PCB-Migrator: Automated PCB PnR Migration

Yaohui Han[1], Beichen Li[2], Rongliang Fu[1], Qunsong Ye[3], Zhiyuan Lu[2], Junchen Liu[2], Bei Yu[1], Tsung-Yi Ho[1], Tinghuan Chen[2,*]

[1]CUHK [2]CUHK-Shenzhen [3]Index Tech

***Abstract*—Despite the availability of numerous frameworks and tools for automated PCB placement and routing, the industry still relies heavily on expert designers to ensure layout reliability and performance. However, when design requirements change, such as adjustments to board dimensions or the addition of new obstacles, experts must often recreate similar layouts from scratch, leading to substantial inefficiencies in both time and resources. To address this challenge, we introduce PCB-Migrator, an automated framework for PCB layout migration. Our approach leverages an offset constraint graph to capture positional relationships among components in the referenced design and effectively map them onto the new PCB. Additionally, PCB-Migrator builds routing path graphs to extract routing characteristics from the reference layout and applies graph matching to guide the routing process on the new board. Experimental results demonstrate that PCB-Migrator outperforms existing baselines, achieving faster runtimes while preserving the key design characteristics and performance of the referenced PCB.**

I. Introduction

Printed circuit board (PCB) plays a critical role in almost all industrial applications. As electronic products have become more versatile and complex, the number of nets and pins on modern PCBs is significantly increasing. However, designing these complicated PCBs presents significant challenges, which involve the analysis of the power and signal integrity [1], as well as many specific design rules for the topology of placement and routing (PnR). After each layout is completed, it must be examined by simulation to ensure its performance [2], resulting in a timing-consuming design process. Therefore, it is critical to develop effective tools for automated PCB design, especially for physical PnR.

The challenges of automated PCB PnR mainly lie in the complex design space and the lack of effective early evaluation. Specifically, each component can be rotated and placed on both sides of the board. The shape of the board frame and the components are often irregular. In addition to the top and bottom layers, there are more mid-layers (up to 30) that can be used for routing, which also significantly increases the design space. Meanwhile, most automated frameworks perform PnR separately. Thus, it is difficult to foresee the subsequent routing at the early placement stage only through conventional metrics like HPWL. This leads to poor routability and performance, and even makes it impossible to get a feasible solution.

To solve the above challenges, various frameworks have been developed for automated PCB PnR design [3]–[21], typically integrating multiple design constraints with optimization objectives. OpenROAD [22] has open-sourced PCB-PR-App [17], which is composed of SA-PCB [19] and PcbRouter [18], respectively solving the PCB PnR problem. NS-Place [7] minimized net congestion using a support vector machine-like formulation and performed legalization by solving a routability-aware Mixed-Integer Linear Programming (MILP). HiePlace [11] proposed a heuristic algorithm to optimize the placement, thereby reducing manufacturing complexity. These methods only use traditional metrics to optimize PnR, which is far from the expert level.

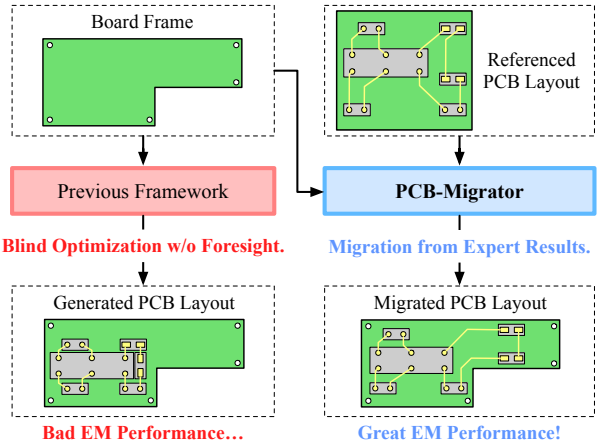


Fig. 1 Our PCB-Migrator compared with previous automated frameworks that design PCBs from scratch. Migration from expert results enables PCB-Migrator to have foresight in the early stage, thus enhancing its practical performance.

Through data-driven approaches, some studies incorporate historical experience into AI models to enhance PCB PnR design. DeepPCB [20] is a powerful commercial tool for implementing PCB PnR using reinforcement learning, aiming to improve the performance of industrial PCBs. PCBAgent [15], a framework combining LLM and RL agents, is proposed to enhance PCB placement. TRouter [5] is a thermal-driven routing framework via a machine learning model. However, these methods heavily rely on the accuracy of the performance model or require large amounts of diverse training data, resulting in very slow optimization efficiency and poor transferability.

In practice, all the above automatic frameworks usually fail to meet the specific requirements of real-world projects. The previous "design-from-scratch" frameworks only use some approximate estimation indicators, such as HPWL at placement and cost in the path search algorithms at routing. This method may have the ability to find feasible PnR solutions, but it falls short of meeting the industrial-grade requirements, with poor performance and reliability. In addition, these automatic frameworks generally lack the ability to foresee potential routing problems, which significantly limits the design quality. Therefore, it is almost impossible for existing "design-from-scratch" frameworks to obtain PCB

This work is supported by the National Key Research and Development Program of China (No. 2023YFB4402900), the National Natural Science Foundation of China (No. 92573108, 62304197), and Shenzhen Index Tech. This paper was conducted in the JC STEM Lab of Intelligent Design Automation funded by The Hong Kong Jockey Club Charities Trust.

*Corresponding author.

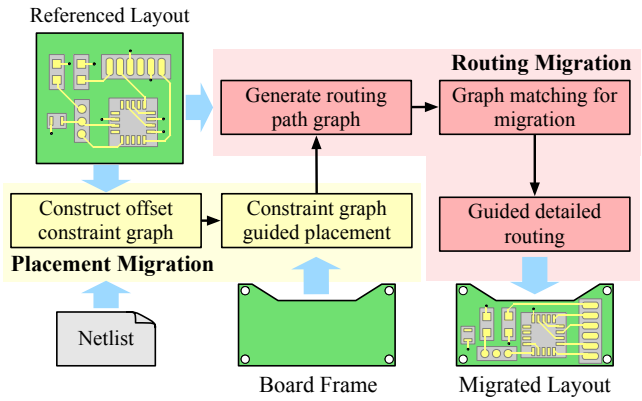


Fig. 2 The overall flow of PCB-Migrator.

layouts that can rival those of expert designs.

Migration is a potential method to address the above limitations and bridge the gap between expert design and actual requirements. As shown in Fig. 1, when design requirements change, such as adjustments to board dimensions or the addition of new obstacles, using existing expert designs as references for migration can achieve comprehensive higher performance, especially in electromagnetic (EM) performance, which can reach expert-level. EM performance directly determines the actual usability of PCB in the industry. Therefore, this paper performs automatic migration to enhance the practical usability and performance of PCBs to expert-level design. Overall, our contributions are as follows:

1) To our best knowledge, this paper first proposes a PCB PnR migration framework named PCB-Migrator to enhance the performance of the migrated PCB layouts.
2) We propose the offset constraint graph to preserve the design characteristics and guide the PCB placement migration. We also design a relaxation and legalization strategy to improve the efficiency significantly.
3) To preserve the routing characteristics, we propose the routing path graph and a representation of neighborhood connections to perform routing migration.

II. Preliminaries

A. PCB PnR Migration

Different from previous automatic frameworks, PCB PnR migration is a new technology that extracts placement and routing characteristics from referenced PCB layouts and applies them to the new designs. This approach not only efficiently achieves design closure but also ensures practical usability and expert-level performance, compared to conventional "design-from-scratch" frameworks.

B. Problem Formulation

In PCB migration, the new board frame may significantly differ from that of the original PCB and may contain additional obstacles. The netlist of the migrated PCB must remain identical to that of the reference. We aim to ensure that the performance and design characteristics of the newly migrated PCB layout are close to those of the referenced layout through PnR migration. Our problem formulation is formally defined as follows:

Problem 1 (*PCB PnR migration*). Given a referenced PCB layout and a modified board frame containing obstacles, the goal is to generate a new PCB layout that preserves the original PnR characteristics to maintain the performance of the referenced PCB.

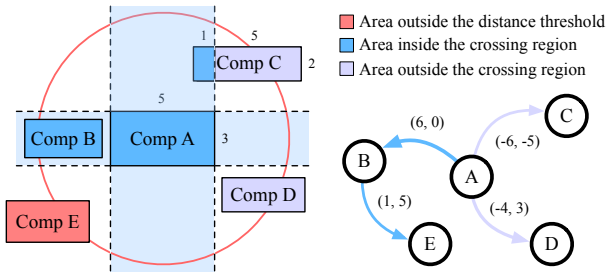


Fig. 3 Example of offset constraint graph construction. Assuming component C is within the distance threshold of component A, an edge exists between components A and C. The edge weight is the offset of their coordinates, *i.e.*, $(-6, -5)$. The overlapping area of component A's crossing region and component C is 2×1. Component C has a smaller area 2×6 than component A. So the edge weight is calculated as $e^{-\sqrt{(-6)^2+(-5)^2}} + \mu \cdot \frac{2\times1}{2\times6}$.

III. Algorithm

This section proposes PCB-Migrator, as shown in Fig. 2. In placement migration, most design characteristics can be reflected in the relative position relationship with weights. Therefore, PCB-Migrator constructs the offset constraint graphs and formulates the placement into an MILP to preserve the placement characteristics. As for routing, because each component may have positional variations, PCB-Migrator constructs rotation-invariant routing path graphs for both the referenced and the new placement. It then performs graph matching to preserve the original routing characteristics, guiding detailed routing to complete the PCB PnR migration.

A. Placement Migration

Offset Constraint Graph (OCG) Construction. In placement migration, it is vital to extract and preserve the positional relationships between components in the referenced PCB. These relationships significantly influence the subsequent routing and directly affect the performance of the migrated PCB. However, the conventional mixed constraint graph (MCG) [23], [24] only extracts approximate relations between components, which cannot effectively guide placement. Furthermore, in the MCG, positional constraints are limited to only between two adjacent components, which severely limits its effectiveness in PCB placement migration.

Compared to the MCG, our proposed offset constraint graph includes more specific information on the edges. For each component, edges are constructed for all other components within the distance threshold, along with their coordinate offsets and calculated weights. The weights of edges are defined as the offsets of different components' coordinates. For all components within the distance threshold, the weights of the edges are calculated as

$$\mathcal{W}_{ij} = e^{-d_{ij}} + \mu \cdot \frac{S_{ij}}{\min(S_i, S_j)}, \tag{1}$$

where d_{ij} represents the Euclidean distance between components i and j. S_{ij} represents the overlapping area of component i's crossing region and component j, as shown in Fig. 3. μ is a balance factor. This setting means that there

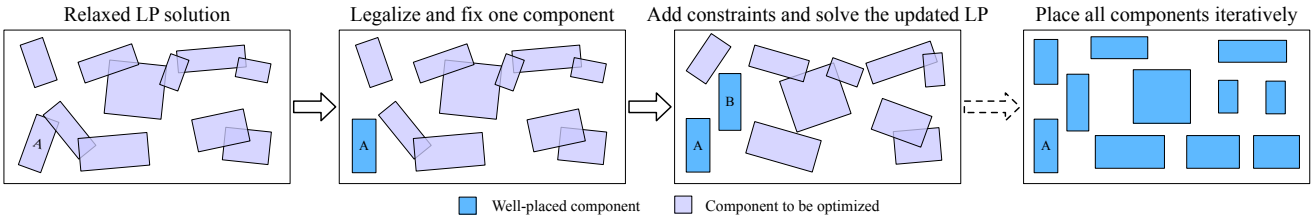


Fig. 4 Relaxation and legalization iteratively to speed up the placement migration.

will be stronger constraints between components with absolute horizontal or vertical position relationships, and closer components have stronger positional constraints. Compared to the conventional MCG, the proposed OCG is more suitable for extracting the complicated positional relationships between components.

Offset Constraint Graph-Guided Placement. With the OCG, PCB-Migrator will get the placement migration result based on the values and weights of positional constraints. Additionally, the total wirelength, approximated as HPWL, is incorporated as another objective. Because continuous coordinates (x, y) and discrete integer angles θ of all components need to be solved simultaneously, the placement migration is formulated as an MILP:

$$\begin{aligned}\min \sum_{i=1}^{|C|}\sum_{j=i}^{|C|} \mathcal{W}_{ij} \cdot (|\Delta x_{ij}^{o} - \Delta x_{ij}^{n}| + |\Delta y_{ij}^{o} - \Delta y_{ij}^{n}|) \\ + \gamma_1 \cdot \sum_{i=1}^{|C|} \Delta\theta_i + \gamma_2 \cdot \sum_{m=1}^{|net|} \text{HPWL}(x, y, \theta), \qquad (2)\end{aligned}$$

$$\text{s.t. } \theta_i \in \{0°, 90°, 180°, 270°\}, \ \forall i \in C, \qquad (3)$$

where $\gamma_{1,2}$ are hyperparameters that balance the relative importance of placement variation and HPWL. For component i and j, the offset variations before and after migration are denoted by $(\Delta x_{ij}^{o}, \Delta y_{ij}^{o})$ and $(\Delta x_{ij}^{n}, \Delta y_{ij}^{n})$, respectively. $\Delta\theta_i$ denotes the rotation variation of component i. The objective minimizes offset variations between components, rotation variations, and HPWL, so as to retain the original placement characteristics as much as possible in the modified board frame with obstacles.

$$x_i^{proj} = |l_i \cdot cos(\theta_i)| + |w_i \cdot sin(\theta_i)|, \qquad (4)$$

$$y_i^{proj} = |l_i \cdot sin(\theta_i)| + |w_i \cdot cos(\theta_i)|, \qquad (5)$$

$$|\Delta x_{ij}| \geqslant \frac{x_i^{proj} + x_j^{proj}}{2} + s, \qquad (6)$$

$$|\Delta y_{ij}| \geqslant \frac{y_i^{proj} + y_j^{proj}}{2} + s, \qquad (7)$$

where s denotes the minimum spacing required between components. To component i, x_i^{proj} is the projection on the x-axis, and l_i and w_i are the length and width. The above constraint ensures that each component cannot overlap with any other component or obstacle.

$$d + \frac{x_i^{proj}}{2} \leqslant x_i \leqslant X - d - \frac{x_i^{proj}}{2}, \qquad (8)$$

$$d + \frac{y_i^{proj}}{2} \leqslant y_i \leqslant Y - d - \frac{y_i^{proj}}{2}, \qquad (9)$$

where d denotes the minimum spacing required between the component and the boundary. As for the board frame, the lower-left corner is located at $(0, 0)$ and the upper-right corner is located at (X, Y). The above constraint ensures that each component is placed within the board frame.

However, the efficiency of directly using the solver to solve the above MILP problem is very low. To accelerate the placement migration while ensuring performance, we proposed a relaxation and legalization strategy. PCB-Migrator initiates the placement migration by relaxing the angle constraint (3), converting the angles into continuous variables, that is $\theta_i \in [0°, 360°]$. So the MILP is converted to several LP problems. These relaxed LPs allow partial overlap between components and continuous rotation angles in the preliminary placement. In the LP solution, PCB-Migrator identifies the unplaced component with the smallest x- and y-coordinates. During the legalization process, for the current component, a hybrid grid search combining coarse and fine steps is then performed around the position solved by the LP to identify a feasible placement. For each candidate position, all four possible rotation angles are attempted. After a valid position and angle are found, both the position and angle are fixed and added as extra constraints to the original LP model. This process is repeated iteratively until all components are well placed, as shown in Fig. 4.

With the help of the above relaxation and legalization strategy, the solution speed of PCB-Migrator in the placement stage is significantly improved, while the placement results achieve high-quality approximations of the global optimal solutions, which will be shown in detail in the experimental section.

B. Routing Migration

Routing Path Graph Generation from Placement. Due to changes in the PCB boundary and the addition of obstacles, the position and angle of individual components may change significantly after placement migration. Therefore, directly using the coordinates of the original routes for migration will lead to misalignment. To preserve the characteristics of the original PCB routing as much as possible while ensuring rotation invariance and robustness, we transform the placement of both the referenced and the new placement into routing path graphs to guide the following routing migration instead of using coordinates directly.

Specifically, the routing path graph consists of three types of nodes: component nodes, pad nodes, and grid nodes. Each component on the PCB corresponds to a component node and is connected with all its corresponding pad nodes. Each grid node represents a square area within the board frame. Generally, areas with denser pads usually include more fine-grained routing characteristics, so the square grids in these regions are divided into finer granularity to improve the accuracy of routing migration. If a grid overlaps with a component or pad, the node corresponding to this grid will establish an edge with the component or pad node.

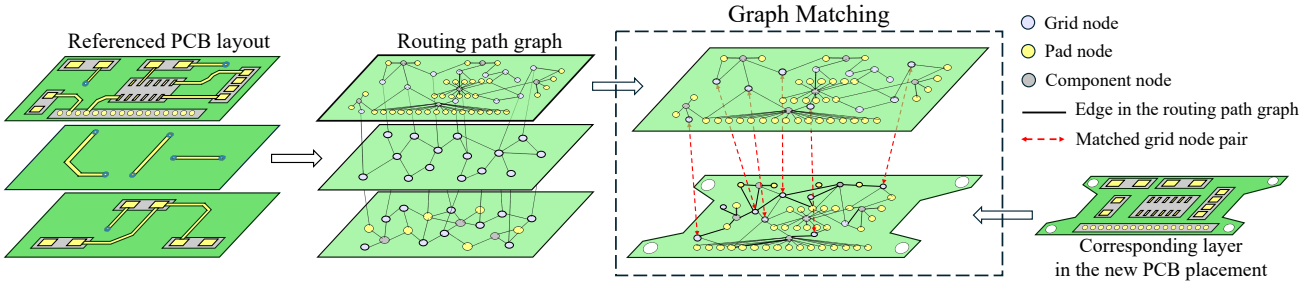


Fig. 5 Routing path graph generation and graph matching in the case of multi-layer PCB layout.

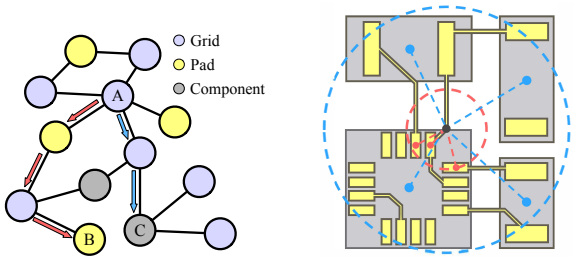


Fig. 6 (a) Example of neighborhood connection calculation. To the grid node A, 3 steps are required to reach pad node B (the red path), so the corresponding value is $(t_p)^3$. Similarly, 2 steps are required to reach component node C (the blue path), so the corresponding value is $(t_c)^2$. (b) Pad-dense area with fine-grained routing characteristics (shown in red). Regular routing area between components, with coarse-grained routing characteristics (shown in blue).

In addition, the grid node will establish edges with all the adjacent nodes in all three dimensions, in order to support routing migration for multi-layer PCB layouts. Through this method, we establish a graph structure without coordinates to perform routing migration, as shown in Fig. 5. The routing path graph exhibits rotation and symmetry invariance, enabling it to handle challenging routing migration situations.

In the proposed routing path graph, component and pad nodes serve as references to guide the subsequent graph matching process. PCB-Migrator will extract the endpoints of each wire segment in the referenced layout and project them to the corresponding positions in the new PCB. Because all wires exist in the areas corresponding to the grid nodes, we propose using neighborhood connections of these grid nodes to extract the original routing characteristics. The total dimension of the neighborhood connections is determined by the sum of the number of components and the number of pads. We apply breadth-first search (BFS) starting from the current grid node to identify reachable component nodes or pad nodes within a specified search range. Specifically, for a component or pad, $x_{\{c,p\}}$ denotes the number of BFS steps required to reach the component or pad from the current grid node, and $t_{\{c,p\}}$ represents the decay coefficient, then the corresponding value of the neighborhood connection $f_{\{c,p\}}$ is calculated as Fig. 6(a) and following equation:

$$f_{\{c,p\}} = (t_{\{c,p\}})^{x_{\{c,p\}}},\ t_{\{c,p\}} < 1,\ x_{\{c,p\}} < x^{max}_{\{c,p\}}, \quad (10)$$

where $x^{max}_{\{c,p\}}$ is the maximum BFS steps allowed for reaching components or pads. If a component or pad cannot be reached within the specific number of steps, its value is 0.

However, the depth of the two BFSs to the component and pad must be set differently. In PCB layouts, components are typically spaced farther apart than pads, resulting in coarse-grained routing characteristics where connections span larger distances, as shown in Fig. 6(b). Therefore, we employ a deeper BFS to build the neighborhood connections for components, enabling the extraction of coarse-grained routing paths between components. Conversely, typically clustered in small areas and have smaller physical dimensions, resulting in fine-grained routing characteristics where connections are confined to shorter distances, as shown in Fig. 6(b). To capture the neighborhood connections for pads, we must use a smaller BFS depth, denoted as x_p, compared to the depth used for components, denoted as x_c. Thus, $x_p < x_c$.

Through this method, both coarse-grained and fine-grained routing characteristics are accurately captured in the neighborhood connections of grid nodes. This enables flexible and robust routing migration.

Routing Path Graph Matching. In order to map the routing result of the referenced PCB to the new PCB with the neighborhood connections of the grid nodes in the routing path graphs, we need to perform a graph matching process.

Since the neighborhood connections of grid nodes are already represented as vectors, cosine similarity is used to match grid nodes from the original and new PCB placement. Because the connection relationship between grid nodes and the component nodes and pad nodes as references has been represented in the neighborhood connections, the routing migration problem can be decomposed into multiple bipartite graph matching problems by layer, as shown in Fig. 5 and the following formulation:

$$\begin{aligned} \max\ & \sum_{u=1}^{|N_o|}\sum_{v=1}^{|N_n|} \cos(u,v)\cdot x_{uv}, \\ \text{s.t.}\ & \sum_{v=1}^{|N_n|} x_{uv} = 1,\ \forall u \in N_o,\ x_{uv} \in \{0,1\},\ \forall u, v, \end{aligned} \quad (11)$$

where u and v represent grid nodes from the original and new routing path graphs, respectively. N_o and N_n are the complete collections of nodes in the original and new routing path graphs. x_{uv} is a binary variable indicating whether node u and node v are matched.

During the extraction of neighborhood connections, only grids containing endpoints of wire segments are involved, as they contain key information about the original routing characteristics. Under this setup, the number of nodes in the original routing path graph is less than that of the new routing path graph, which means that each node of the original routing path graph can correspond to one node in

TABLE I Benchmark statistics.

Case	Referenced Layout			Boundary		Obstacles	
	WL / Via	↓ IL	↓ XT	L (%)	W (%)	Num	Area (%)
P1	1300.7 / 44	3.63	-45.19	+36.42	-17.11	2	8.92
P2	1043.2 / 37	1.51	-42.73	+5.00	-12.20	1	5.62
P3	681.6 / 33	1.62	-35.44	-10.91	+21.43	1	15.97
P4	3721.3 / 166	3.26	-44.38	0.00	0.00	3	1.76
P5	1743.4 / 37	4.27	-43.32	+24.14	-26.89	2	4.47
P6	309.1 / 8	5.07	-38.17	-38.00	-18.18	4	14.91
P7	1125.6 / 72	5.43	-41.89	0.00	-33.21	2	5.71
P8	579.5 / 19	4.10	-44.81	-38.00	0.00	1	9.34
P9	257.3 / 16	1.51	-42.03	-16.84	0.00	2	3.36
P10	1523.7 / 99	1.67	-50.81	0.00	-12.73	4	2.17
P11	1245.8 / 34	2.95	-43.22	-7.17	-10.27	4	5.25

the new routing path graph with the highest similarity, that is $N_o < N_n$, which ensures that the bipartite graph matching can be performed correctly. In PCB-Migrator, the Kuhn-Munkres algorithm is employed to solve this bipartite graph matching problem.

Guided Detailed Routing. Following the graph-based global routing, PCB-Migrator generates approximate routing paths that serve as guidance for A* search-based detailed routing. These preliminary paths are subsequently utilized to formulate a refined cost map for detailed path exploration. PCB-Migrator reduces the cost of the area within the rectangle formed by two adjacent matched grid nodes from the same net, thus guiding the detailed routing. The performance of graph matching may degrade significantly in the top or bottom layers with very few components due to the limited availability of reference nodes. Therefore, as shown in Equation (12), we determine the minimum value of the modified cost based on the number of components in the current layer. Specifically, layers with fewer components are assigned a lower minimum cost to prioritize routing flexibility and maintain routing quality during migration. In addition, for multi-layer PCBs, since there will be no components in the middle signal layers, we use η to adjust the minimum cost of the cost map:

$$\Upsilon^l_{\min} = (e^{-\frac{|C_l|}{\psi}} + \eta) \cdot \Upsilon_b, \tag{12}$$

$$\Upsilon^l(d) = \frac{\Upsilon^l_{\min} + \Upsilon_b}{2} - \frac{\Upsilon^l_{\min} - \Upsilon_b}{2} \cos(\frac{\pi d}{d_{max}}), \tag{13}$$

where $\Upsilon^l_{\min}$ and Υ_b represent the minimum cost of matched grid on layer l and the base cost, respectively. ψ, α and β are hyperparameters to adjust the cost. C_l represents the complete collection of components on layer l. Equation (13) defines the cost of any location within the maximum distance d_{max} from the location with the minimum cost. This ensures smooth transitions in the cost map, enhancing the effectiveness and robustness of detailed routing.

IV. Experimental Results

A. Experiment Setting

We implement PCB-Migrator in C++, and Gurobi [25] for linear programming solving. The Boost C++ library [26] is used to perform geometric computations within our proposed algorithm. We run all the experiments of PCB-Migrator on a Linux server with 76 Intel Xeon CPU cores.

In the process of constructing the OCG, the distance threshold is set as 30 mm. As for the calculation of neighborhood connections, the maximum search steps x_c^{max} and x_p^{max} are set as 30 and 15, the decay coefficients t_c and t_p are set as 0.6 and 0.8. The rip-up and reroute processes are performed 5 times.

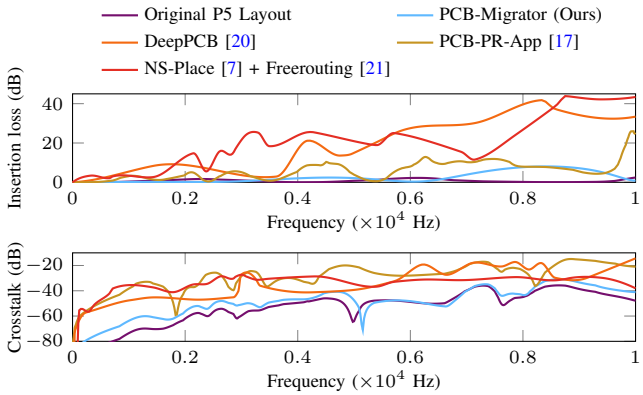


Fig. 7 Comparison of our PCB-Migrator with other baselines on (top) insertion loss and (bottom) crosstalk for case `P5`.

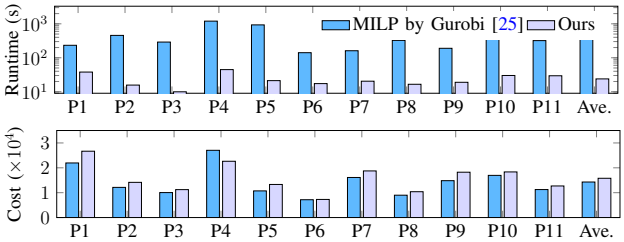


Fig. 8 Comparison of placement between our PCB-Migrator and the baseline on (top) runtime and (bottom) cost.

B. Benchmarks, Baselines and Performance Evaluation

All PCB layout benchmarks are from OpenROAD [22] and real-world industrial applications. To simulate real PCB migration scenarios, we modify the boundaries and add extra obstacles to each benchmark, as listed in TABLE I. Compared with the referenced layouts, the PnR of all the benchmarks is significantly more challenging. Consequently, directly transferring the referenced layout results to the new board frames is infeasible.

The baselines in our experiments are as follows:

1) **PCB-PR-App** [17]: an open-sourced and effective framework for PCB PnR, released by OpenROAD [22], composed of SA-PCB [19] and PcbRouter [18]. The PnR process is implemented using simulated annealing and A* search, respectively.
2) **NS-Place** [7] + **Freerouting** [21]: NS-Place [7] is a popular framework for general PCB placement that reduces net congestion to enhance routability. And Freerouting [21] is a powerful, fully autonomous PCB routing software that can be integrated into Kicad [27].
3) **DeepPCB** [20]: a powerful commercial tool for PCB PnR based on reinforcement learning; its placer and router leverage lots of design experience through learning.

The design rule checks of all PCB layouts are performed by Kicad [27]. Keysight ADS [28] is used to simulate the key electromagnetic indicators of PCB layouts: insertion loss (IL) and crosstalk (XT) [2]. These two metrics are both expressed in decibels in our experiments.

TABLE II Performance comparison of PCB layout design.

Case	PCB-PR-App [17]				NS-Place [7] + Freerouting [21]				DeepPCB [20]				**PCB-Migrator (Ours)**			
	WL / Via	EM Perf.		RT (s)	WL / Via	EM Perf.		RT (s)	WL / Via	EM Perf.		RT (s)	WL / Via	EM Perf.		RT (s)
		↓ IL	↓ XT			↓ IL	↓ XT			↓ IL	↓ XT			↓ IL	↓ XT	
P1	1765.6 / 153	7.38	-33.20	340	1510.9 / 63	7.29	-33.42	10992	1197.3 / 49	9.34	-37.16	12761	**1136.0 / 37**	**4.27**	**-37.35**	**180**
P2	1205.7 / 73	2.73	-38.51	182	1035.1 / 65	3.68	-37.11	6331	990.6 / 43	5.61	-40.63	5772	**895.7 / 28**	**1.80**	**-41.04**	**68**
P3	760.0 / 53	2.31	-34.81	164	**552.1** / 26	5.45	-34.48	5109	729.6 / 37	6.01	-34.83	4433	553.8 / **18**	**1.72**	**-35.23**	**55**
P4	4400.5 / 205	5.89	-39.29	657	3358.9 / 140	6.57	-29.49	11466	3705.2 / **101**	6.63	-36.65	7128	**3231.6** / 114	**3.69**	**-40.33**	**941**
P5	1701.7 / 49	8.31	-38.66	558	1881.2 / 74	8.71	-29.30	7825	1524.3 / 50	9.56	-38.71	6312	**1515.1 / 44**	**5.85**	**-40.56**	**280**
P6	650.7 / 38	11.71	-35.20	105	1018.8 / 42	7.18	-35.89	2869	688.0 / **29**	7.08	-36.41	4238	**541.0** / 37	**6.62**	**-37.22**	**74**
P7	1484.9 / 85	8.22	-39.91	500	1525.6 / 67	11.11	-40.24	10751	1389.7 / 37	8.82	-40.48	8985	**990.0 / 20**	**6.42**	**-41.16**	**90**
P8	694.3 / **11**	8.15	-36.26	140	1120.9 / 85	8.27	-36.18	11568	918.0 / 25	9.89	-37.18	9620	**584.6** / 18	**5.33**	**-38.40**	**57**
P9	737.3 / **19**	2.22	-37.63	147	914.4 / 37	2.32	-30.52	3248	988.8 / 26	2.10	-38.78	6957	**333.2** / 21	**1.71**	**-40.60**	**61**
P10	2171.6 / 137	2.81	-42.87	836	2096.9 / 109	3.41	-38.73	19856	1950.2 / **51**	7.08	-44.10	9282	**1880.1** / 87	**1.97**	**-45.60**	**152**
P11	**1244.3** / 28	8.63	-34.59	166	1329.6 / 65	8.20	-34.42	5495	1364.4 / 47	9.71	-34.09	5674	1382.4 / **21**	**3.77**	**-35.53**	**129**
Avg.	1528.78 / 77.36	6.21	-37.36	344.92	1485.85 / 70.27	6.56	-34.53	8682.73	1404.19 / 45.00	7.44	-38.09	7378.36	**1185.77 / 40.45**	**3.92**	**-39.37**	**189.69**

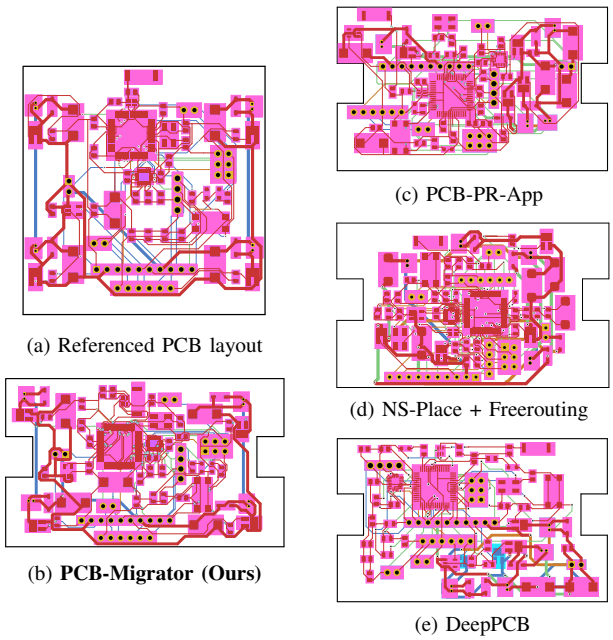
C. Migration Performance

The overall results of PCB migration on all benchmarks are shown in TABLE II. Because the boundaries of PCB benchmarks are compressed and there are additional obstacles, the difficulty of placement and routing is significantly increased. Compared with the three baselines, our PCB-Migrator achieves an average speedup of 1.8×, 45.9×, 39.1× separately. Moreover, PCB-Migrator can route with 22.4%, 20.1%, 15.6% fewer wirelength and 48.1%, 42.8%, 11.1% fewer vias than PCB-PR-App [17], NS-Place [7] + Freerouting [21], DeepPCB [20], respectively. The main reason for achieving such great performance with a shorter runtime is that we migrated the better PnR characteristics from the expert-level layout, rather than just blindly performing optimizations through conventional metrics.

Simulation results of EM performance are shown in TABLE II. Our PCB-Migrator is significantly better than all three baselines in all situations, achieving an average reduction of 36.8%, 40.2%, 47.3% in IL and 5.4%, 14.1%, 3.4% in XT, respectively. Even compared with the referenced layout, PCB-Migrator retains 81.2% on IL and 92.6% on XT of the original performance. Fig. 7 shows that the IL and XT of our PCB-Migrator are much better than any other baseline and are very close to those of the referenced layout. The reason our PCB-Migrator achieves expert-level EM performance is that it migrates from expert-level layouts, preserving the excellent PnR characteristics and design styles. The above results demonstrate that PCB-Migrator can achieve excellent EM performance close to that of the expert results.

To verify the effectiveness and efficiency of PCB-Migrator in placement, we compared the cost and runtime of PCB-Migrator and MILP by Gurobi [25], as shown in Fig. 8. PCB-Migrator achieves legal placement while requiring only 5.56% of the runtime and attains 110.51% cost compared with MILP. Notably, on the benchmark P4, PCB-Migrator achieves a cost that is 16.29% lower than MILP. This is because MILP reached its maximum solution time and was unable to converge to the optimal solution. The above experiment results denote that PCB-Migrator can achieve highly optimal placement results with a much shorter runtime.

Fig. 9 shows the migration result of P5, a relatively complicated benchmark with multiple layers. The objective of the "design-from-scratch" frameworks is usually to optimize without considering the potential impacts on performance and usability. As a result, the layouts generated by PCB-PR-App [17], NS-Place [7] + Freerouting [21], and DeepPCB [20] contain more unreasonable design styles, which will negatively impact performance. It is evident that the result of our PCB-Migrator retains the characteristics of the referenced PCB to a great extent and is significantly better than that of all other baselines. The results shown in Fig. 9 clearly illustrate the effectiveness of our proposed PCB-Migrator; expert-level PnR is the reason for its better performance.



(a) Referenced PCB layout
(b) **PCB-Migrator (Ours)**
(c) PCB-PR-App
(d) NS-Place + Freerouting
(e) DeepPCB

Fig. 9 Comparison between the referenced P5 layout and migrated layouts from our PCB-Migrator and other baselines. PCB-Migrator's layout is highly similar to the reference and more reasonable, resulting in improved EM performance.

V. Conclusion

This paper proposed PCB-Migrator, the first PCB PnR migration framework. In placement migration, PCB-Migrator extracts the positional relation between components of the referenced PCB to construct an offset constraint graph. With this graph, PCB placement is formulated as an MILP problem and solved in a speedup strategy to significantly improve computational efficiency. As for routing migration, PCB-Migrator generates routing path graphs for both the referenced and new layouts, and performs graph matching to guide detailed routing to complete PCB PnR migration. Experimental results demonstrate that PCB-Migrator can generate migrated PCB layouts with highly similar design characteristics and electromagnetic performance to the referenced PCB effectively and efficiently.

REFERENCES

[1] T.-L. Wu, H.-H. Chuang, and T.-K. Wang, "Overview of power integrity solutions on package and PCB: Decoupling and EBG isolation," *IEEE Transactions on electromagnetic compatibility*, vol. 52, no. 2, pp. 346–356, 2010.

[2] E. Bogatin, *Signal and power integrity–simplified*. Pearson Education, 2010.

[3] H. Kong, T. Yan, and M. D. Wong, "Automatic bus planner for dense PCBs," in *Proc. DAC*, 2009, pp. 326–331.

[4] Q. Liu, Q. Tang, J. Chen, C. Chen, Z. Zhu, H. He, J. Chen, and Y.-W. Chang, "Disjoint-path and golden-pin based irregular PCB routing with complex constraints," in *Proc. DAC*, 2023, pp. 1–6.

[5] T. Chen, S. Xiong, H. He, and B. Yu, "TRouter: thermal-driven PCB routing via nonlocal crisscross attention networks," *IEEE TCAD*, vol. 42, no. 10, pp. 3388–3401, 2023.

[6] T.-C. Lin, D. Merrill, Y.-Y. Wu, C. Holtz, and C.-K. Cheng, "A unified printed circuit board routing algorithm with complicated constraints and differential pairs," in *Proc. ASPDAC*, 2021, pp. 170–175.

[7] C.-K. Cheng, C.-T. Ho, and C. Holtz, "Net separation-oriented printed circuit board placement via margin maximization," in *Proc. ASPDAC*, 2022, pp. 288–293.

[8] T. Yan and M. D. Wong, "Recent research development in PCB layout," in *Proc. ICCAD*, 2010, pp. 398–403.

[9] R. Liu, X. Hong, S. Dong, Y. Cai, and J. Gu, "VLSI/PCB placement with predefined coordinate alignment constraint based on sequence pair," in *ASICON 2001. 2001 4th International Conference on ASIC Proceedings (Cat. No. 01TH8549)*. IEEE, 2001, pp. 167–170.

[10] C.-H. Tsou, S.-Y. Lin, W.-C. Hung, and Y.-W. Chang, "Late breaking results: Modern automatic PCB placement with complex constraints," in *Proc. DAC*, 2024, pp. 1–2.

[11] S. Li, Z. Zhuang, M. Liu, W. Sheng, B. Yu, and T.-Y. Ho, "HiePlace: Efficient hierarchical PCB placement," *IEEE TCAD*, 2025.

[12] M. M. Ozdal and M. D. Wong, "A length-matching routing algorithm for high-performance printed circuit boards," *IEEE TCAD*, vol. 25, no. 12, pp. 2784–2794, 2006.

[13] T. Yan, P.-C. Wu, Q. Ma, and M. D. Wong, "On the escape routing of differential pairs," in *Proc. ICCAD*, 2010, pp. 614–620.

[14] L. Vassallo and J. Bajada, "Learning circuit placement techniques through reinforcement learning with adaptive rewards," in *Proc. DATE*. IEEE, 2024, pp. 1–6.

[15] L. Chen, R. Chen, S. Hu, X. Yao, Z. Tang, S. Kai, S. Xu, M. Yuan, J. Hao, B. Yu *et al.*, "PCBAgent: An agent-based framework for high-density printed circuit board placement," in *Proc. ASPDAC*, 2025, pp. 781–787.

[16] K. Zhu, M. Liu, Y. Lin, B. Xu, S. Li, X. Tang, N. Sun, and D. Z. Pan, "Geniusroute: A new analog routing paradigm using generative neural network guidance," in *Proc. ICCAD*. IEEE, 2019, pp. 1–8.

[17] PCB-PR-App. [Online]. Available: https://github.com/The-OpenROAD-Project/PCB-PR-App

[18] PcbRouter. [Online]. Available: https://github.com/The-OpenROAD-Project/PcbRouter

[19] SA-PCB. [Online]. Available: https://github.com/The-OpenROAD-Project/SA-PCB

[20] DeepPCB. [Online]. Available: https://app.deeppcb.ai/

[21] Freerouting. [Online]. Available: https://www.freerouting.app/

[22] OpenROAD. [Online]. Available: https://github.com/the-openroad-project

[23] E. Young, C. Chu, and M. Ho, "Placement constraints in floorplan design," *IEEE TVLSI*, vol. 12, no. 7, pp. 735–745, 2004.

[24] X. Gao, H. Zhang, M. Liu, L. Shen, D. Z. Pan, Y. Lin, R. Wang, and R. Huang, "Interactive analog layout editing with instant placement and routing legalization," *IEEE TCAD*, vol. 42, no. 3, pp. 698–711, 2022.

[25] Gurobi Optimization. [Online]. Available: https://www.gurobi.com/

[26] Boost. [Online]. Available: https://www.boost.org/

[27] Kicad. [Online]. Available: https://www.kicad.org/

[28] Keysight ADS. [Online]. Available: https://www.keysight.com/us/en/products/software/pathwave-design-software/pathwave-advanced-design-system.html