. PCB-Migrator (Han et al., 2026a) is a PCB PnR migration framework, but it needs existing expert results as references, which limits its broader applications. All these methods rely solely on traditional metrics to optimize PnR, as shown in Figure 1, which is far from expert-level performance and not flexible enough to handle varying user requirements.

[1]Anonymous Repository: https://anonymous.4open.science/r/EvoPCB-ICLR-2027-8E0E/. All EvoPCB code will be open-sourced after acceptance.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

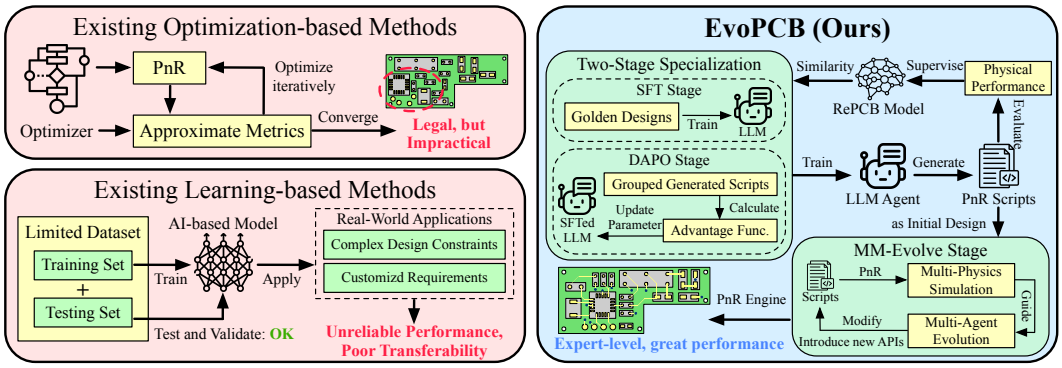


Figure 1: Comparison of existing optimization- and learning-based PCB physical design frameworks with the proposed EvoPCB.

Through data-driven approaches, several studies incorporate historical experience into AI models to enhance PCB PnR design. DeepPCB (InstaDeep, 2026) is a commercial tool that implements PCB PnR using reinforcement learning, aiming to reduce wirelength and via count in industrial PCBs. PCBAgent (Chen et al., 2025) is a framework that combines LLM and RL agents to enhance PCB placement by considering constraints from multiple masks (Lai et al., 2022). Although this method may achieve low HPWL, its placement may be unroutable and cannot be utilized in actual applications. Meanwhile, it incurs high training costs and is extremely difficult to converge. ModuPlace (Han et al., 2026b) proposes training an LLM to learn design experience from expert designs and use a modular design paradigm to perform PCB placement. However, its generalizability is significantly poor. TRouter (Chen et al., 2023) is a thermal-driven PCB routing framework via a machine learning-based surrogate model. Most of the above-mentioned frameworks still focus on conventional metrics (e.g., wirelength, via count) without considering actual performance and usability. Meanwhile, existing AI-based methods rely excessively on training data and models, resulting in poor generalization and reliability, as shown in Figure 1. Therefore, few of them have been adopted as end-to-end solutions in real-world industrial flows.

Recent advances in LLMs provide new opportunities for high-quality PCB physical design automation. Unlike conventional analytical formulations, LLMs can learn design guidelines, user requirements, and aspects of expert design practice that are difficult to encode as analytical objectives, through supervised fine-tuning (SFT) and dynamic sampling policy optimization (DAPO) (Yu et al., 2026; Hsiao et al., 2025; Han et al., 2026b). In this way, LLMs can generate expert-like layouts while remaining flexible to varying user requirements. Furthermore, agents can inspect generated layouts and propose iterative evolutions based on user requirements and multi-physics simulation feedback (Uchendu et al., 2026). Building on these capabilities, we propose **EvoPCB**, an LLM-based end-to-end PCB physical design framework that optimizes both conventional physical metrics and multi-physics performance, as shown in Figure 1.

Overall, our main contributions are as follows:

1. We propose EvoPCB, an LLM-based end-to-end framework to perform expert-level PCB physical design, which leverages the powerful generative capabilities of LLM to design layouts like human experts. Experiments demonstrate that EvoPCB outperforms all state-of-the-art PCB PnR baselines.
2. EvoPCB introduces RePCB, a performance-aware representation learning method specialized for PCB physical design, to capture both topological features and performance, enabling automated early evaluation of PCB layout quality compared to expert results.
3. To bridge the gap between LLM agents and human engineers in PCB physical design, EvoPCB implements several PnR APIs and utilizes a two-stage specialization, including customized SFT and DAPO, for high-quality PCB PnR script generation.
4. We propose MM-Evolve, which coordinates specialist agents to refine generated initial layouts using visual analysis and multi-physics simulation feedback, thereby co-optimizing wirelength, via usage, EM, and thermal performance.

2 PROBLEM FORMULATION

Inputs A PCB netlist $\mathcal{N}(C, E)$ containing a set of components C and their connectivity E, together with design guidelines, and physical constraints.

Output A generated PCB layout $L_s = (P, R)$, where P specifies the physical placement of all components and R specifies the routing paths that connect their pins.

Constraints The layout must strictly satisfy two sets of design rules:
1) **Placement Rules**: All components must be placed within the board boundary without overlap. The orientation of each component is restricted to $\theta_i \in \{0°, 90°, 180°, 270°\},\ \forall i \in C$.
2) **Routing Rules**: All pins belonging to the same net must be electrically connected, and no conductive path may short two distinct nets. The geometry of wires and vias must satisfy physical design rules. Thus, the final layout must be both DRC-clean and LVS-clean.

Goal Our objective is to establish an effective mapping scheme to map a PCB layout L that satisfies all constraints and minimizes the following cost function:

$$\mathbb{C}(L) = \mu_1 \cdot \mathrm{WL} + \mu_2 \cdot \mathrm{Via} + \mu_3 \cdot \mathcal{S}_{\mathrm{EM}} + \mu_4 \cdot \mathcal{S}_{\mathrm{Th}}, \tag{1}$$

where WL and Via are the wirelength and via count, while $\mathcal{S}_{\mathrm{EM}}$ and $\mathcal{S}_{\mathrm{Th}}$ are the dimensionless EM and thermal costs defined in Section E.3. All four terms are direction-aligned so that lower values indicate better performance. The hyperparameter set μ encodes the relative importance of the four objectives.

3 METHODOLOGY

3.1 PNR-RELATED API IMPLEMENTATION

For LLMs, directly generating PCB physical designs is challenging for two main reasons. First, modern PCB designs usually contain many more components and nets, which means the connections between components are more complicated, and the solution space of PnR is enormous. Second, existing LLMs struggle to fully understand the geometrical relationships between components and wires in structured language descriptions. Therefore, it is unrealistic to expect LLMs to directly output usable PnR designs, as this would lead to serious hallucination problems and illegal results (Lu et al., 2026a;b). However, leveraging LLMs to invoke existing algorithms and customize them for different designs is a promising direction (Zang et al., 2025; Ren et al., 2026; Chang et al., 2025). On the one hand, this significantly constrains the solution space while preserving the majority of legal solutions. On the other hand, LLMs can comprehend the design from a global perspective and yield superior results by modifying API calls and parameters (Ren et al., 2026). In EvoPCB, we encapsulate several PCB PnR algorithms as APIs the LLM agent can call, so PCB physical design can be well formulated as a script-generation task, as shown in Table 3 and Section B.

3.2 PERFORMANCE-AWARE SUPERVISED LAYOUT REPRESENTATION

To compare the PnR topology of LLM-generated PCB layouts with expert designs, we propose **RePCB**, a supervised representation model that jointly encodes PnR topology and physical performance. As illustrated in Figure 2, RePCB contains a GraphSAGE-based branch for placement and a CNN branch for routing. Extracted PnR features are then processed by an attention-based fusion model supervised by performance.

Placement Representation. We propose the Comprehensive Placement Graph (CPG) $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to represent PCB placement, where $\mathcal{V} = \mathcal{V}_c \cup \mathcal{V}_n$ contains component and net vertices. The graph has two edge types: a component-net edge connects a component to each net incident to its pins. In contrast, a component-component edge connects spatially adjacent components selected by Z-nearest neighbors. Each edge (u, v) is assigned by $\mathcal{A}_{(u,v)} = \big(\Delta x/W, \Delta y/H, d/\mathcal{D}, t\big)$, where W, H, and $\mathcal{D}$ are the board width, height, and diagonal, d is the endpoint distance, and t identifies the edge type. These attributes encode scale-normalized relative geometry while distinguishing physical adjacency from electrical connectivity. Component and net features, denoted by $\mathcal{F}_c$ and $\mathcal{F}_n$, are summarized in Table 4. The two vertex types are first projected into a shared latent space and then processed by the edge-conditioned GraphSAGE-based network (Hamilton et al., 2017)

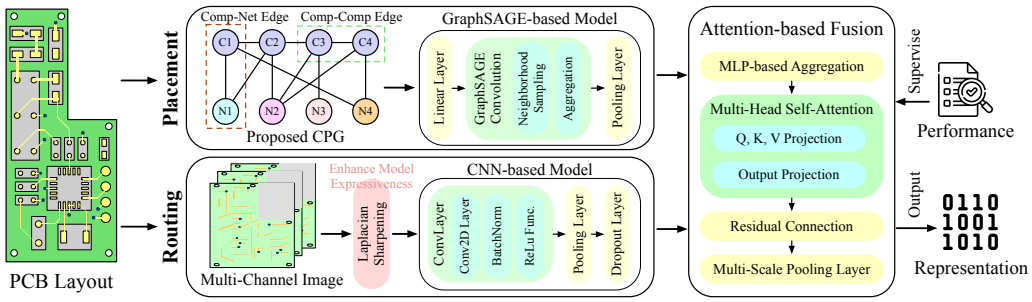


Figure 2: Overview of RePCB, the performance-aware supervised PCB layout representation model.

in Algorithm 2. At each layer, messages are computed from the source embedding and its edge attribute, followed by mean aggregation, layer normalization, dropout, and a residual connection. Mean and max pooling over all component and net vertices produce the placement feature $\mathbf{P}$.

Routing Representation. To represent detailed PCB routing geometry, we rasterize each routed layout into a six-channel image. As listed in Table 5, the channels encode pin occupancy, local net multiplicity, wire density, and via density. Count-based channels use logarithmic scaling to retain sensitivity in sparse regions without being dominated by dense regions, as described in Section C. Since the values in the routing images do not change continuously, RePCB applies channel-wise Laplacian sharpening to emphasize boundaries in the routing images:

$$g(x, y) = f(x, y) - \eta \cdot \nabla^2 f(x, y), \tag{2}$$

where $g(x, y)$ and $f(x, y)$ represent the sharpened and original images, respectively. η is the sharpening coefficient. $\nabla^2 f(x, y)$ is the Laplacian operator, defined in Equation (3):

$$\nabla^2 f(x, y) = f(x + 1, y) + f(x - 1, y) + f(x, y + 1) + f(x, y - 1) - 4f(x, y) \tag{3}$$

The sharpened routing image is then encoded by a CNN-based model with downsampling and global average pooling, yielding the routing feature $\mathbf{R}$. The PnR features are then fused by an MLP and multi-head self-attention (MHSA) as $\mathbf{u} = \mathrm{MLP}([\mathbf{P}\|\mathbf{R}])$, $\mathbf{v} = \mathrm{MHSA}(\mathbf{u}) + \mathbf{u}$, and $e = \mathrm{Normalize}_2(\mathrm{MPool}(\mathbf{v}))$. MPool represents multi-scale pooling, including max and average pooling, enabling RePCB to simultaneously capture PnR information at different scales. Before performance-aware supervision, this embedding e captures the topological structure but does not yet provide a reliable measure of performance.

Performance-aware Supervision. Topological similarity alone does not fully reflect the design quality and performance of layouts. Meanwhile, geometrically different layouts may exhibit comparable physical performance. We therefore supervise RePCB using wirelength w_i and via count v_i. For layout i, the target embedding $\mathbf{z}_i$ is obtained by standardizing $[\log(1 + w_i), \log(1 + v_i)]$ to reduce the impact of extreme values. The regression and similarity losses are defined as follows:

$$\mathcal{L}_{\mathrm{reg}} = \sum_{i \in \mathcal{P}} \mathrm{SmoothL1}\left(\hat{\mathbf{z}}_i, \mathbf{z}_i\right), \quad \mathcal{L}_{\mathrm{sim}} = \sum_{i,j \in \mathcal{P}} \left(\mathrm{sim}(e_i, e_j) - \exp(-\frac{\|\mathbf{z}_i - \mathbf{z}_j\|_2}{T})\right)^2, \tag{4}$$

where $\mathcal{P}$ denotes the training set and T is a temperature hyperparameter controlling how rapidly target similarity decreases with performance distance. The regression loss $\mathcal{L}_{\mathrm{reg}}$ maps the layout embedding to the prediction $\hat{\mathbf{z}}_i$, providing pointwise supervision. The similarity loss $\mathcal{L}_{\mathrm{sim}}$ term aligns embedding similarity with performance similarity, providing pairwise metric supervision. The final objective is $\mathcal{L} = \lambda_{\mathrm{reg}}\mathcal{L}_{\mathrm{reg}} + \lambda_{\mathrm{sim}}\mathcal{L}_{\mathrm{sim}}$. Consequently, the learned embedding captures both PnR topology and performance, providing valid similarity rewards for the subsequent LLM specialization stage.

3.3 LLM Specialization for Initial Layout Generation

In PCB physical design, a good initial design is the foundation for strong performance and usability. Human engineers typically make fine adjustments to the initial design to gradually improve overall

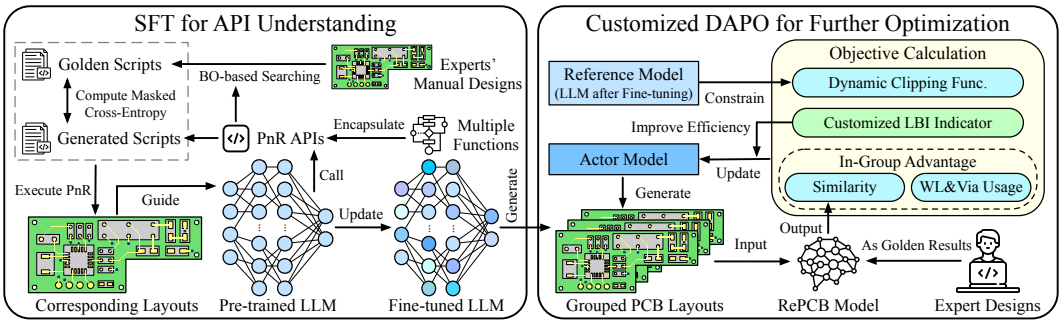


Figure 3: The two-stage model specialization for EvoPCB, including customized SFT and DAPO.

performance, including wirelength, via count, EM and thermal performance, etc. Therefore, the quality of the initial design results is extremely crucial to PCB physical design. As shown in Table 3, because multiple PnR algorithms are implemented as APIs, the LLM agent can generate scripts to complete the entire PCB PnR flow. In the initial PCB layout design process, EvoPCB aims to initially establish the script-layout mapping relationship, providing a solid foundation for further optimization. However, pre-trained LLMs lack a deep understanding of the specific design process of these algorithms, and real-world cases are usually complex and varied, so zero-shot results are not sufficient enough. Therefore, EvoPCB proposes a two-stage specialization, including SFT and DAPO, to enhance script generation for PCB PnR design.

SFT for PnR API understanding. First, we perform SFT to build an initial understanding of several PnR APIs for the agent, including API call order, applicable scenarios, and parameter settings. In this stage, we design a code template for EvoPCB to generate valid initial layouts in a one-shot generation. Let $\mathcal{D}_{\text{SFT}} = \{(q_i, s_i^g)\}_{i=1}^N$ denote the expert-designed PCB layout dataset, where q_i is a query and s_i^g is its corresponding reference script. We update the LLM parameters θ through teacher forcing with the token-level cross-entropy loss:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\frac{1}{\sum_{t=1}^{|s_i^g|} m_{i,t}}\sum_{t=1}^{|s_i^g|} m_{i,t}\log \pi_\theta\left(s_{i,t}^g \mid q_i, s_{i,<t}^g\right), \tag{5}$$

where $s_{i,t}^g$ is the t-th target token and $m_{i,t} \in \{0,1\}$ is the customized script mask. The mask excludes prompt and padding tokens from gradient computation while retaining the API names, call order, and parameters that determine the generated script. Minimizing this cross-entropy objective directly updates the LLM to maximize the likelihood of expert API-call sequences.

Customized DAPO for further optimization. After SFT, the model has acquired a basic understanding of the implemented PnR APIs. However, relying solely on SFT to align EvoPCB with expert scripts may cause the model to overfit the training data, limiting its ability to generalize to diverse PCB design scenarios. To address this limitation, we propose a customized DAPO method to further improve the quality of the generated layouts.

For each query q, the LLM agent generates a group of scripts $S = \{s_1, s_2, \ldots, s_G\}$, where G is the group size and each script produces a corresponding PCB layout. Let $s_i = (s_{i,1}, \ldots, s_{i,|s_i|})$ denote the token sequence of the i-th script, where $s_{i,t}$ is its t-th token and $s_{i,<t}$ denotes the preceding tokens. EvoPCB evaluates the scripts using physical metrics and their RePCB similarity to the corresponding reference layout. The customized DAPO objective is as follows:

$$\begin{aligned}\mathcal{J}(\theta) = \mathbb{E}_{q\sim Q,\,\{s_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)}\left[\frac{1}{M}\sum_{i=1}^{G}\sum_{t=1}^{|s_i|}\text{LBI}(s_i,s^g)m_{i,t}\cdot\min\left(I_{i,t}(\theta)\hat{A}_i, \text{clip}\left(I_{i,t}(\theta), 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}}\right)\hat{A}_i\right)\right],\\ \text{s.t.}\ \sum_{i=1}^{G}\text{LBI}(s_i,s^g) > 0, \quad M > 0,\end{aligned} \tag{6}$$

where $\text{clip}(\cdot)$ restricts the probability ratio to the specified range, while ϵ_{low} and ϵ_{high} determine its asymmetric lower and upper clipping ranges. $M = \sum_{i=1}^{G}\sum_{t=1}^{|s_i|}\text{LBI}(s_i,s^g)m_{i,t}$ counts the unmasked tokens across all scripts that are both legal and not identical to the reference. For token

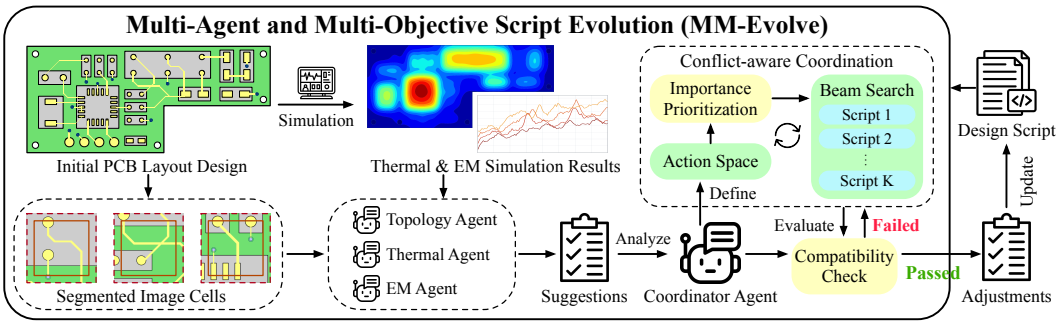


Figure 4: The pipeline of the proposed MM-Evolve, progressively improving the final PnR scripts within a unified evolutionary loop.

$s_{i,t}$, the token-level probability ratio $I_{i,t}(\theta)$ between the current policy π_θ and the old policy $\pi_{\theta_{\text{old}}}$ is as follows:

$$I_{i,t}(\theta) = \frac{\pi_\theta(s_{i,t} \mid q, s_{i,<t})}{\pi_{\theta_{\text{old}}}(s_{i,t} \mid q, s_{i,<t})} \tag{7}$$

As for the in-group relative advantage $\hat{A}_i$ of script s_i, it combines wirelength w_i, via count v_i, and RePCB similarity of the corresponding reference layout:

$$\hat{A}_i = \alpha \cdot \frac{\bar{w} - w_i}{\sigma_w + \delta} + \beta \cdot \frac{\bar{v} - v_i}{\sigma_v + \delta} + \gamma \cdot \text{sim}(e_i, e_i^g), \tag{8}$$

where $\bar{w}$ and $\bar{v}$ are the in-group means of wirelength and via count, respectively, and σ_w and σ_v are their corresponding in-group standard deviations. The constant $\delta > 0$ prevents division by zero, while α, β, and γ balance the three terms. Moreover, e_i and e_i^g denote the RePCB embeddings of the generated and reference layouts, respectively. As shown in Equation (6), we apply the sequence-level advantage $\hat{A}_i$ to every unmasked token in s_i, while computing $I_{i,t}(\theta)$ and clipping separately at each token position.

Furthermore, we apply the same masking rule used during SFT to the sampled scripts. Additionally, to prevent uninformative sample groups from participating in policy updates, we introduce the Legal-But-Imperfect (LBI) indicator to customize the DAPO:

$$\text{LBI}(s_i, s^g) = \mathbb{I}\left[\text{Legal}(s_i) = 1 \ \wedge \text{Layout}(s_i) \not\equiv \text{Layout}(s^g)\right], \tag{9}$$

where $\mathbb{I}$ is the indicator function, which returns only 0 or 1. s^g is the reference script associated with query q, $\text{Legal}(s_i) = 1$ indicates that s_i produces a DRC- and LVS-clean layout, and $\text{Layout}(s_i) \not\equiv \text{Layout}(s^g)$ indicates that the generated script is not identical to the reference script. Thus, $\text{LBI}(s_i, s^g)$ equals one only when s_i produces a legal script that does not exactly match the reference script; it equals zero for both illegal layouts and layouts that already match the reference. The LBI indicator effectively prevents uninformative groups from contributing to the policy update, significantly improving training efficiency. Moreover, compared with GRPO (Shao et al., 2024), DAPO omits the KL-divergence term and instead combines token-level clipping with in-group advantages. For script generation tasks related to PCB layout design, our customized DAPO can perform fine-grained policy optimization for key tokens that have a greater impact on performance.

Through SFT and customized DAPO, EvoPCB bridges the gap between LLM-generated and expert-designed layouts in terms of topology and design preference. By optimizing the script-generation task, EvoPCB can design promising initial layouts that approximate expert designs. MM-Evolve then refines these initial layouts using multi-objective simulation feedback.

3.4 Multi-Agent and Multi-Objective Script Evolution for Final Design

Optimizing geometric metrics alone is insufficient for industrial-grade PCB physical design. Conventional PnR algorithms efficiently generate valid layouts but often overlook multi-physics performance, whereas thermal and EM simulations provide actionable feedback for further optimization. To jointly optimize these potentially conflicting objectives, EvoPCB employs a centralized Multi-Agent and Multi-Objective Script Evolution (**MM-Evolve**) architecture.

Vision-based PnR Topology Optimization. Conventional PnR algorithms often leave local inefficiencies, including routing detours, redundant vias, and unnecessary bends, etc. (Chen et al., 2023; Han et al., 2026a). Because PCB layouts are generally larger in scale and sparser in geometry than integrated circuits, these inefficiencies are well suited to visual analysis. EvoPCB therefore employs a VLM-based topology optimization agent to analyze both local and global regions, as shown in Figure 4. For each layer, EvoPCB partitions the layout into uniform image cells and extends the boundary of each cell to ensure each image cell can comprehensively contain the PnR topology. Each segmented image cell, together with the corresponding global layout image, is provided to the VLM agent. The agent then identifies suboptimal wires, vias, and bends from the PnR topology and outputs structured suggestions to the coordinator agent for further decision-making.

Simulation-aware Thermal and Electromagnetic Optimization. Thermal simulation can produce a spatial heat map from which the peak temperature $\mathcal{T}_{\max}$ and maximum temperature-gradient magnitude $||\nabla\mathcal{T}||_{\max}$ are extracted. High $\mathcal{T}_{\max}$ and steep temperature gradients increase thermomechanical stress, accelerating device ageing and degrading reliability. Given the heat map and relevant context, the VLM-based agent identifies hotspots where local replacement may reduce $\mathcal{T}_{\max}$, as well as high-gradient regions where rearranging components may improve thermal uniformity. EM simulation evaluates critical nets using insertion loss (IL) and crosstalk (XT). The corresponding agent combines simulation results with layout context to identify specification violations, infer physical causes, and recommend corrective actions, including widening traces, increasing aggressor-victim spacing, or relocating components, etc. These PnR adjustment suggestions are sent to the coordinator agent for a consolidated decision.

Conflict-aware Coordination. However, since there are multiple optimization objectives, suggestions from the agents may conflict. For example, increasing component spacing may enhance thermal performance but increase local wirelength. As shown in Figure 4, EvoPCB addresses such trade-offs through a coordinator agent that acts as the central decision hub. At each iteration, the coordinator collects structured suggestions from all specialist agents and performs a compatibility check. If conflicts arise, it first converts each agent's suggestions into specific PnR actions to form an action space. According to the conflict status and importance of the actions, the coordinator agent then performs a beam search-based scheme to generate and execute the top-K conflict-free PnR scripts. The coordinator then decides which adjustments to integrate into the current design script until no candidate adjustments can improve the overall performance. Through centralized, conflict-aware coordination, MM-Evolve progressively improves the PnR script while balancing all the metrics within a unified evolutionary loop.

4 Experiment

4.1 Experimental Settings

We select Qwen3-14B as the base model of EvoPCB and Qwen3-VL-8B as the VLM agent in MM-Evolve. The unit of wirelength is mm. In the SFT stage, the batch size is 8, and the learning rate is 1.6×10^{-5}. For the LoRA configuration, the rank is 16, alpha is 32, dropout is 0.05, and the target modules include the query, key, value, and output of the attention mechanism components. We set the DAPO learning rate to 2×10^{-6}, which decays linearly to zero. During the DAPO sampling phase, we set the model temperature to 0.7 to ensure sample diversity. We set the lower and upper clipping ranges to 0.2 and 0.28, respectively. The maximum sequence length is 8192 to handle long-context script outputs. The hyperparameters α, β, and γ are analyzed in detail in Section E.5 of the appendix. We run all the experiments on a Linux server equipped with 56 Intel Xeon CPU cores and 8 NVIDIA A800 GPUs, each with 80GB of memory.

We use openEMS (Liebig, 2026) to simulate the key EM indicators of PCB layouts (Bogatin, 2010). Both IL and XT are reported in decibels (dB) and are combined into the aggregate EM cost $\mathcal{S}_{\text{EM}}$. As for thermal performance, we use OpenFOAM (OpenCFD, 2026) to simulate the peak temperature $\mathcal{T}_{\max}$ and the peak temperature-gradient magnitude $||\nabla\mathcal{T}||_{\max}$ of the PCB layouts. Because these two metrics have different units, we first normalize them into a dimensionless thermal cost $\mathcal{S}_{\text{Th}}$ to evaluate thermal performance. The implementations of EM and thermal costs are detailed in Section E.3. As for the rule checks, KiCad (KiCad, 2026) is used to perform the design rule checks (DRC) and layout-versus-schematic (LVS) verification for all PCB layouts.

Table 1: Overall performance comparison between existing PCB physical design frameworks and our EvoPCB. For all four metrics, lower values demonstrate better performance.

Case	PCB-PR-App			NS-Place + FreeRouting			PCBAgent + FreeRouting			DeepPCB			**EvoPCB (Ours)**		
	WL / Via	$\downarrow \mathcal{S}_{\text{EM}}$	$\downarrow \mathcal{S}_{\text{Th}}$	WL / Via	$\downarrow \mathcal{S}_{\text{EM}}$	$\downarrow \mathcal{S}_{\text{Th}}$	WL / Via	$\downarrow \mathcal{S}_{\text{EM}}$	$\downarrow \mathcal{S}_{\text{Th}}$	WL / Via	$\downarrow \mathcal{S}_{\text{EM}}$	$\downarrow \mathcal{S}_{\text{Th}}$	WL / Via	$\downarrow \mathcal{S}_{\text{EM}}$	$\downarrow \mathcal{S}_{\text{Th}}$
P1	1762.9 / 161	12.89	1.803	1516.2 / 66	12.56	2.058	1679.8 / 83	13.29	1.969	1195.5 / 51	10.79	2.641	1140.7 / 49	8.07	1.718
P2	1209.9 / 77	2.43	1.824	1028.4 / 68	4.89	2.101	1206.1 / 90	3.90	1.986	991.5 / 45	3.23	2.014	993.9 / 34	0.22	1.515
P3	768.5 / 56	5.88	1.539	549.0 / 27	9.52	1.965	693.8 / 33	5.64	1.798	736.8 / 39	9.74	1.979	567.6 / 24	7.85	1.444
P4	4390.6 / 215	4.93	2.982	3361.2 / 147	15.94	3.499	4175.3 / 249	4.76	3.304	3700.5 / 106	8.48	4.729	3211.4 / 112	-2.40	2.102
P5	1708.1 / 51	8.14	1.678	1872.4 / 78	18.38	2.122	1810.1 / 92	9.15	1.960	1527.5 / 53	9.39	2.120	1496.2 / 48	1.23	1.691
P6	649.4 / 40	14.32	2.446	1027.9 / 44	9.86	2.933	419.7 / 43	15.41	2.735	685.4 / 30	9.20	2.893	559.0 / 34	-7.46	1.934
P7	1489.4 / 89	6.72	1.478	1518.4 / 70	9.42	2.143	1214.3 / 97	8.50	1.889	1393.6 / 39	6.76	3.641	992.7 / 44	1.91	1.681
P8	688.6 / 12	10.49	2.166	1129.3 / 89	10.69	2.162	586.7 / 40	15.60	2.206	917.8 / 26	11.34	2.661	579.5 / 29	-2.31	1.624
P9	741.1 / 20	2.82	2.193	909.8 / 39	10.39	3.156	825.2 / 43	6.05	2.509	990.3 / 38	1.49	2.438	913.3 / 19	7.82	1.992
P10	2165.1 / 144	-1.09	1.789	2099.6 / 114	2.91	1.898	2056.7 / 201	-0.98	1.846	1946.8 / 54	1.12	2.656	1885.9 / 87	7.74	1.566
P11	1253.8 / 34	12.74	2.109	1379.8 / 45	12.47	3.701	1579.4 / 67	10.46	3.200	1345.6 / 56	14.41	6.601	953.1 / 38	7.96	1.826
Avg.	1529.8 / 81.7	7.30	2.001	1490.2 / 71.5	10.64	2.522	1477.0 / 94.4	8.34	2.309	1402.8 / 48.8	7.81	3.125	**1208.5 / 47.1**	**2.78**	**1.736**

Table 2: PCB layout performance of EvoPCB and the GPT-6 Astra agent integrated with KiCad.

Design Method	Metrics	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	Avg.
GPT-6 Astra	WL	1129.9	1098.2	544.1	2685.4	1730.4	914.2	1125.6	844.5	834.1	2164.5	1735.8	1346.1
	Via	80	244	94	118	46	54	147	36	34	80	38	88.3
	$\mathcal{S}_{\text{EM}}$	3.30	8.76	12.93	10.09	10.48	5.48	3.23	6.21	11.44	10.06	11.22	8.47
	$\mathcal{S}_{\text{Th}}$	1.990	1.658	2.472	2.324	1.789	2.066	2.352	2.067	2.456	2.287	2.533	2.181
EvoPCB (Ours)	WL	1140.7	993.9	567.6	3211.4	1496.2	559.0	992.7	579.5	913.3	1885.9	953.1	**1208.5**
	Via	49	34	24	112	48	34	44	29	19	87	38	**47.1**
	$\mathcal{S}_{\text{EM}}$	8.07	0.22	7.85	-2.40	1.23	-7.46	1.91	-2.31	7.82	7.74	7.96	**2.78**
	$\mathcal{S}_{\text{Th}}$	1.718	1.515	1.444	2.102	1.691	1.934	1.681	1.624	1.992	1.566	1.826	**1.736**

4.2 Benchmarks and Baselines

All PCB layout benchmarks are from OpenROAD (OpenROAD, 2019a) and real-world industrial applications, as shown in Table 6. The end-to-end PCB PnR baselines in our experiments are as follows, including both SOTA commercial tools and academic frameworks:

1. **PCB-PR-App** (OpenROAD, 2019b): an open-source and effective framework for PCB PnR, comprising SA-PCB (OpenROAD, 2019d) and PcbRouter (OpenROAD, 2019c). The PnR process is implemented using simulated annealing and A* search, respectively.
2. **NS-Place** (Cheng et al., 2022) + **FreeRouting** (Fuchs, 2026): NS-Place is a density-aware framework for general PCB placement that reduces net congestion to enhance routability. FreeRouting is a powerful, fully autonomous PCB routing software.
3. **PCBAgent** (Chen et al., 2025) + **FreeRouting** (Fuchs, 2026): PCBAgent is an effective LLM-based framework for placement, which utilizes an RL agent to perform optimization and multiple masks (Lai et al., 2022) to generate design constraints. We also implement FreeRouting to complete the whole PCB PnR design.
4. **DeepPCB** (InstaDeep, 2026): a powerful RL-based commercial tool for PCB PnR; its placer and router leverage lots of design experience through learning.
5. **GPT-6 Astra** (OpenAI, 2026): a current SOTA LLM that can be directly integrated with several PCB design tools, allowing it to act as an agent for PCB placement, routing, and layout refinement.

4.3 Experiment Results

Overall Performance. Table 1 summarizes the overall performance comparison of PCB physical design. Compared with PCB-PR-App (OpenROAD, 2019b) and NS-Place (Cheng et al., 2022) combined with FreeRouting (Fuchs, 2026), EvoPCB reduces the wirelength by 21.0% and 18.9%, respectively, and the via count by 42.4% and 34.2%. As for PCBAgent (Chen et al., 2025),

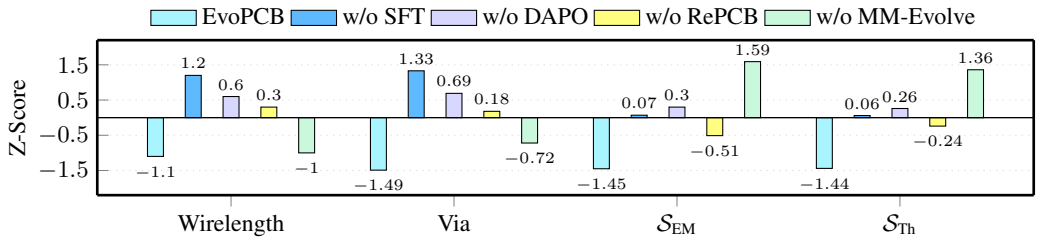


Figure 5: Ablation study. Results are z-score normalized within each metric, and lower values indicate better performance, so the changes caused by removing a component reflect its importance.

EvoPCB saves 18.2% wirelength and 50.1% via usage. Compared with DeepPCB (InstaDeep, 2026), EvoPCB achieves a 13.9% reduction in wirelength and a 3.5% reduction in via count. In terms of the scaled multi-physics costs, EvoPCB lowers the aggregate EM cost by 4.52, 7.86, 5.56, and 5.03 relative to PCB-PR-App, NS-Place, PCBAgent, and DeepPCB, respectively, and lowers the thermal cost by 0.265, 0.786, 0.573, and 1.389 relative to PCB-PR-App, NS-Place, PCBAgent, and DeepPCB, respectively. Overall, EvoPCB achieves the best performance across all metrics among the methods in Table 1, while case-level trade-offs remain. These results support its effectiveness in jointly improving routing quality and the evaluated EM and thermal objectives. To evaluate the SOTA LLM agent baseline, we integrate GPT-6 Astra (OpenAI, 2026) with KiCad and compare it with EvoPCB in Table 2. EvoPCB reduces the wirelength from 1346.1 to 1208.5, and the via count from 88.3 to 47.1, respectively, and lowers the aggregate EM cost from 8.47 to 2.78 and the thermal cost from 2.181 to 1.736. Therefore, although GPT-6 Astra is capable of generating PCB layouts, it still lacks expert design experience and performance considerations due to the absence of specialized training. More detailed comparisons and performance analysis between GPT-6 Astra and our EvoPCB are in Section E.2.

Ablation Study. To further investigate the contribution of each component to the overall performance of EvoPCB, we conduct an ablation study, as shown in Figure 5. Overall, the complete EvoPCB framework achieves the best and most balanced performance across several metrics, indicating that these components play complementary roles in the optimization process. Removing SFT causes large drops across all metrics, especially wirelength and via count, showing SFT is essential for EvoPCB to understand and correctly invoke PnR APIs and to provide reliable initialization for layout optimization. Removing DAPO also consistently reduces layout quality, confirming RL is crucial for translating metric feedback into effective optimization. Although milder than SFT removal, degradation in multi-physics and conventional objectives shows DAPO helps refine API usage and discover globally better solutions. Removing RePCB causes moderate but consistent declines, confirming that PCB-specific representation provides stable reward signals across training stages. In contrast, removing MM-Evolve severely hurts electromagnetic and thermal performance while slightly affecting routing metrics. This trade-off shows that without multimodal evolutionary feedback, optimization favors conventional routing objectives and overlooks electromagnetic and thermal board properties. MM-Evolve is thus indispensable for guiding EvoPCB to satisfy routing efficiency and multi-physics requirements jointly.

5 Conclusion

This paper proposed EvoPCB, an end-to-end LLM-based framework for expert-like PCB physical design, transferring the design task into a code generation task. For automated evaluation and dataset generation of PCB layouts, we first proposed RePCB, a supervised, performance-aware embedding model to represent layouts and compute their similarities with expert-level designs. To enable the LLM to truly master expert design logic and topologies, EvoPCB employs a two-stage specialization approach: SFT is used to help the model understand the PnR APIs and perform initial optimization, followed by customized DAPO to further improve performance and meet the design preferences of human engineers. To specifically enhance multi-physics performance, we proposed MM-Evolve, leveraging a multi-agent cooperative scheme that optimizes various performance metrics by adjusting the initial script. Experimental results show that EvoPCB outperforms all existing state-of-the-art baselines, achieving the best overall performance on the industrial-grade benchmarks.

AI USE STATEMENT

In this work, we used generative AI tools for aiding and polishing paper writing, and assisting with programming. We have not used generative AI tools for other tasks. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Eric Bogatin. *Signal and power integrity–simplified*. Pearson Education, 2010.

Eric Bogatin. *Bogatin's Practical Guide to PCB Design for New Product Development*. Artech, 2021.

Chen-Chia Chang, Chia-Tung Ho, Yaguang Li, Yiran Chen, and Haoxing Ren. Drc-coder: Automated drc checker code generation using llm autonomous agent. In *Proceedings of the 2025 International Symposium on Physical Design*, pp. 143–151, 2025.

Lin Chen, Ran Chen, Shoubo Hu, Xufeng Yao, Zhentao Tang, Shixiong Kai, Siyuan Xu, Mingxuan Yuan, Jianye Hao, Bei Yu, et al. PCBAgent: An agent-based framework for high-density printed circuit board placement. In *Proceedings of the 30th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 781–787, 2025.

Tinghuan Chen, Silu Xiong, Huan He, and Bei Yu. TRouter: thermal-driven PCB routing via non-local crisscross attention networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(10):3388–3401, 2023.

Chung-Kuan Cheng, Chia-Tung Ho, and Chester Holtz. Net separation-oriented printed circuit board placement via margin maximization. In *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 288–293. IEEE, 2022.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

Zhaohai Di, Jindong Tu, Zhiyuan He, Yuan Pu, Jiawei Liu, Chong Tong, Tsung-Yi Ho, Bei Yu, and Tinghuan Chen. Smart-pclib: A llm-based multi-agent framework for automated pcb component library generation. In *2026 Design, Automation & Test in Europe Conference (DATE)*, pp. 1–7. IEEE, 2026.

Andras Fuchs. Freerouting, 2026. URL `https://www.freerouting.app/`.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

Shanshan Han, Qifan Zhang, Weizhao Jin, and Zhaozhuo Xu. Llm multi-agent systems: Challenges and open problems. *arXiv preprint arXiv:2402.03578*, 2024.

Yaohui Han, Beichen Li, Rongliang Fu, Qunsong Ye, Zhiyuan Lu, Junchen Liu, Bei Yu, Tsung-Yi Ho, and Tinghuan Chen. PCB-Migrator: Automated PCB PnR migration. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–7. IEEE, 2026a.

Yaohui Han, Beichen Li, Mingyang Zhao, Rongliang Fu, Qunsong Ye, Tinghuan Chen, Bei Yu, and Tsung-Yi Ho. ModuPlace: LLM-Assisted Modular PCB Placement via Preference-Optimized Constraint Graph Generation. In *2026 Design, Automation Conference (DAC)*, pp. 1–7, 2026b.

Hao-Hsiang Hsiao, Yi-Chen Lu, Sung Kyu Lim, and Haoxing Ren. Buffalo: Ppa-configurable, llm-based buffer tree generation via group relative policy optimization. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–9. IEEE, 2025.

InstaDeep. DeepPCB, 2026. URL `https://app.deeppcb.ai/`.

Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

KiCad. Kicad, 2026. URL https://www.kicad.org/.

Yao Lai, Yao Mu, and Ping Luo. Maskplace: Fast chip placement via reinforced visual representation learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:24019–24030, 2022.

Shanyi Li, Zhen Zhuang, Mingyu Liu, Weihua Sheng, Bei Yu, and Tsung-Yi Ho. HiePlace: Efficient hierarchical PCB placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 45(1):428–440, 2026.

Yunxuan Li, Yibing Du, Jiageng Zhang, Le Hou, Peter Grabowski, Yeqing Li, and Eugene Ie. Improving multi-agent debate with sparse communication topology. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 7281–7294, 2024.

Thorsten Liebig. openEMS, 2026. URL https://www.openems.de/.

Ting-Chou Lin, Devon Merrill, Yen-Yi Wu, Chester Holtz, and Chung-Kuan Cheng. A unified printed circuit board routing algorithm with complicated constraints and differential pairs. In *Proceedings of the 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 170–175, 2021.

Siting Liu, Qi Sun, Peiyu Liao, Yibo Lin, and Bei Yu. Global placement with deep learning-enabled explicit routability optimization. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1821–1824. IEEE, 2021.

Taiting Lu, Kaiyuan Lin, Ziwei Dong, Sisong Bei, Haolin Ye, Yuxin Tian, Runze Liu, Mingjia Wang, Jingying Zeng, Hongxing Pan, et al. Omnirouting: A semantic-coupled multimodal benchmark for constraint-aware spatial reasoning in pcb routing. *arXiv preprint arXiv:2608.04434*, 2026a.

Taiting Lu, Kaiyuan Lin, Mingjia Wang, Haolin Ye, Runze Liu, Yuxin Tian, Vahe Melkonyan, Haoyu Wang, Muchuan Wang, Chufan Hong, et al. Omnilayout: A schematic-coupled multimodal benchmark for constraint-aware geometric reasoning in pcb layout. *arXiv preprint arXiv:2607.03261*, 2026b.

Xiaowen Ma, Yunpu Ma, Chenyang Lin, Sikuan Yan, Jinhe Bi, Zixuan Cao, Yijun Tian, Volker Tresp, and Hinrich Schuetze. Self-evolving multi-agent systems via textual backpropagation. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 9918–9951, 2026.

David JC MacKay et al. Introduction to gaussian processes. *NATO ASI series F computer and systems sciences*, 168:133–166, 1998.

Ancău Mircea. Main aspects concerning PCB manufacturing optimization. *Circuit World*, 38(2): 75–82, 2012.

Kraig Mitzner. *Complete PCB design using OrCAD Capture and PCB editor*. Newnes, 2009.

OpenAI. GPT-6 Astra, 2026. URL https://openai.com/index/gpt-6-astra/.

OpenCFD. OpenFOAM, 2026. URL https://www.openfoam.com/.

OpenROAD. OpenROAD, 2019a. URL https://github.com/the-openroad-project.

OpenROAD. PCB-PR-App, 2019b. URL https://github.com/The-OpenROAD-Project/PCB-PR-App.

OpenROAD. PcbRouter, 2019c. URL https://github.com/The-OpenROAD-Project/PcbRouter.

OpenROAD. SA-PCB, 2019d. URL https://github.com/The-OpenROAD-Project/SA-PCB.

Shuo Ren, Zijin Cheng, Yaohui Han, Libo Shen, Leilei Jin, Wanting Tian, Rongliang Fu, Chao Wang, Bei Yu, and Tsung-Yi Ho. Agenticpd: A stage-aware agentic framework for physical design qor optimization. *arXiv preprint arXiv:2607.04758*, 2026.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Matthias Seeger. Gaussian processes for machine learning. *International journal of neural systems*, 14(02):69–106, 2004.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Chien-Hao Tsou, Sheng-Yah Lin, Wei-Chen Hung, and Yao-Wen Chang. Late breaking results: Modern automatic PCB placement with complex constraints. In *Design Automation Conference (DAC)*, pp. 1–2, 2024.

Ikechukwu Uchendu, Swati Goel, Karly Hou, Ebrahim Songhori, Kuang-Huei Lee, Joe Wenjie Jiang, Vijay Janapa Reddi, and Vincent Zhuang. See it to Place it: Evolving Macro Placements with Vision-Language Models. *arXiv preprint arXiv:2603.28733*, 2026.

Yiru Wang, Xinyue Shen, Yaohui Han, Michael Backes, Pin-Yu Chen, and Tsung-Yi Ho. Orgagent: Organize your multi-agent system like a company. *arXiv preprint arXiv:2604.01020*, 2026.

Haoyuan Wu, Haisheng Zheng, Zhuolun He, and Bei Yu. Divergent thoughts toward one goal: Llm-based multi-agent collaboration system for electronic design automation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1710–1721, 2025.

Tzong-Lin Wu, Hao-Hsiang Chuang, and Ting-Kuang Wang. Overview of power integrity solutions on package and PCB: Decoupling and EBG isolation. *IEEE Transactions on electromagnetic compatibility*, 52(2):346–356, 2010.

Shuwen Yang, Zhihao Yang, Dong Li, Yingxueff Zhang, Zhanguang Zhang, Guojie Song, and Jianye Hao. Versatile multi-stage graph neural network for circuit representation. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:20313–20324, 2022.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems (NeurIPS)*, 38:113222–113244, 2026.

Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Dongsheng Li, and Deqing Yang. Evoagent: Towards automatic multi-agent generation via evolutionary algorithms. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6192–6217, 2025.

Zelin Zang, Yuhang Song, Aili Wang, Bingo Wing-Kuen Ling, Qi Sun, Zhen Lei, Fuji Yang, Cheng Zhuo, and Jiebo Luo. The dawn of agentic eda: A survey of autonomous digital chip design. *arXiv preprint arXiv:2512.23189*, 2025.

Dawei Zhan and Huanlai Xing. Expected improvement for expensive optimization: a review. *Journal of Global Optimization*, 78(3):507–544, 2020.

Yuxiang Zhao, Xun Jiang, Qiang Xu, Runsheng Wang, Yibo Lin, et al. DeepLayout: Learning neural representations of circuit placement layout. In *International Conference on Machine Learning (ICML)*, 2025.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A Additional Backgrounds

A.1 PCB Physical Design

PCB physical design primarily consists of two stages: placement and routing (PnR). Placement involves determining the specific locations and orientations of components from the schematic within a designated PCB board frame, ensuring no spatial overlap and adherence to minimum spacing constraints; components may generally be placed on either the top or bottom side of the PCB board (Cheng et al., 2022; Han et al., 2026b; Chen et al., 2025). Routing involves connecting pins that share electrical connections using conductive wires, based on the provided PCB netlist. To ensure EM performance, routing bends are typically restricted to 45-degree and 135-degree angles. While modern PCBs often utilize vias to route wires across multiple layers, thereby alleviating congestion, this approach also significantly expands the solution space for routing (Chen et al., 2023). At the same time, the quality of the placement largely determines the routing result; human engineers typically need to foresee routing during the placement stage to ensure routing quality (Han et al., 2026b). However, existing frameworks cannot automate this process. Following PnR, human engineers typically conduct performance evaluation and iterative layout refinements, alongside DRC and LVS verification, to ensure the design meets all specifications and functions correctly. Given the multi-step nature of the design process and the reliance on expert experience, automating PCB physical design has long remained a challenging problem. Existing PCB physical design automation frameworks fail to meet industrial-grade requirements; most address only isolated issues (Han et al., 2026a; Lin et al., 2021; Chen et al., 2023) and lack an end-to-end design workflow, preventing them from considering the PCB physical design from a holistic perspective.

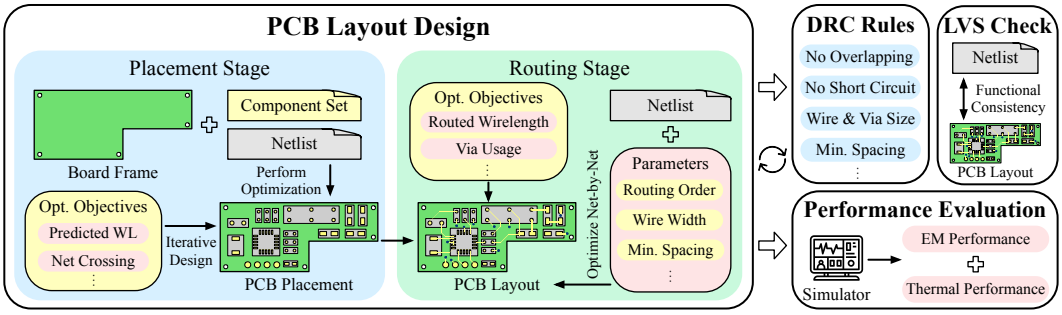


Figure 6: PCB physical design flow, primarily including placement, routing, DRC&LVS checks, and performance evaluation.

A.2 Decoupled Clip and Dynamic Sampling Policy Optimization

There are several methods for training the LLM agent to align with human preferences, namely Reinforcement Learning from Human Feedback (RLHF) (Christiano et al., 2017; Ziegler et al., 2019). Among them, Proximal Policy Optimization (PPO) (Schulman et al., 2017) and Group Relative Policy Optimization (GRPO) (Shao et al., 2024) have been applied widely. However, PPO requires training a value network whose size is comparable to the reference model as a critic to precisely estimate the reward function. As for GRPO, it removes the dependency on the value model, significantly improving scalability, but it still falls short in terms of efficiency and stability. GRPO employs sample-level loss calculation, which may ignore the impact of key tokens in long sequences, leading to difficulties in training convergence. Therefore, in some complicated generation tasks, such as physical design in EDA, applying these two methods may be quite challenging.

To address these limitations, Decoupled Clip and Dynamic Sampling Policy Optimization (DAPO) (Yu et al., 2026) provides a more efficient and robust policy-optimization procedure. Its Clip-Higher mechanism decouples the clipping interval into a lower bound ϵ_{low} and an upper bound ϵ_{high}. A larger upper margin allows low-probability tokens to receive stronger positive updates, increasing policy diversity, while a conservative lower margin limits destructive negative updates and helps prevent policy collapse. DAPO also uses dynamic sampling to retain groups with nonzero reward variance, ensuring that each retained group contributes a meaningful relative-advantage gradient. Finally,

token-level loss aggregation weights valid generated tokens directly rather than first averaging the loss within each response. This formulation avoids underweighting informative decisions in long structured outputs such as PCB PnR scripts.

In PCB physical design, because the solution space is large and the number of samples is limited, the model is prone to getting trapped in a local optimum or lacking effective gradient updates (Chen et al., 2025; Han et al., 2026b; Hsiao et al., 2025). DAPO's Clip-Higher and Dynamic Sampling mechanisms precisely address this problem. Moreover, token-level policy gradient loss allows the model to more specifically quantify the impact of specific steps or parameter settings in physical design on the overall result.

A.3 Multi-Agent Collaborative Evolution

LLM-based multi-agent systems distribute a complex task among interacting agents rather than relying on a single model trajectory (Han et al., 2024). A common design assigns agents complementary roles and coordinates their intermediate outputs through message passing or a shared workspace (Di et al., 2026). Role specialization can reduce the context and tool knowledge required by each agent, while cross-agent review provides an additional mechanism for detecting errors and combining distinct solution strategies (Wang et al., 2026). The coordination structure may be centralized, with a manager aggregating specialist outputs (Di et al., 2026), or decentralized, with agents communicating directly (Li et al., 2024); the appropriate choice depends on whether a task benefits more from controlled decomposition or open-ended exploration. Among these, the evolution of generation targets by multi-agent systems is considered an effective method for enhancing the generative qualities of the LLM-based system (Ma et al., 2026). Individual agents play distinct, fixed roles within the overall generation system, handling tasks such as generation, execution, and verification, under the unified coordination and control of a coordinator agent, thereby significantly enhancing generation quality and reliability within a limited number of iterations (Yuan et al., 2025). Crucially, the paradigm places adaptive decision-making at the heart of the evolutionary process. Multi-Agent systems can modulate their exploration-exploitation trade-offs, decide when to diverge from the current solution, and alter the evaluation criteria they prioritize (Wu et al., 2025). In EvoPCB, MM-Evolve adopts a more constrained variant tailored to PCB design: specialist roles and the coordinator are fixed, while evolution occurs through iterative proposal, conflict resolution, execution, and re-evaluation of layout modifications. This centralized structure preserves domain-specific responsibilities and ensures that every accepted action is converted into a legal PnR API sequence.

B Implementation Details of Main PnR APIs

EvoPCB implements multiple PnR-related APIs for LLM calling. Specifically, we provide three placement methods: simulated annealing (SA), mixed-integer linear programming (MILP), and differentiable programming (DP). All three require appropriate invocation order and parameter settings to produce high-quality results. In addition, EvoPCB provides a constraint-setting API that encodes pairwise constraints between components. For a component pair (c_i, c_j), the API specifies the desired relative displacement $(\Delta x^*_{ij}, \Delta y^*_{ij})$, relative orientation $\Delta\theta^*_{ij}$, and constraint weight ε. These solver-agnostic constraints are converted into penalty terms in the placement objective, allowing all three placers to preserve expert-specified relative geometry. For routing, we implement A* search as the basic router and expose configurable parameters such as movement costs and grid resolution. In addition, some other key functions, such as rip-up and reroute, are also encapsulated as APIs for LLM invocation to enhance flexibility and performance.

B.1 Implementation of Placement-related APIs

Simulated Annealing (SA) is used in PCB placement as a stochastic, discrete optimizer over component positions, orientations, and even layer assignments (OpenROAD, 2019d). The placement state is represented by a vector of components, each with geometric attributes (center, size, orientation, layer, and pin offsets). The search starts from either a user-provided initial placement or a random configuration. At each step, the placer applies a random perturbation to the current state, choosing among several move types: swapping two components, shifting a component, rotating it, or changing its layer, with predefined probabilities. After generating a candidate move, the algorithm

Table 3: Overview of PnR API implementation. Only APIs marked with "†" can be generated in the SFT stage; these are essential APIs for generating the initial design. Other APIs are used during the evolution process to further improve the layout quality.

API Name	Purpose	#Parameters
	Placement-related APIs	
SA-Place()†	Perform simulated annealing-based placement	3
Diff-Place()†	Perform differentiable programming-based placement	2
MILP-Place()†	Perform mixed-integer linear programming-based placement	4
Constraint-Set()†	Set pairwise relative-position and orientation constraints	4
Comp-Fix()	Place certain components at specific positions	4
Location-Search()	Greedy search for legal placement for a single component	4
	Routing-related APIs	
A*-Route()†	A* search-based PCB routing	3
NetOrder-Set()†	Set the net order of PCB routing	3
Ripup-Reroute()†	Rip-up and reroute for specific nets	4
CostMap-Set()†	Set the cost parameters in costmap	2
Modify-Segment()	Modify the width, start/end point, etc., of a specific wire segment	4
Set-Spacing()	Set the minimum spacing constraint between two nets	4

computes a composite cost that combines normalized wirelength, overlap penalty, and optionally routability-related density terms. The move is accepted according to the Metropolis criterion: a move that reduces the cost is always accepted, while a move that increases cost is accepted with probability $\exp(-\Delta C/T)$, where T is the current temperature.

A key design choice in the SA-based approach is that the search space is constrained by board boundaries not through a hard feasibility solver, but by clipping each proposed component position to the legal PCB envelope, ensuring its bounding box stays within the board limits (determined from either edge cuts or pin locations, depending on configuration). Overlap is not forbidden outright; instead, it is measured geometrically via polygon intersection and added as a penalty term in the objective function. Thus, overlap is treated as a soft constraint that guides the optimization toward non-overlapping arrangements. The algorithm keeps track of the best solution found so far (the one with the lowest overlap) and, at the end, restores that best recorded solution rather than the final annealing state. Consequently, our algorithm biases the result toward legal placements through boundary clipping and overlap penalization, but legality is not strictly guaranteed by the annealing process alone; a final design-rule or geometry check is still advisable.

Differentiable Programming (DP) underpins modern analytical global placement by relaxing the discrete layout problem into a continuous optimization framework. The method formulates a differentiable objective—typically comprising a smooth wirelength approximation and a density penalty that mitigates component overlap—and iteratively updates component coordinates along the negative gradient direction. Because the relaxed objective remains non-convex, the method generally converges to a local optimum; initialization and the density-penalty schedule therefore affect final quality. In our implementation, each component i is assigned an equivalent charge q_i proportional to its physical component area ($w_i \times h_i$), and the resulting electrostatic potential provides the density term. For the wirelength term, we define $\widetilde{\text{HPWL}}_m$ as a differentiable approximation of HPWL for net m using the Log-Sum-Exp (LSE) function:

$$\widetilde{\text{HPWL}}_m = \alpha \ln\Big(\sum_{p\in\mathcal{P}_m} e^{x_p/\alpha}\Big) + \alpha \ln\Big(\sum_{p\in\mathcal{P}_m} e^{-x_p/\alpha}\Big) + \alpha \ln\Big(\sum_{p\in\mathcal{P}_m} e^{y_p/\alpha}\Big) + \alpha \ln\Big(\sum_{p\in\mathcal{P}_m} e^{-y_p/\alpha}\Big), \tag{10}$$

where $\mathcal{P}_m$ is the set of pin coordinates belonging to net m, and α is a smoothing parameter; smaller α yields a tighter approximation to the true HPWL.

Mixed-Integer Linear Programming (MILP) models placement as a discrete optimization problem with a linear objective and linearized spatial and non-overlap constraints. Unlike the preceding heuristic methods, MILP does not require a user-provided feasible initialization. A solver can certify

Algorithm 1 A* Search for PCB Routing

Require: Grid G, terminals s, t, obstacles O, movement cost c, heuristic h
Ensure: Path P from s to t, or failure
1: $\Omega \leftarrow \{s\}$, $\Phi \leftarrow \emptyset$; $g(s) \leftarrow 0$, $f(s) \leftarrow h(s)$ ▷ Initialize the sets and the start cell
2: $g(v), f(v) \leftarrow \infty$ and $parent(v) \leftarrow$ null for all $v \in G \setminus \{s\}$
3: **while** $\Omega \neq \emptyset$ **do**
4: Remove a cell $n \leftarrow \arg\min_{v \in \Omega} f(v)$ from Ω ▷ Pop the open cell with the lowest priority
5: **if** $n = t$ **then**
6: **return** path obtained by tracing $parent$ from t to s ▷ Reconstruct and return the path
7: **end if**
8: $\Phi \leftarrow \Phi \cup \{n\}$ ▷ Mark n as finalized
9: **for** each neighbor m of n that is inside G and not in O **do**
10: $\hat{g} \leftarrow g(n) + c(n, m)$ ▷ Tentative cost of reaching m through n
11: **if** $m \notin \Phi$ and $\hat{g} < g(m)$ **then**
12: $parent(m) \leftarrow n$; $g(m) \leftarrow \hat{g}$; $f(m) \leftarrow \hat{g} + h(m)$
13: Insert m into Ω, or update its key if already present
14: **end if**
15: **end for**
16: **end while**
17: **return** -1 ▷ Open set exhausted: no feasible path exists

global optimality for the formulated MILP if it terminates at the required optimality gap; solving only the linear-programming relaxation does not, in general, produce a legal integral placement. Practical scalability is limited by the growth in binary separation variables and constraints. Given relative-position constraints generated by the LLM, the MILP balances their weighted violations against HPWL. Because continuous coordinates (x, y) and discrete orientations θ must be solved jointly, we formulate the objective as

$$\min \sum_{i=1}^{|C|} \sum_{j=i+1}^{|C|} \mathcal{W}_{ij} \left(|\Delta x_{ij}^* - \Delta x_{ij}| + |\Delta y_{ij}^* - \Delta y_{ij}| \right) + \gamma_1 \sum_{i=1}^{|C|} \Delta\theta_i + \gamma_2 \sum_{m=1}^{|\mathcal{N}|} \mathrm{HPWL}_m(x, y, \theta),$$
$$\text{s.t. } \theta_i \in \{0^\circ, 90^\circ, 180^\circ, 270^\circ\}, \ \forall i \in C, \tag{11}$$

where γ_1 and γ_2 balance orientation changes and HPWL, respectively. For components i and j, $(\Delta x_{ij}^*, \Delta y_{ij}^*)$ is the API-specified target displacement and $(\Delta x_{ij}, \Delta y_{ij})$ is the displacement in the candidate placement. $\Delta\theta_i$ denotes the deviation from the requested orientation. Absolute-value terms and non-overlap disjunctions are linearized using standard auxiliary and binary variables.

B.2 Implementation of Routing-related API

A* Search models PCB routing as a shortest-path problem on a discretized routing graph. Each vertex represents a legal grid location (and, in a multilayer extension, a location-layer pair), while an edge represents an allowed routing action such as a horizontal or vertical move, a diagonal move when the manufacturing rules permit it, or a layer transition through a via. The grid, terminals, obstacles, movement cost, and heuristic are supplied as the routing instance (lines 1-2). Cells occupied by components, keep-out regions, or clearance-expanded existing conductors are included in O and are therefore excluded from expansion (line 11). The edge cost $c(n, m) \geq 0$ can combine geometric length with penalties for vias, bends, local congestion, proximity to obstacles, and other design-rule preferences (line 12). Consequently, the returned path minimizes the accumulated cost encoded by the router rather than necessarily minimizing geometric length alone.

For each cell n, A* maintains the best discovered source cost $g(n)$, an estimate $h(n)$ of the remaining cost to t, and the priority $f(n) = g(n) + h(n)$. The start cell is initialized with $g(s) = 0$, all other cells are assigned infinite costs, and all parent pointers are cleared (lines 3-4). The open set Ω stores discovered cells that may still be expanded, whereas the closed set Φ stores cells whose lowest-cost paths have been finalized (lines 3 and 10). As summarized in Algorithm 1, the main loop repeatedly removes the open cell with minimum f (lines 5-6). If this cell is the target, the algorithm follows its parent pointers backward and returns the reconstructed path (lines 7-9). Otherwise, the cell is closed,

Algorithm 2 GraphSAGE-based CPG Processing

Require: CPG $\mathcal{G}(\mathcal{V}, \mathcal{E})$; vertex features $\mathcal{F}$; edge attributes $\mathcal{A}$; depth D
Ensure: Placement feature $\mathbf{P}$
1: $\mathbf{h}_v^0 \leftarrow \mathrm{Linear}_c(\mathcal{F}_v), \quad \forall v \in \mathcal{V}_c$ ▷ Component projection
2: $\mathbf{h}_v^0 \leftarrow \mathrm{Linear}_n(\mathcal{F}_v), \quad \forall v \in \mathcal{V}_n$ ▷ Net projection
3: **for** $k = 1, \ldots, D$ **do**
4: **for** $v \in \mathcal{V}$ **do**
5: $\mathbf{m}_{u \to v}^k \leftarrow \mathrm{MLP}_k([\mathbf{h}_u^{k-1} \| \mathcal{A}_{(u,v)}]), \quad \forall u \in \mathcal{N}(v)$ ▷ Edge-conditioned message
6: $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow (1/|\mathcal{N}(v)|) \cdot \sum_{u \in \mathcal{N}(v)} \mathbf{m}_{u \to v}^k$ ▷ Mean aggregation
7: $\mathbf{h}_v^k \leftarrow \mathbf{W}_{\mathrm{self}}^k \mathbf{h}_v^{k-1} + \mathbf{W}_{\mathcal{N}}^k \mathbf{h}_{\mathcal{N}(v)}^k$ ▷ Combine self and neighborhood
8: **end for**
9: $\mathbf{h}^k \leftarrow \mathrm{Dropout}\left(\sigma(\mathrm{LayerNorm}(\mathbf{h}^k))\right)$ ▷ Normalize, activate, regularize
10: **if** $k > 1$ **then**
11: $\mathbf{h}^k \leftarrow \mathbf{h}^k + \mathbf{h}^{k-1}$ ▷ Residual connection
12: **end if**
13: **end for**
14: $\mathbf{P} \leftarrow \mathrm{MLP}\left([\mathrm{MeanPool}(\mathbf{h}^D) \| \mathrm{MaxPool}(\mathbf{h}^D)]\right)$ ▷ Global readout: pool then project
15: **return** $\mathbf{P}$

and each legal neighbor is examined (lines 10-11). A relaxation is accepted only when the candidate cost $\hat{g} = g(n) + c(n, m)$ improves the neighbor's current g-value (lines 12-13); the algorithm then records n as its parent, updates $g(m)$ and $f(m)$, and inserts or reprioritizes m in Ω (lines 14-15). Exhausting Ω without extracting t means that no path exists in the current discretized graph under the specified obstacles and movement rules (lines 18-19).

This closed-set variant is optimal when h is consistent, i.e., $h(n) \leq c(n, m) + h(m)$ for every edge and $h(t) = 0$. Consistency ensures that f does not decrease along a path and justifies permanently excluding closed cells from relaxation (lines 10 and 13). For orthogonal routing with a minimum per-grid-step cost $c_{\min}$, $h(n) = c_{\min}(|x_n - x_t| + |y_n - y_t|)$ is consistent; a layer-aware heuristic may additionally use a lower bound on the required via cost. If the heuristic is only admissible but inconsistent, the condition in line 13 must be modified so that a closed cell is reopened whenever a lower g-value is found. With a binary heap for Ω, the minimum-priority extraction and key update operations (lines 6 and 15) give a worst-case time complexity of $O(|E| \log |V|)$; the cost, parent, and set records (lines 3-4) require $O(|V|)$ storage. An informative heuristic typically causes A* to explore far fewer cells than uninformed maze routing.

C Implementation Details of RePCB

As introduced in Section 3.2, RePCB jointly encodes the PnR structure of a PCB layout through a GraphSAGE-based placement branch and a CNN-based routing branch, and its embedding is supervised by physical performance, including wirelength and via usage. This section details the implementation of the two branches in the proposed RePCB model.

Placement Features. RePCB operates on the Comprehensive Placement Graph (CPG) $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ defined in Section 3.2, where $\mathcal{V} = \mathcal{V}_c \cup \mathcal{V}_n$ combines component and net vertices, and each edge (u, v) carries the scale-normalized geometric attribute $\mathcal{A}_{(u,v)}$. Algorithm 2 summarizes the forward pass of the edge-conditioned GraphSAGE-based model. Component and net vertices are first projected into a shared latent space through separate linear layers (lines 1-2). The network then performs D rounds of message passing: for each vertex v, the message from a neighbor u is produced by an MLP that takes the neighbor embedding and the edge attribute as input (line 5); these messages are averaged to form the neighborhood embedding (line 6), which is combined with the self-embedding through layer-specific weight matrices (line 7). After each round, layer normalization, activation, and dropout are applied (line 9), and a residual connection is added from the previous round (lines 10-11). Finally, mean and max pooling over all component and net vertices, followed by an MLP, produce the placement feature $\mathbf{P}$ (line 14). The definitions of the input vertex features $\mathcal{F}_c$ and $\mathcal{F}_n$ are listed in Table 4. $\mathcal{F}_c$ contains component-related features, including the size, position, and ro-

Table 4: Vertex features in the proposed CPG.

Feature	Type	Definition
$\mathcal{F}_c$	Component	Normalized center $(x/W, y/H)$
		Normalized width and height
		Pin count and net count
	Geometric	Orientation $(\sin\theta, \cos\theta)$
		Copper-side flag
		Movability flag
$\mathcal{F}_n$	Connection	Corresponding components
	Span	Range along the horizontal direction
		Range along the vertical direction

Table 5: Description of the proposed six-channel routing image.

Channel	Content	Definition
1	Front pin occupancy	Binary, shared with Ch. 2
2	Bottom pin occupancy	Binary, shared with Ch. 1
3	Local net multiplicity	$\min(1, \log(1 + N_{\text{net}})/3)$
4	Front wire density	$\log(1 + \cdot)$, shared with Ch. 5
5	Bottom wire density	$\log(1 + \cdot)$, shared with Ch. 4
6	Via density	$\log(1 + \cdot)$, min-max

tation status, etc. As for $\mathcal{F}_n$, it describes the net connection information, including components corresponding to each net, and the range of the net in the horizontal and vertical directions.

Routing Features. RePCB describes the routed layout through a six-channel image, whose channels are detailed in Table 5. The first two channels record pin occupancy on the front and bottom copper layers; if the current grid contains an area belonging to a pin, it is assigned a value of 1; otherwise, it is assigned a value of 0. The third channel captures local net multiplicity; the grid value for this channel corresponds to the number of net types it contains. The remaining three channels encode front and bottom wire density and via density. All count-based channels are logarithmically scaled to retain sensitivity in sparse regions, while the front and bottom channel pairs share the same normalization to preserve layer symmetry. After the channel-wise Laplacian sharpening and the CNN-based model described in Section 3.2, this branch yields the routing feature $\mathbf{R}$, which is fused with $\mathbf{P}$ into the normalized embedding e to support the subsequent performance-aware supervised training.

D BO-BASED DATASET PREPARATION SCHEME

Many expert-designed PCB layouts are available without the scripts used to generate them, resulting in a lack of such layout-script pairs that can be directly used for LLM training. To address this limitation, we propose a Bayesian Optimization (BO) based dataset preparation scheme to align layouts and scripts. Given a reference layout, BO searches for an executable PCB PnR script whose generated layout is both physically efficient and geometrically similar to the reference layout.

Let $x = (\mathbf{z}, \boldsymbol{\theta})$ denote a script configuration, where $\mathbf{z}$ specifies the implementation of PnR APIs and $\boldsymbol{\theta}$ specifies their parameter values. Executing the script instantiated by x produces a layout with wirelength $\mathrm{WL}(x)$, via count $\mathrm{Via}(x)$, and embedding $e(x)$. Specifically, the objective of the BO process is to minimize the following objective:

$$f(x) = \alpha \cdot \left| \frac{\mathrm{WL}(x)}{\mathrm{WL}^g + \delta} - 1 \right| + \beta \cdot \left| \frac{\mathrm{Via}(x)}{\mathrm{Via}^g + \delta} - 1 \right| - \gamma \cdot \mathrm{sim}(e(x), e^g)\,, \tag{12}$$

where WL^g and Via^g denote the wirelength and via count of the reference layout, respectively, and $\delta > 0$ is a small constant preventing division by zero. e^g represents the embedding of the reference layout and $\mathrm{sim}(\cdot, \cdot)$ denotes an embedding-similarity measure by RePCB, as introduced

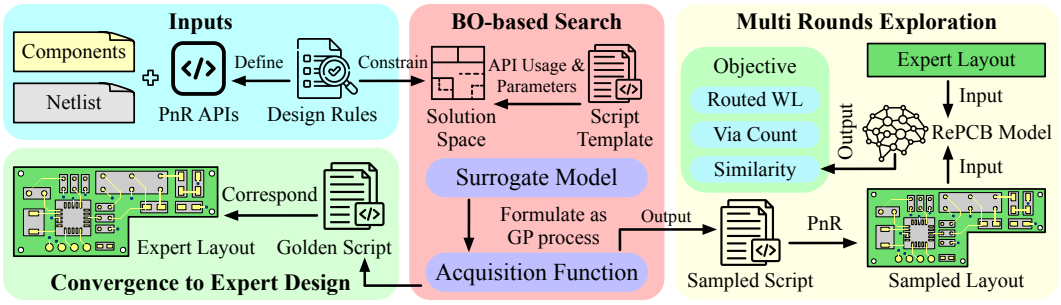


Figure 7: Overall flow of BO-based search for layout-script dataset preparation.

in Section 3.2. The hyperparameters α, β, and γ control the trade-off among the three terms. The first two terms measure the relative deviation of the generated layout from the reference layout in wirelength and via count, so that all three terms are dimensionless and of comparable scale. The last term encourages BO to find scripts whose generated layouts are topologically similar to the expert-designed PnR results.

First, we assume the objective function $f(x)$ follows a Gaussian Process (GP) prior (MacKay et al., 1998; Seeger, 2004), as shown in Equation (13). With the GP as the surrogate model, the objective function can be learned with a finite number of evaluations to enhance efficiency.

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')) \,, \tag{13}$$

where $m(x)$ is the prior mean function and $k(x, x')$ is the covariance kernel. For the hybrid search space considered in EvoPCB, the kernel accounts for both the discrete API-invocation variables and the continuous or discrete API parameters. Specifically, given the observations $\mathcal{D}_n = \{(x_i, y_i)\}_{i=1}^n$, where $y_i = f(x_i) + \epsilon_i$ is the observation result of the script x_i, $\epsilon_i \sim \mathcal{N}(0, \sigma_\epsilon^2)$ is a random signal governed by Gaussian distribution. And the posterior predictive distribution at a candidate configuration x is $f(x) \mid \mathcal{D}_n \sim \mathcal{N}\big(\mu_n(x), \sigma_n^2(x)\big)$. The posterior mean and variance are given by:

$$\mu_n(x) = m(x) + \mathbf{k}(x)^T \left(\mathbf{K} + \sigma_\epsilon^2 \mathbf{I}\right)^{-1} (\mathbf{y} - \mathbf{m}) \,, \tag{14}$$

$$\sigma_n^2(x) = k(x, x) - \mathbf{k}(x)^T \left(\mathbf{K} + \sigma_\epsilon^2 \mathbf{I}\right)^{-1} \mathbf{k}(x), \tag{15}$$

where $K_{ij} = k(x_i, x_j)$, $\mathbf{k}(x) = [k(x_1, x), \ldots, k(x_n, x)]^T$, and $\mathbf{m} = [m(x_1), \ldots, m(x_n)]^T$. The parameter σ_ϵ^2 models observation noise and also improves numerical stability. When a zero-mean GP is assumed, the $m(x)$ and $\mathbf{m}$ terms can be omitted.

As for the acquisition function, we select Expected Improvement (EI) (Jones et al., 1998; Zhan & Xing, 2020), because it balances exploitation of configurations with low predicted objective values and exploration of uncertain solution space. For the minimization problem, let $f_{\text{best}} = \min_{1 \le i \le n} y_i$ denote the best observed objective value. The EI acquisition function is defined in Equation (16):

$$\text{EI}(x) = \mathbb{E}\left[\max(f_{\text{best}} - f(x) - \xi, 0)\right] , \tag{16}$$

where $\xi \ge 0$ is the exploration parameter. Because the GP prediction is a probability distribution rather than a deterministic value, the expectation incorporates both the posterior mean and predictive uncertainty. In closed form, Equation (16) can be represented as Equation (17):

$$\text{EI}(x) = \left(f_{\text{best}} - \mu_n(x) - \xi\right) \Phi(u) + \sigma_n(x) \phi(u), \tag{17}$$

where

$$u = \frac{f_{\text{best}} - \mu_n(x) - \xi}{\sigma_n(x)}, \tag{18}$$

and $\Phi(\cdot)$ and $\phi(\cdot)$ are the cumulative distribution function and probability density function of the standard normal distribution. When $\sigma_n(x) = 0$, the acquisition function $\text{EI}(x)$ is set to zero.

However, a direct search over all API invocations and parameter values may produce incomplete, conflicting, or otherwise invalid scripts. We therefore utilize customization to restrict BO to a feasible script space. Let $\mathcal{A}$ be the set of candidate APIs, partitioned into a set of individually mandatory

APIs $\mathcal{A}_{\text{req}}$ and a set of optional APIs $\mathcal{A}_{\text{opt}} = \mathcal{A} \setminus \mathcal{A}_{\text{req}}$. In EvoPCB, A*-Route is individually mandatory, whereas SA-Place, Diff-Place, and MILP-Place are alternative implementations of the mandatory placement stage, from which exactly one must be selected. For each API $a \in \mathcal{A}$, let $z_a \in \{0, 1\}$ indicate whether API a is invoked in the current PnR script. Each individually mandatory API must satisfy $z_a = 1, \forall a \in \mathcal{A}_{\text{req}}$. For an invoked API, its parameter vector must lie within its legal domain, and if an optional API is inactive, its parameters are assigned a distinguished null value $\theta_a^{\emptyset}$, as shown in Equation (19) and Equation (20), respectively:

$$z_a = 1 \Longrightarrow \theta_a \in \Theta_a, \quad \forall a \in \mathcal{A} \tag{19}$$

$$z_a = 0 \Longrightarrow \theta_a = \theta_a^{\emptyset}, \quad \forall a \in \mathcal{A}_{\text{opt}} \tag{20}$$

This conditional representation prevents parameters associated with inactive APIs from affecting script execution or introducing spurious search dimensions.

To ensure that the BO search PnR scripts function correctly and generate valid PCB layouts, we also propose several design constraints to the customized BO. Specifically, in EvoPCB, the three alternative placement APIs form the mutually exclusive group $G_{\text{place}} = \{\text{SA-Place}, \text{Diff-Place}, \text{MILP-Place}\}$ because invoking multiple placement algorithms in a single stage could produce conflicting or ambiguous behavior. Let $\mathcal{M} \subseteq 2^{\mathcal{A}}$ be the collection of mutually exclusive API groups. For every group $G \in \mathcal{M}$, at most one API may be invoked:

$$\sum_{a \in G} z_a \leq 1, \quad \forall G \in \mathcal{M}. \tag{21}$$

Some mutually exclusive API groups may implement mandatory PnR stages. Let $\mathcal{S}_{\text{req}} \subseteq 2^{\mathcal{A}}$ denote the collection of API groups corresponding to mandatory stages, such as initialization, design parsing, placement, routing, and layout export. At least one API must be selected from each mandatory-stage group to ensure the PnR design can be correctly generated:

$$\sum_{a \in S} z_a \geq 1, \quad \forall S \in \mathcal{S}_{\text{req}} \tag{22}$$

Furthermore, because G_{place} is both a mandatory-stage group and a mutually exclusive group, Equations (21) and (22) jointly requires exactly one placement API to be selected, while $z_{\text{A*-Route}} = 1$ independently requires A*-Route. This distinction avoids incorrectly treating every alternative implementation of a mandatory stage as an individually mandatory API.

At the same time, within the PnR scripts of EvoPCB, certain APIs have dependencies on others; for instance, placement-related functions must be completed before attempting routing. API dependencies are represented by a set $\mathcal{D}$ of ordered pairs. For each $(a, b) \in \mathcal{D}$, API b depends on API a, and therefore:

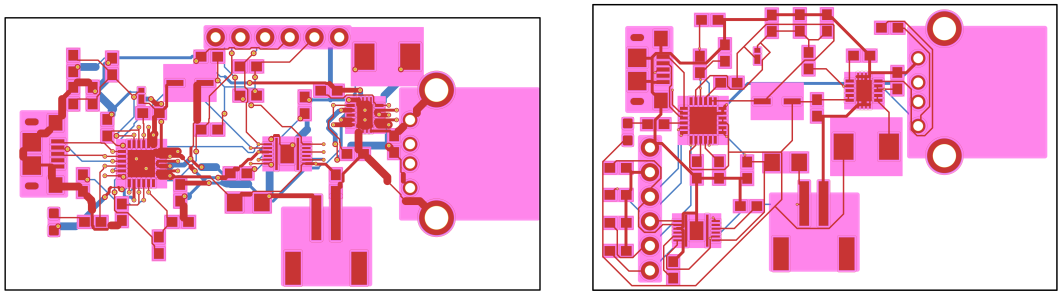
$$z_b \leq z_a, \quad \forall (a, b) \in \mathcal{D}. \tag{23}$$

These constraints ensure that every candidate PnR script contains all mandatory APIs and PnR stages, uses legal parameter values, avoids incompatible API combinations, and satisfies all API dependencies, so that the corresponding PCB layouts can be generated correctly. Under these constraints, the solution space of our customized BO can be represented by $\mathcal{X}$.

Since our BO process involves only API calls and parameter settings, we specifically introduced a mask to define the search range for the BO. During acquisition optimization, feasibility is enforced using the following hard mask:

$$\text{EI}_{\mathcal{X}}(x) = \begin{cases} \text{EI}(x), & x \in \mathcal{X}, \\ -\infty, & x \notin \mathcal{X}. \end{cases} \tag{24}$$

Infeasible configurations cannot be selected in our customized BO-based search process. In implementation, $-\infty$ can be realized by rejecting infeasible candidates or assigning them a sufficiently large negative acquisition value. In the BO-based search process, the next script configuration is selected by solving the equation $x^* = \arg\max_{x \in \mathcal{X}} \text{EI}(x)$. However, the above constraints only guarantee the validity of the PnR script, but the search space is still quite large. Therefore, to improve search efficiency, we perform acquisition optimization hierarchically. First, feasible API-invocation patterns are generated subject to the mandatory-stage, mutual-exclusion, and dependency



(a) GPT-6 Astra (b) Our proposed EvoPCB

Figure 8: Layout comparison for case P3. Red and blue denote wires on the front and bottom copper layers, respectively. Small yellow circles represent vias. The layout designed by GPT-6 Astra contains more vias and longer parallel wire segments around the main IC, whereas EvoPCB produces a more reasonable routing topology. Quantitative results are reported in Table 2.

Table 6: Benchmark statistics.

Case	#Components	#Fixed	#Pins	#Nets	Boundary (mm)	#Layers	Design Complexity
P1	62	4	231	81	55×55	2	High
P2	49	1	161	54	41×40	2	High
P3	35	1	138	38	55×28	2	Medium
P4	64	3	312	63	86×71.5	4	High
P5	60	2	233	35	58×59.5	4	High
P6	46	3	113	40	102×60	2	Low
P7	72	4	167	52	64×55	2	Medium
P8	55	6	129	43	100×55	2	Low
P9	48	2	105	21	97×15.5	2	Medium
P10	97	4	214	61	98×55	2	Medium
P11	71	6	176	57	90×59	2	Low

constraints. For each feasible invocation pattern, only the parameters of active APIs are then optimized within their corresponding domains. This constrained optimization strategy prevents evaluations from being wasted on invalid scripts and reduces the effective dimensionality of the hybrid search space. Based on the above settings, the BO process can effectively and efficiently search for reference PnR scripts to generate a reference layout-script dataset that supports the two-stage optimization of EvoPCB, thereby significantly improving the overall design quality and performance.

E Supplementary Experiment

E.1 Additional Experiment Settings

We implement EvoPCB in Python, PyTorch, and Hugging Face Transformers for LLM inference and training. All random seeds are set to 2026. In the GraphSAGE-based placement feature extraction, the message passing depth D in Algorithm 2 is 6. As for RePCB, the nearest-neighbor parameter Z is set to 6, and the resolution of the routing image is fixed at 128×128. The Laplacian sharpening strength η of Equation (2) is set to 0.5, and the weights of the regression loss $\mathcal{L}_{\text{reg}}$ and similarity loss $\mathcal{L}_{\text{sim}}$ are 1.2 and 0.7, respectively. The temperature T in Equation (4) is 1.0. The batch size of the supervised RePCB model is 16, and the learning rate is 3×10^{-4}. In the BO-based dataset preparation, the maximum number of search iterations is 300. The nugget of the kernel function is set to 1×10^{-5}, and the candidate pool of each iteration contains 256 script configurations. K of the beam search in MM-Evolve is set to 6. The temperature of inference models is set to 0.3 and 0.7 in the SFT and DAPO stages, respectively. Specifically, we use 2341 and 5741 expert-designed PCB layouts for the SFT and DAPO stages, respectively. For benchmarks, we select 11 representative test cases with different levels of design complexity, none of which appeared in the training

dataset. The design complexity presented in Table 6 was assessed by human experts, primarily based on factors such as the proportion of layout area occupied by components, connectivity within the netlist, and specific design constraints. The training and testing datasets are independent; the data used for training does not appear in either the testing phase or the benchmarks to ensure overall transferability.

E.2 Detailed Comparison with GPT-6 Astra

Upon the release of GPT-6 Astra, OpenAI claimed that its physical PCB design capabilities could partially replace expert engineers (OpenAI, 2026). Therefore, we selected it as a baseline. To complement the quantitative evaluation, Figure 8 compares the P3 layouts produced by GPT-6 Astra and our EvoPCB. We carefully designed the prompt for GPT-6 Astra to allow it to perform to its full potential. The complete prompt used for GPT-6 Astra is provided in the Prompt block.

Prompt for generating PCB layouts with GPT-6 Astra

Role: You are an expert physical design engineer specializing in PCB placement and routing.

Task: I have uploaded a KiCad PCB file. I need you to complete the component placement first and then wire the routing for this board without the schematic.

Strict Constraints:

1. Fixed Layer Usage: Do not modify the existing number of layers of the board under any circumstances.
2. Sequential Workflow: You must finalize all component placement before initiating any wire routing. During the routing process, you can slightly change the placement to optimize the routing.
3. 45-Degree Routing Only: Strictly limit all wire routing to 45-degree angles. Do not use 90-degree corners or arbitrary angles.
4. Schematic-Free Execution: You must complete the PCB layout relying solely on the connections and netlist embedded within the provided PCB file. Do not request an external schematic.

Optimization Goals: Please use your intelligence and knowledge of best practices to optimize this layout for electronics performance and thermal performance. At the same time, complete the design using minimal wirelength and the fewest possible vias.

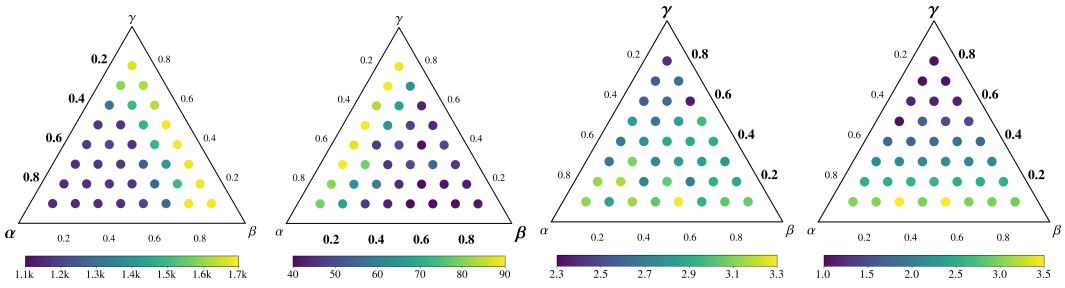
Deliverables: (1) The fully component-placed and routed KiCad PCB file. (2) A clear and short design log. Don't be too long-winded.

In the GPT-6 Astra layout, power and signal wires are concentrated in several local regions, with long parallel segments and many vias around the main IC. The generated wires also exhibit abnormal and substantial width variation. In contrast, EvoPCB produces a more compact topology with fewer vias and less extensive parallel routing. These qualitative observations are consistent with the quantitative P3 results in Table 2. Based on the logs of GPT-6 Astra, its approach to physical design primarily involves invoking open-source PCB PnR tools, including SA-PCB (OpenROAD, 2019d), PcbRouter (OpenROAD, 2019c), FreeRouting (Fuchs, 2026), etc. However, it cannot integrate expert experience or preferences, relying instead on a blind design process driven solely by its understanding of natural language descriptions and layout images. GPT-6 Astra evidently relies on these open-source tools for blind iterative design; while this ensures DRC- and LVS-clean results, the resulting EM and thermal performance is notably poor.

In the experiment, we observed that GPT-6 Astra exhibits poor stability and reproducibility in PCB physical design. GPT-6 Astra cannot guarantee a correct result on the first attempt; for P10, it took two attempts to produce a valid output. An analysis of design logs reveals that GPT-6 Astra does not truly understand physical spatial relationships between components and nets; instead, it achieves designs primarily by invoking open-source repositories and performing iterative parameter tuning. While utilizing open-source code as foundational tools ensures the validity of the resulting designs, this approach is highly inefficient and struggles to identify optimal solutions. Although GPT-6 Astra is one of the current SOTA LLMs, our EvoPCB is a framework specifically designed and optimized for performance-driven PCB physical design, implementing a two-stage specialization process and

Table 7: Results of the prediction errors on PCB-related metrics for the proposed RePCB and the baseline representation methods. The four metrics correspond to their respective MSE results.

Model	Wirelength	Via Count	$\mathcal{S}_{\text{EM}}$	$\mathcal{S}_{\text{Th}}$	$\overline{\text{MSE}}$	$\overline{\text{MAE}}$
RePCB (Ours)	**0.821**	**0.856**	**1.104**	**0.948**	**0.932**	**0.733**
GPDL (Liu et al., 2021)	0.968	1.021	1.308	1.154	1.113	0.858
DeepLayout (Zhao et al., 2025)	1.162	1.246	1.623	1.437	1.367	1.026
Circuit GNN (Yang et al., 2022)	1.410	1.534	2.564	2.146	1.914	1.271



(a) Wirelength (b) Via Count (c) $\mathcal{S}_{\text{EM}}$ (d) $\mathcal{S}_{\text{Th}}$

Figure 9: Hyperparameter analysis. The four subfigures show the relationships between α, β, and γ and their corresponding roles in optimizing the four performance metrics of generated layouts.

MM-Evolve to significantly enhance PCB layout quality. As shown in Table 2, it achieves performance that clearly surpasses GPT-6 Astra.

E.3 Implementations of Multi-physics Costs

Because the raw simulation results differ in both scale and units, we convert them into two dimensionless, direction-aligned costs according to the industrial requirements. For EM performance, insertion loss (IL) and crosstalk (XT) are both measured in decibels, but they occupy different physical ranges: IL is naturally referenced to a lossless channel, whereas XT is referenced to a coupling budget. In our convention, we report IL as a positive magnitude in dB, where larger values indicate more loss, whereas XT is negative in dB, where smaller (more negative) values indicate weaker crosstalk. We therefore measure each metric against a fixed reference and sum the two offsets, as shown in Equation (25):

$$\mathcal{S}_{\text{EM}} = (\text{IL} - \text{IL}_{\text{ref}}) + (\text{XT} - \text{XT}_{\text{ref}}), \tag{25}$$

with $\text{IL}_{\text{ref}} = 0\,\text{dB}$, the ideal lossless reference, and $\text{XT}_{\text{ref}} = -40\,\text{dB}$, the crosstalk budget commonly adopted in industrial-grade PCB physical design (Bogatin, 2021). Both terms thus grow monotonically with degradation, so $\mathcal{S}_{\text{EM}}$ is the total offset relative to the reference combination and is smaller-is-better. Since XT is negative in dB, layouts whose crosstalk is suppressed below the $-40\,\text{dB}$ budget yield a negative offset; consequently, a negative $\mathcal{S}_{\text{EM}}$ indicates that the layout performs better than the reference budget.

For thermal performance, the peak temperature $\mathcal{T}_{\max}$ and the peak temperature-gradient magnitude $||\nabla\mathcal{T}||_{\max}$ carry different units and cannot be summed directly. We therefore normalize each by its corresponding admissible budget and add the resulting dimensionless results:

$$\mathcal{S}_{\text{Th}} = \frac{\mathcal{T}_{\max} - T_{\text{amb}}}{T_{\text{ref}} - T_{\text{amb}}} + \frac{||\nabla\mathcal{T}||_{\max}}{G_{\text{ref}}}, \tag{26}$$

where $T_{\text{amb}} = 25\,^{\circ}\text{C}$ is the standard ambient temperature in the PCB design domain and $T_{\text{ref}} = 125\,^{\circ}\text{C}$ is a typical maximum junction temperature of industrial-grade PCBs (Bogatin, 2021), so that the denominator $T_{\text{ref}} - T_{\text{amb}} = 100\,^{\circ}\text{C}$ represents the admissible temperature-rise budget. The reference gradient $G_{\text{ref}} = 30\,^{\circ}\text{C/mm}$ is chosen around the magnitude at which thermal-mechanical stress becomes a practical reliability concern (Bogatin, 2021; 2010; Mitzner, 2009), and it matches the typical gradient magnitude observed across our benchmarks. Each term measures the fraction of

the corresponding budget consumed by a layout, which makes the two terms comparable, dimensionless, and smaller-is-better, allowing them to be combined into the unified cost $\mathcal{S}_{\mathrm{Th}}$ in Equation (26).

In our experiments, all PCBs are simulated using the same evaluation protocol. As for simulation implementations, we evaluate EM performance with openEMS (Liebig, 2026), a finite-difference time-domain (FDTD) solver. Each board is solved from 100 MHz to 3 GHz with $50\,\Omega$ lumped ports. The background mesh is 0.5 mm with additional mesh lines inserted along wire boundaries, smoothed with a growth ratio of 1.5, and terminated by a 4-cell PML. Through EM simulation, we can obtain the IL and XT values at the PCB operating frequency and calculate $\mathcal{S}_{\mathrm{EM}}$. We solve steady-state heat conduction with OpenFOAM (OpenCFD, 2026) using the `laplacianFoam` solver on a 0.5 mm uniform grid. Free-air convection is applied on the top and bottom surfaces with $h = 10$ W/(m^2·K), the ambient temperature is $25\,^\circ$C, and the board edges are insulated. From the resulting temperature map, we report the peak temperature $\mathcal{T}_{\max}$ and the peak temperature-gradient magnitude $\|\nabla\mathcal{T}\|_{\max}$, normalized into $\mathcal{S}_{\mathrm{Th}}$ as in Equation (26).

E.4 Layout Representation

As for representation baselines, we select the existing SOTA layout representation frameworks in EDA: 1) GPDL (Liu et al., 2021): a well-known CNN-based representation model used for macro PnR. 2) DeepLayout (Zhao et al., 2025): a general layout representation framework through mask strategies. 3) Circuit GNN (Yang et al., 2022): a customized GNN framework that fuses both topological and geometrical features for circuit layouts. To ensure fairness, we use the exact same MLP-based model for prediction on the embeddings output by all baselines and our RePCB.

Since the PCB layout representation is proposed to calculate the similarity between expert-designed and LLM-generated layouts, we first evaluate the performance of RePCB on layout representation. Specifically, we design a comprehensive prediction task as a downstream task on PCB layouts, where the wirelength, via count, $\mathcal{S}_{\mathrm{EM}}$, and $\mathcal{S}_{\mathrm{Th}}$ form a four-element vector to be predicted. Each target is standardized using the mean and standard deviation computed on the training set, and the reported errors are measured on this standardized scale so that the four metrics are directly comparable. Note that wirelength and via count are the supervised targets used to train RePCB, whereas $\mathcal{S}_{\mathrm{EM}}$ and $\mathcal{S}_{\mathrm{Th}}$ are physical quantities never seen during representation training; reporting them separately verifies that the learned embedding generalizes beyond its own supervision.

We compare the Mean Squared Error (MSE) and Mean Absolute Error (MAE) of RePCB with those of other baselines. Table 7 reports the per-metric MSE for the four targets together with the overall average MSE and MAE. RePCB consistently achieves the lowest error on every individual metric across all three state-of-the-art representation frameworks. In terms of overall MSE, RePCB achieves reductions of 16.3% over GPDL (Liu et al., 2021), 31.8% over DeepLayout (Zhao et al., 2025), and 51.3% over Circuit GNN (Yang et al., 2022). For overall MAE, RePCB improves by 14.6%, 28.6%, and 42.3%, respectively. Notably, the advantages hold not only on the two supervised targets but also on the two unseen multi-physics targets $\mathcal{S}_{\mathrm{EM}}$ and $\mathcal{S}_{\mathrm{Th}}$, confirming that the embedding captures generalizable layout structure rather than merely memorizing the training labels.

E.5 Hyperparameter Analysis

As introduced in Section 3, α, β, and γ are used to balance the three optimization objectives: wirelength, via count, and the similarity between reference designs, while calculating the objective function (Equation (12)) in BO and the advantage function (Equation (8)) in the customized DAPO. In all the experiments in Section 4, α, β, and γ are set to 0.4, 0.4, and 0.2, respectively. To further analyze these hyperparameters and understand their respective roles, we vary their values within specified ranges in the experiment. As shown in Figure 9, it is clear that the advantage becomes more pronounced as the parameters corresponding to a metric increase. However, because α, β, and γ are inversely related, the performance of the other two objectives typically decreases to varying degrees. Since α, β, and γ represent the importance of three objectives, and these objectives may have conflicting relationships, adjusting their values can change the direction of optimization. Notably, since the EM performance is coupled with multiple factors; consequently, its optimization is not particularly stable. However, increasing γ still improved the overall $\mathcal{S}_{\mathrm{EM}}$. This is because EvoPCB performs EM-optimized PCB physical design by learning expert design experience and preferences through the proposed two-stage specialization.