HiePlace: Efficient Hierarchical PCB Placement

Shanyi Li, Zhen Zhuang, Mingyu Liu, Weihua Sheng, Bei Yu, *Senior Member, IEEE*, and Tsung-Yi Ho, *Fellow, IEEE*

***Abstract*—Due to the rapid expansion of printed circuit board (PCB) designs, accompanied by diverse design rules and specific constraints, there has been a substantial increase in manual design engineering efforts. To address this challenge, industries are seeking productivity improvements through automated placement techniques. However, existing placers primarily target VLSI placement and do not align well with PCBs' unique characteristics. This mismatch arises from both the customization of PCBs and the complexity of the problem, which involves considering various constraints such as priorities, irregularities, and alignment. This article introduces HiePlace, an efficient mathematical-programming (MP)-based placement framework designed explicitly for PCBs. It aims to address the diverse constraints and achieve better performance. To address the issue of time-consuming computation in the direct MP-based algorithm, we present two innovative acceleration techniques: 1) in the initial stage, we introduce a dynamic programming approach to prioritize the placement of core components. This technique effectively reduces the solution space and enhances the overall placement quality and 2) in addition, we propose a relaxation algorithm to minimize the number of boolean variables and further narrow down the solution space. This approach enables more efficient placement results by considering the problem's specific constraints. Experimental results show that the proposed framework produces 7.7× speed up and 66% cost reduction.**

***Index Terms*—Design automation, mathematical programming, optimization, printed circuits.**

I. Introduction

PRINTED circuit board (PCB) placement is a critical procedure in industrial chip design. However, the current approach heavily relies on manual designs, leading to significant time consumption, especially for large-scale designs. PCB placement problems also encompass a wide range of challenges, from accommodating multiple types of components with different shapes, sizes, and functions to adhering to various design rules and incorporating specific constraints. As a result, existing EDA tools cannot effectively support PCB placement issues.

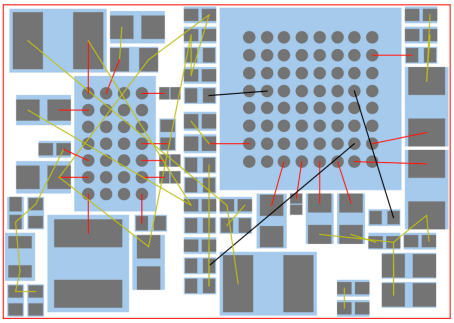


Fig. 1. Example of the PCB placement.

VLSI placement remains a key focus of academic research, primarily addressing the challenges of large-scale designs. The emphasis is on standardization to simplify the design process and ensure consistency across various projects. Advanced algorithms and automated tools play a crucial role in VLSI placement, allowing engineers to efficiently manage extensive projects through fully automated placement capabilities [1], [2], [3]. In contrast, PCB placement operates on a more human-controllable scale. As shown in Fig. 1, it involves the meticulous arrangement of components on a PCB, considering various design constraints and rules. PCB placement offers a higher degree of customization compared to digital placement. This level of control allows for fine-tuning and optimization, particularly in cases where unique design constraints or specialized requirements are present. However, this level of manual control comes at the cost of increased time and effort, particularly for larger and more complex designs.

Moreover, Pu et al. [4] mentioned that in the macro placement of VLSI, macros are supposed to be placed in the boundary area of the chip and the much smaller standard cells in the center of the region. It helps improve the performance of VLSI designs and prevents macro blockage. On the contrary, an effective way to enhance the layout quality in PCB placement is to place the larger core chip in the center and its related components around the chip. Modern PCB designs face additional complexities as they expand to accommodate contemporary devices. These designs require precise spacing between component pairs to facilitate direct network connections [5]. Notably, PCBs exhibit significantly higher component density despite having fewer components compared to VLSI circuits. Furthermore, PCB placement must address unique constraints not present in traditional VLSI placement problems, making the direct application of VLSI placement techniques inadequate.

Received 22 December 2024; revised 1 April 2025 and 26 May 2025; accepted 7 June 2025. Date of publication 17 June 2025; date of current version 6 January 2026. This work was supported in part by the JC STEM Laboratory of Intelligent Design Automation funded by The Hong Kong Jockey Club Charities Trust; in part by the Research Grants Council of Hong Kong, SAR, under Grant CUHK14211324 and Grant CUHK14211824; and in part by the Science and Technology Innovation Key Research and Development Program of Chongqing. This article was recommended by Associate Editor I. H. R. Jiang. *(Corresponding author: Mingyu Liu.)*

Shanyi Li, Zhen Zhuang, Bei Yu, and Tsung-Yi Ho are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, SAR.

Mingyu Liu is with the Industrial Software Platform Cloud Service Product Department, Huawei Cloud Computing Technology Company Ltd., Shenzhen 518129, China (e-mail: liumingyu@huawei.com).

Weihua Sheng is with the Design Automation Lab, Huawei HiSilicon, Hong Kong.

Digital Object Identifier 10.1109/TCAD.2025.3580511

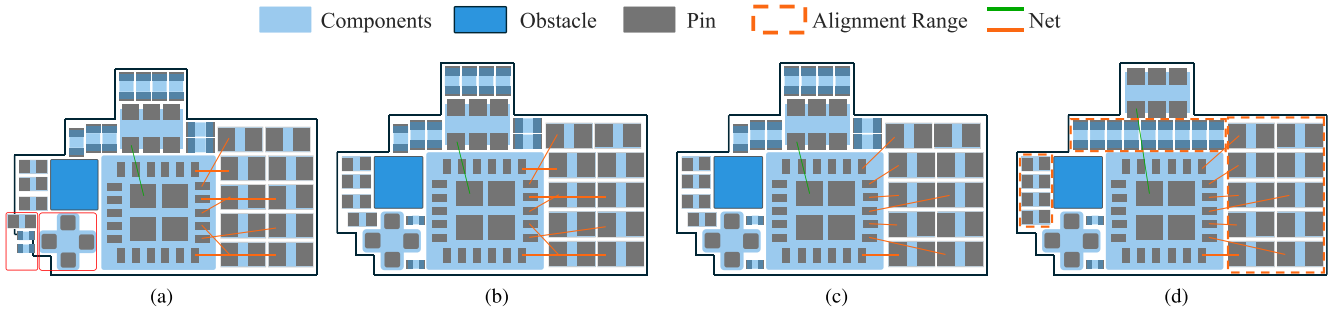


Fig. 2. Comparison of PCB placement results using different algorithms. (a) Result by analytical placer w/o. considering irregular components. (b) Result by analytical placer considering irregular components. (c) Result by HiePlace: w/o. considering alignment. (d) Result by HiePlace: considering components priority and alignment.

The first is irregularity. As shown in Fig. 2, the PCB outline of practical devices exhibits a distinctive and intricate structure that sets it apart from traditional regular shapes. The intricate nature of the PCB outline is due to the complexity of modern electronic devices, which integrate various components like ICs, connectors, capacitors, and resistors [6]. Each component requires a designated position and interconnection within the PCB layout. Thus, the PCB outline must be carefully crafted to accommodate these components, ensuring sufficient space, routing channels, and adherence to design constraints and functional requirements. According to industrial manufacturing requirements, in addition to the irregular outline, some related parts must also overlap symmetrically in the center, forming an irregular polygon, like the polygon with a red border in Fig. 2(a). These components serve as a strategic arrangement to address specific design considerations and ensure the optimal performance of associated components. By embracing this unconventional geometry, PCB designers can effectively manage signal integrity and power distribution, contributing to the electronic system's overall performance.

Component alignment represents a critical constraint in PCB placement optimization. A common requirement is the adjacent positioning of identical modules in vertical or horizontal arrangements. This alignment consideration directly impacts the manufacturing process, where automated robotic arms perform precise component placement. Well-optimized component alignment streamlines the robotic arm's movement patterns, thereby reducing mechanical complexity, minimizing positioning adjustments, and ultimately enhancing manufacturing efficiency and production yield. Furthermore, there are additional general constraints to consider, such as spacing constraints [7], preplaced components [8], [9], and obstacle blocks [10], which further complicate our problem.

A. Previous Work

The mainstream of PCB placement is still human-driven and assisted by semi-automatic methods. However, the scale of PCB design has been rapidly increasing in recent years, and the amount of manual design engineering work required has increased significantly due to the various components, design rules, and specific constraints involved. Therefore, the industry is actively pursuing automation to improve productivity and efficiency in this field. Previous research in PCB placement has largely overlooked the unique characteristics inherent to PCBs [11], [12]. Conventional approaches primarily utilize B*tree representations and focus on optimizing wirelength and area as the main objectives. While simulated annealing (SA) [13] algorithms are commonly employed to enhance placement results, the tree-based representations suffer from incomplete solution space coverage [14], limiting the achievement of optimal placement results. Another category of approaches encompasses analytical methods, which employ mathematical and physical principles [15], [16], [17] to construct PCB placement models. Force-directed algorithms [18] exemplify these methods, transforming the placement challenge into an optimization problem by formulating a smooth objective function and seeking solutions that minimize this function under specified constraints. However, these analytical approaches encounter several significant limitations. The validation test cases are frequently oversimplified or synthetic, failing to reflect the complexity encountered in real-world PCB designs. Furthermore, the challenge of accurately representing intricate design constraints persists, potentially undermining the practical viability of these methods in industrial applications.

Recent advances in integrated circuit (IC) placement have introduced novel approaches that formulate placement tasks as Markov decision processes. These methods place components sequentially on a grid-based canvas and employ reinforcement learning (RL) techniques to optimize placement outcomes [19], [20], [21], [22], [23]. While PCB placement shares certain conceptual similarities with IC placement, it presents distinct challenges, including higher component density, irregular boundary constraints, and diverse spacing requirements. These fundamental differences significantly limit the direct applicability of IC placement algorithms to PCB scenarios. A particularly critical challenge emerges in grid-based approaches: during sequential placement, positioning the final components while satisfying spacing and overlap constraints becomes increasingly difficult. This difficulty stems from an inherent tradeoff: coarse grid meshes result in inefficient space utilization for most components, while fine-grained meshes generate an overwhelming number of state candidates, making RL approaches time-consuming and memory-intensive.

To represent the diverse constraints accurately, a novel mathematical programming (MP) method can be employed to formulate these constraints into a mathematical framework. Zhuang et al. [24] and Chen et al. [25] proposed MP-based algorithms that address the reliability constraints for chiplet placement and package placement. The problem scale in these scenarios is manageable for an MP solver, making it easier to obtain feasible solutions. However, in the case of PCB placement, the number of components is larger, and the placement area is denser. Therefore, additional techniques are required to reduce the problem solution space and enhance efficiency.

Fig. 2 compares PCB placement results using different methods. In an irregular outline, the figure showcases the positioning of a core chip and multiple components, including irregular and preplaced components. Notably, the orange-marked nets connect the core chip and the highest-priority components. Fig. 2(a) shows a result achieved through an analytical placer, employing a simple rectangle bounding box for irregular components. As shown in the figure, some components exceed the boundary limits, and the spacing performance is unsatisfactory since the irregular components waste a lot of space. At the same time, the previous layout still caused some network intersection issues and lack of alignment issues. By accounting for irregular components, hard constraints such as boundary and spacing requirements can be satisfied, as shown in Fig. 2(b). However, these results lack consideration for the placement of first-priority components. To address this, our method emphasizes first-priority placement, aiming to eliminate crossings between high-priority nets. In addition, we focus on aligning components of similar sizes and optimizing their specific positions. As a result, Fig. 2(d) shows the results of our approach, where components of similar sizes are well-aligned, effectively utilizing the irregular outline and irregular components. This ensures adherence to specific constraints while maintaining minimal changes to overall wirelength and significantly improving routability.

B. Our Contributions

This article comprehensively defines the specific constraints related to PCB placement and proposes a hierarchical placement framework, HiePlace, for PCB placement using an MP-based algorithm. The proposed approach aims to optimize the placement performance and reduce the runtime to achieve an efficient solution. We summarize our major contributions as follows.

1) Regarding the nature of PCB placement, we adopted an MP-based approach. This approach involves converting each constraint into a corresponding mathematical formula, allowing for precise modeling and optimizing the placement process.
2) For the time-consuming issues in the MP-based algorithm, we propose a novel grouping algorithm that clusters highly correlated components together to improve the placement alignment and reduce the solution space.
3) We present a dynamic-programming (DP)-based algorithm designed to accelerate the priority placement of core components while minimizing net intersections.
4) For the subsequent placement, we introduce a relaxation algorithm to narrow down the solution space and reduce the runtime of the MP solver.
5) In addition, we present a postprocessing fine-tuning algorithm to enhance alignment and improve routability.
6) Experimental results demonstrate that our method effectively satisfies various constraints while improving both performance and runtime.

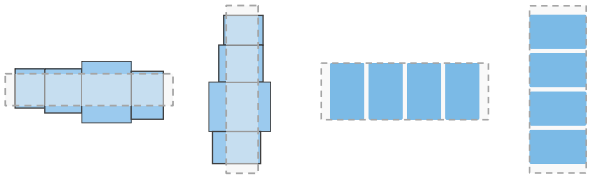
The remainder of this article is organized as follows. Section II introduces the problem formulation. Section III illustrates a general mathematical formulation considering the specific constraints for PCB, and Section IV presents our fast hierarchical placement flow. Section V reports the experimental results and Section VII concludes this article.

II. Problem Formulation

In this section, we will begin by introducing the terminologies used in our framework. Within each outline, we define a module $\mathcal{M}$ as a combination of a core chip and its surrounding components. In each module, components are assigned distinct priorities based on their connection relationships and importance to the chip. Consequently, during placement, the wirelength of each net carries different weights, enabling the distinction of priorities [25]. A single outline can contain anywhere from one to several modules. Within each module, the components directly connected to the outermost pins of the core chip are referred to as first-priority components. As depicted by the right-aligned components connected to the orange net in Fig. 2(c), these components exhibit the most frequent connections to the core chip. Hence, minimizing first-priority network wirelength becomes crucial to reduce the latency and ensure signal integrity. Ideally, the net connections between the chip and the first-priority components are on the same layer of the PCB. This approach reduces the need for drilling on multiple layers, enhancing the routability of the detailed PCB placement results.

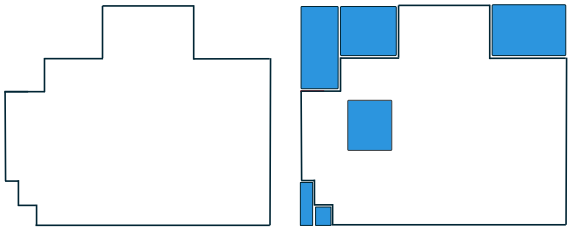
In PCB placement, the alignment constraint is also a significant concern. However, previous works [26], [27] addressing alignment constraints have been comparatively less stringent than those in PCB placement, as shown in Fig. 3. In traditional alignment scenarios, an alignment path exists, and each block satisfying the constraint can overlap with the path either horizontally or vertically. However, in PCB placement, the alignment constraint differs: components of the same size must align together, and the width of the alignment path matches the component's width, as illustrated in Fig. 3(b). This alignment approach substantially enhances data transmission efficiency, reduces complexity in the manufacturing process, and ultimately improves overall efficiency and yield. In our problem, the number of misaligned areas will be counted, and the lower the count, the more components are aligned together.

Considering the above PCB constraints, we formulate our placement problem as follows. The input of this work includes



(a) (b)

Fig. 3. Illustration of alignment constraint. (a) Alignment path in VLSI placement. (b) Alignment path in PCB placement.



(a) (b)

Fig. 4. (a) Example of the irregular outline. (b) Adjustment of irregular outline.

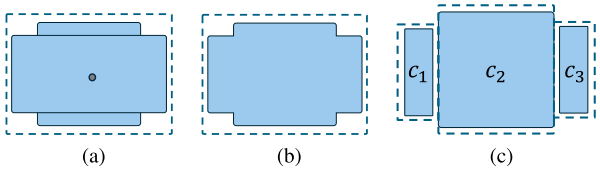
an irregular fixed outline, a set of modules $\mathcal{M}$, which contains a set of chips $\mathcal{U}$, and a set of components $\mathcal{C}$. On each component, we have multiple pins $\mathcal{P}$ with fixed coordinates, and we also have a netlist $\mathcal{N}$ that connects the pins of different components. The output is the detailed placement solution, which should place chips and components within the outline. The problem's constraints can be described below. A spacing table is provided that defines the spacing constraint between every two components, as well as the spacing between components and outline. The coordinates and shapes of the preplaced components and the obstacle blocks are given. The primary objective of this problem is to minimize both the weighted wirelength and white space ratio. It is essential to satisfy alignment constraints by reducing the number of misaligned areas. In addition, minimizing net crossings for high-priority component placement is crucial to achieving an optimal solution.

III. Placement: Mathematical Formulation

In this section, we comprehensively explain the fundamental mathematical formulation for PCB placement. This work is distinguished from previous MP-based approaches because it considers irregular outlines and components. To address these constraints, we incorporate a preprocessing step to handle irregular outlines and components effectively. This step aids in preparing the input for the subsequent MP model. By leveraging this model, we can optimize component placement while accounting for the irregularities and constraints inherent in PCB designs.

A. Preprocessing

Due to the complicated structure of PCB components, the outline becomes irregular and flexible, as shown in Fig. 4(a). As mentioned in [8], we use a rectangle bounding box to surround the irregular outline, and the white space between the outline and the bounding box can be treated as several preplaced components. Fig. 4(b) provides an example of outline



(a) (b) (c)

Fig. 5. Illustration of irregular components. (a) Two components are stacked symmetrically. (b) Stacked components form an irregular boundary. (c) Decompose and generate new bounding boxes.

preprocessing. To fill the irregular outline, we employ five blocks. Similar to the preplaced components, the lower left corner coordinates (x_t, y_t) remain fixed during placement.

The presence of irregular components formed by stacking introduces significant challenges in placement. Since stacked components are rectangular, centrosymmetric, and can be placed horizontally or vertically, they often create a centrally symmetric sawtooth shape, leading to inefficient space utilization, particularly in corner regions, as illustrated in Fig. 5(b). Such inefficiency can even render placement solutions infeasible. To address this issue, we propose a subdivision strategy that decomposes these combined components into smaller subcomponents from left to right, redefining the bounding box for each component [Fig. 5(c)]. In addition, our placement framework incorporates a cohesion constraint to maintain the structural integrity of these subdivided modules. This constraint guarantees fixed relative coordinates among the subcomponents, ensuring their proper relationship within the PCB placement. Considering c_2 as the center, the constraint can be formulated as follows:

$$x_{c_1} = x_{c_2} - r_{c_2} \times 0.5 \times \left(h_{c_1} - h_{c_2}\right) - \left(1 - r_{c_2}\right) \times w_{c_1} \quad (1)$$

$$y_{c_1} = y_{c_2} - \left(1 - r_{c_2}\right) \times 0.5 \times \left(h_{c_2} - h_{c_1}\right) - r_{c_2} \times w_{c_1} \quad (2)$$

$$x_{c_3} = x_{c_2} - r_{c_2} \times 0.5 \times \left(h_{c_3} - h_{c_2}\right) + \left(1 - r_{c_2}\right) \times w_{c_2} \quad (3)$$

$$y_{c_3} = y_{c_2} - \left(1 - r_{c_2}\right) \times 0.5 \times \left(h_{c_2} - h_{c_1}\right) + r_{c_2} \times w_{c_2} \quad (4)$$

$$r_{c_2} \in \{0, 1\} \quad (5)$$

where x_{c_i}, y_{c_i}, x_{c_i}, y_{c_i} represent the x-coordinate of lower-left corner, y-coordinate of lower-left corner, width, and height of c_i, respectively. r_{c_i} is used to indicate whether a component is rotated. In this particular example, where c_2 is the center component, we only take into account the rotation of c_2, while the other two components follow its rotation.

B. MP-Based Method for Placement

In addition to the previously formulated constraints, the primary constraints in the previous approach were the overlap constraint and the boundary constraint. These constraints play a crucial role in ensuring that components do not overlap and remain within the specified boundaries.

For each component $c \in \mathcal{C}$, it cannot overlap with other components and thus can be constrained as

$$x_{c_i} + w_{c_i} \le x_{c_j} + W \cdot \left(p_{i,j} + q_{i,j}\right) \quad (6)$$

$$y_{c_i} + h_{c_i} \le y_{c_j} + H \cdot \left(1 + p_{i,j} - q_{i,j}\right) \quad (7)$$

$$x_{c_i} - w_{c_i} \ge x_{c_j} - W \cdot \left(1 - p_{i,j} + q_{i,j}\right) \quad (8)$$

$$y_{c_i} - h_{c_i} \ge y_{c_j} - H \cdot \left(2 - p_{i,j} - q_{i,j}\right) \quad (9)$$

$$p_{i,j}, q_{i,j} \in \{0, 1\}, 1 \le i < j \le |\mathcal{C}| \quad (10)$$

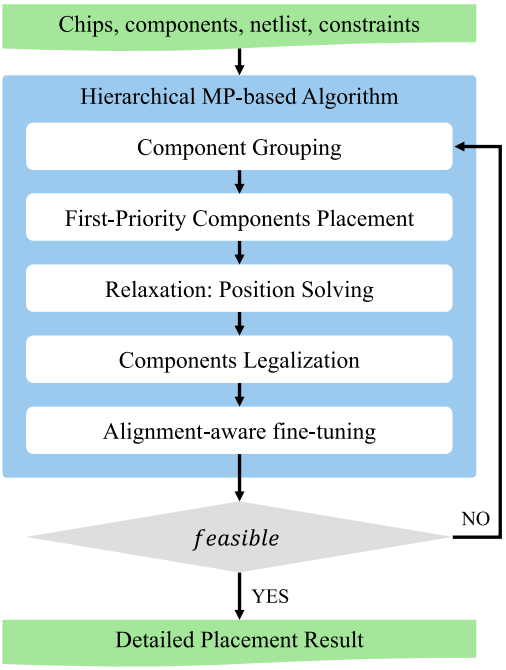


Fig. 6. Design flow of HiePlace.

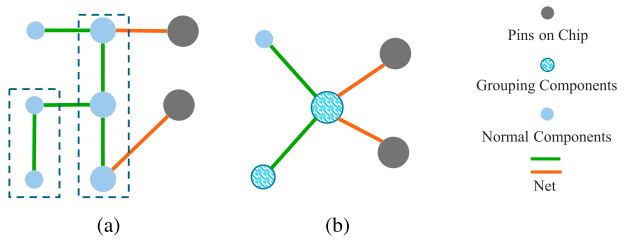


Fig. 7. Illustration of components construction graph. (a) Current components graph. (b) Nodes are merged by net connection and component size.

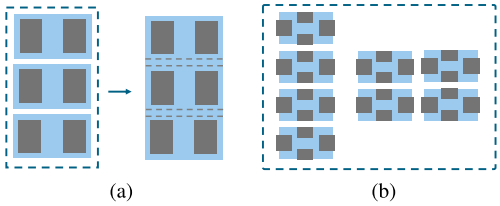


Fig. 8. Illustration of component grouping. (a) Example of the component grouping. (b) Different patterns for a single group.

where $p_{i,j}$ and $q_{i,j}$ represent the relative relationship between the components c_i and c_j. W and H represent the width and height of the outline, respectively.

However, the presence of Boolean variables, such as $p_{i,j}$ and $q_{i,j}$, significantly increases the runtime of the MP solver, making it challenging to obtain improved results. To overcome this limitation and enhance the placement process, it is crucial to determine the components' positions or their relative positions in a way that minimizes the usage of Boolean variables. To address this issue, we propose a hierarchical approach that reduces the reliance on Boolean variables and narrows the solution space. This approach aims to accelerate the MP-based algorithm, enabling us to achieve a more efficient and optimized placement result.

IV. Efficient Hierarchical Placement

In this section, we provide an overview of our placement framework and then detail our methods in each stage. The algorithm has five steps, as shown in Fig. 6. In the grouping stage, we initially group the first-priority components based on the position of the pins on the chip. This grouping takes into account the interconnections between the components. Considering their connections, we also group other small components with the same height and width. In the first-priority components placement stage, we utilize a DP algorithm to determine the optimal placement solution for the core components. In the subsequent two steps, we employ a relaxation algorithm to narrow the solution space further and obtain an approximate module location. This helps refine the placement process and achieve better results. Following this, an optimized MP-based algorithm is utilized to place the remaining components and fine-tune the module locations. In the final stage, we employ a fast greedy algorithm that performs local fine-tuning to satisfy alignment constraints better, yielding superior placement results.

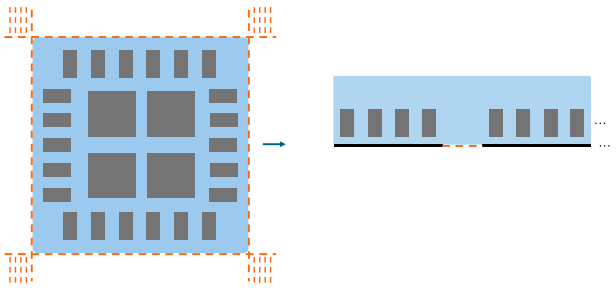
A. Component Grouping

During the components grouping stage, the first step involves constructing a component graph for the given PCB. This component graph, denoted as G_c, is generated from the schematic diagram by condensing all pins within the same component into a single vertex, except for the chip. Each edge in G_c represents a net connection between components. An example of a component graph is illustrated in Fig. 7, where circles of different sizes represent components of varying sizes. As depicted in Fig. 7(a), the nodes along the middle line possess the same size and are connected by a single net. Consequently, we can group these three nodes into a larger consolidated node, as shown in Fig. 7(b).

During the process of grouping components, it is essential to consider space constraints. As illustrated in Fig. 8(a), there must be a certain amount of space left between each pair of grouped components, as indicated by the two dashed lines. Consequently, if we merge the components vertically, as shown in the figure, the resulting height of the components will be

$$\mathcal{C}h_{\text{new}} \equiv \mathcal{C}h_{\text{origin}} \times V_{cnt} + sp \times (V_{cnt} - 1) \tag{11}$$

where $\mathcal{C}h_{\text{origin}}$, V_{cnt}, and sp represent the single component height, the vertical components count, and the minimum space constraint between two components, respectively. Once a group is formed, each group will have a distinct pattern that can be utilized during the placement phase. Different patterns allow for adaptation to various scenarios. For instance, as depicted in Fig. 8(b), four components can be grouped using the 1 by 4 or 2 by 2 patterns. Moreover, if the hierarchical placement framework fails to generate a feasible solution, we can return to the grouping phase, divide the group into subpatterns, and then proceed with the placement. By



(a) (b)

Fig. 9. Illustration of chip expansion. (a) Chip with multiple pins. (b) Result of 1-D expansion.

processing the input data at the initial stage, we can effectively handle the PCB irregularities and enhance the alignment of the final solution. Simultaneously, it reduces the solution space and contributes to runtime optimization.

B. First-Priority Components Placement

Following the grouping process, our focus shifts to the placement of first-priority components to minimize network crossings and wirelength. In this stage, we transform the 2-D rectangular chip into a 1-D segment, with the pins connecting the first-priority components positioned above the segment. As illustrated in Fig. 9, we initiate the expansion process from one corner of the chip, converting it into a 1-D representation. Due to the direct expansion of the rectangle, there may be some unused space in the corners. A portion of the segment is further extended to address this issue and accommodate the placement of components in the corners. This area is described by the orange dashed segment in Fig. 9(b).

Once expanded to 1-D, we calculate the HPWL between the external pins and the priority component, starting from left to right. Due to the possibility of a component being connected to various pins on the chip, multiple net crossings may occur. To mitigate this, we employ a reordering algorithm, referred to as Algorithm 1, which aims to minimize the number of crossings and initially reduce the wirelength.

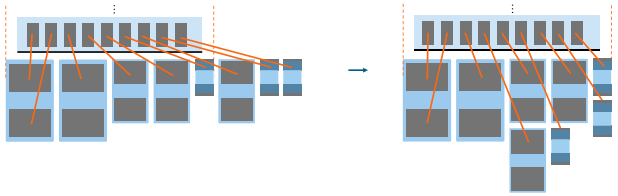
Algorithm 1 begins by parsing the input data and performing dimensional reduction on the chip layout. This preprocessing step yields the initial ordering of first-priority components, which we initialize directly from the input component sequence. Moving forward, within the subsequent loop, we systematically rearrange the positions of pairs of components and calculate the number of net crossings each time. Since our primary goal is to enhance routability, we consider the crossing count a more significant metric. Therefore, only when the new crossing count is lower or equal to the previous count can we recalculate the wirelength and update the vector that holds the components' order. Finally, we obtain a final order for the components and input for the DP phase.

After the reordering, we achieve a favorable arrangement of the priority components. However, it is common for these components' combined widths to exceed the chip's boundaries, as shown in Fig. 10(a). As a result, the single-line layout does not align with the actual layout scenario. This issue becomes

Algorithme 1: Reordering

Input : A set of components $\mathcal{C}$, a set of pins $\mathcal{P}$, an set of net $\mathcal{N}$.
Output : Ordered components vector $\boldsymbol{V}$

$\boldsymbol{V} \leftarrow \texttt{ParseData}(\mathcal{C}, \mathcal{P}, \mathcal{N})$;
Let wirelength $wl \leftarrow \texttt{getWirelength}(\boldsymbol{V})$;
Let intersectCnt $ic \leftarrow \texttt{getIntersectCnt}(\boldsymbol{V})$;
$MinValue \leftarrow wl \times ic$;
for *each two cell* c_1 *and* $c_2 \in \boldsymbol{V}$ **do**
  $\boldsymbol{V_{temp}} \leftarrow \texttt{Swap}(c_1, c_2)$;
  $ic_{new} \leftarrow \texttt{getIntersectCnt}(V_{temp})$;
  if $ic_{new} > ic$ **then**
    `Continue`;
  else
    $wl_{new} \leftarrow \texttt{getWirelength}(V_{temp})$;
    if $wl_{new} \times ic_{new} < MinValue$ **then**
      $\boldsymbol{V} \leftarrow \texttt{Update}(\boldsymbol{V_{temp}})$;
      $MinValue \leftarrow wl_{new} \times ic_{new}$;
return $\boldsymbol{V}$;



(a) (b)

Fig. 10. Illustration of component selection. (a) Result after ordering. (b) Result after DP.

particularly problematic when the last component is placed far from the chip boundary, leading to excessively long wirelength and potential signal transmission loss. So, we need to choose some components for a limited layout region. We formulate the process as a 0/1 knapsack problem. The layout region can be considered as a knapsack, where the components represent items with varying values. In our specific problem, the values are determined based on the weighted component height and the wirelength. The detailed formulation of the value can be expressed as follows:

$$val(i) = h_{i_c} \times \left(1 - wl_{i_c}/len_{\text{chip}}\right) \tag{12}$$

where h_{i_c}, wl_{i_c}, and len_{chip} represent the individual component height, HPWL for the component, and the boundary of the chip, respectively. Equation (12) clearly demonstrates that components with shorter wirelength and larger sizes are prioritized for first-row placement. Following this selection criterion, unplaced components are automatically reassigned to the second layer, as visualized in Fig. 10(b). The complete selection process is implemented in Algorithm 2 using DP. The output vector $\boldsymbol{D}$ from Algorithm 2 contains the optimized positions for all selected components. We employ an interpolation strategy for unselected components: they are positioned

Algorithme 2: DP-Based Selection

Input : A vector of components $\boldsymbol{V}$, chip boundary length $\mathcal{L}$.
Output : A vector of Detailed location of components $\boldsymbol{D}$

Define dp matrix $\boldsymbol{DP} \leftarrow \mathbf{0}$; **for** $c \in \boldsymbol{V}$ **do**
  for $w \in (0, \mathcal{L})$ **do**
    if $w - h_c < 0$ **then**
      $\boldsymbol{DP}_w^i = \boldsymbol{DP}_w^{i-1}$;
    else
      $val_w \leftarrow \texttt{calValue}(c)$; $\boldsymbol{DP}_w^i = \max(\boldsymbol{DP}_w^{i-1}, \boldsymbol{DP}_{w-h_c}^{i-1} + val_w)$;
for $i \in (0, Count(\mathcal{V}))$ **do**
  if $\boldsymbol{DP}[i][\mathcal{L}] \neq \boldsymbol{DP}[i-1][\mathcal{L}]$ **then**
    $\mathcal{L} -= h_i$;
    Put $\mathcal{C}_i$ into $\boldsymbol{D}$;
return $\boldsymbol{D}$;

with the same x-coordinate with immediately succeeding components previously ordered in Algorithm 1.

C. Relaxation: Position Solving

After the first-priority placement stage, multiple grouped components still need to be placed around the chip. However, due to the slower convergence speed of the MP-based algorithm, we introduce a relaxation model to expedite our algorithm. The proposed approach aims to enhance the efficiency of the placement process for the remaining grouped components.

Prior studies [8], [28] addressed layout optimization through MP formulations that approximate soft blocks as circular entities. However, as our work targets hard block placement challenges, we have strategically adapted the solution granularity to prepare for subsequent detailed placement. We initially consider all the components in each module as a whole. Each module is composed of a central chip and other components connected to it. Each module m is represented by a circle with a radius R_m. The radius size can be formulated as $R_m = \alpha\sqrt{Area_m}$, where α is a user-defined scaling factor to increase the module's range of influence, and $Area$ represents the sum of the areas of the components in the module. The circle center coordinates correspond to the grouped module center coordinates. In our relaxation model, we utilize the Quadratic Length $L_{ij} = (x_i - x_j)^2 + (y_i - y_j)^2$ to represent the wirelength. Here, x and y denote the x-coordinate and y-coordinate, respectively. The following formulation serves as the basis for our model:

$$F_{i,j}(x_i, y_i) = \begin{cases} w_{ij} \times L_{ij} + s_{ij} * \left(t_{ij}/L_{ij} - 1\right), & L_{ij} \leq \left(R_i + R_j\right)^2 \\ w_{ij} \times L_{ij} + t_{ij}/L_{ij} - 1, & \text{otherwise} \end{cases} \quad (13)$$

where $t_{ij} = (r_i + r_j)^2$ is the target distance of two circle modules and $s_{ij} = (r_i \times r_j)^2$ is a relatively large value, which leads modules to separate rapidly when there is excessive overlap between them. And w_{ij} as the weight assigned to the connection between module m_i and m_j. So, the objective function in this stage is formulated as

$$\begin{aligned} \min \quad & \sum_{m_i, m_j \in \mathcal{M}} F_{ij}(x_i, y_i) \\ \text{s.t.} \quad & x_i + R_i \leq W, \quad x_i - R_i \geq 0 \\ & y_i + R_i \leq H, \quad y_i - R_i \geq 0 \end{aligned} \quad (14)$$

where W, and H, represent the width and height of the outline, respectively. We solve this problem using the SLSQP library, which efficiently handles such optimization problems with constraints.

The constraints specified in (14) enforce limitations on each module to prevent it from exceeding the boundaries defined by its outline. By solving the proposed model in (14), we can derive a solution as illustrated in Fig. 11. This solution determines the approximate position of each module and establishes the relative positions between different modules. Furthermore, it defines the allowed placement range for components within each individual module. For example, in the bottom-right module shown in Fig. 11, the chip is positioned at the center of the circle. The relative positions between first-priority components and the chip remain fixed, as indicated by the red rectangle. The remaining standard components are constrained to positions within the green rectangular region shown in Fig. 11 and the exact position of the standard components will be solved by the MP-model in Section IV-D. This approach ensures the modularity of the placement results while maintaining design constraints.

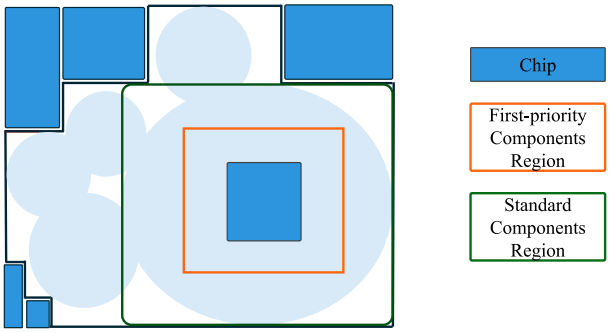


Fig. 11. Distribution of the position solving result.

After obtaining the temporary result, we need to convert and transform the components from circles back to rectangles and perform the legalization process. This requires redefining our overlap constraints to ensure that no two components can overlap, as shown in (10). Fortunately, we can obtain the relationship matrix for boolean variables $p_{i,j}$ and $q_{i,j}$ from the intermediate solution of our relaxation model. The matrix is designed to accelerate subsequent MP tasks. Given the high density of the PCB layout region, excessive constraints often lead to infeasible solutions. To address this, we do not verify all possible relationships between every pair of components. Instead, we construct a relationship matrix that captures the connections between components within each module and the

core chips in other modules. These relationships are then stored in a vector for further use.

D. Components Legalization

Afterward, we also formulated the placement legalization problem as an MP problem. Having obtained the partial set of relative relationships from the previous step, we implement rigorous segregation of module positioning based on absolute cardinal relationships (up, down, left, right) for modules without overlap, substantially accelerating MP model performance. For example, when module M_{up} is positioned superior to module M_{down} with no spatial intersection, or when module M_{left} is situated leftward of module M_{right} absent any overlap, specific positional constraints govern the spatial relationships between all constituent components belonging to these respective modules

$$y_{c_i} + w_{c_i} \times r_{c_i} + h_{c_i} \times \left(1 - r_{c_i}\right) \leq y_{c_j} \quad \forall c_i \in \boldsymbol{C}_{\text{down}} \quad \forall c_j \in \boldsymbol{C}_{\text{up}} \tag{15}$$

$$x_{c_i} + h_{c_i} \times r_{c_i} + w_{c_i} \times \left(1 - r_{c_i}\right) \leq x_{c_j} \quad \forall c_i \in \boldsymbol{C}_{\text{left}} \quad \forall c_j \in \boldsymbol{C}_{\text{right}} \tag{16}$$

where sets $\boldsymbol{C}_{\text{up}}$ and $\boldsymbol{C}_{\text{down}}$ correspond to the component collections within modules having an up-down relationship, and sets $\boldsymbol{C}_{\text{left}}$ and $\boldsymbol{C}_{\text{right}}$ represent the component collections within modules having a left-right relationship. This approach allows us to reduce the number of Boolean variables in MP and minimize the layout area for each component.

The formulation indicates the presence of nonoverlap constraints for neighboring modules, both horizontally and vertically. The boundary constraints are quite similar to those proposed in (14). However, it is important to note that the coordinates of each component, represented by x_i and y_i, correspond to the lower left coordinate, and the radius should be transformed into the width and height of the component. In addition, in our MP-based model, within each module area, the chip is placed in the center and aligned with a slight movement option *th*. The relative position relationship of the first-priority components to the chip remains unchanged. The chip and the first-priority components are prohibited from rotating during the layout process. The remaining components are placed arbitrarily within the module area. The objective function of the final placement mathematical model is to minimize a weighted sum of wirelength, which can be expressed as

$$\min \sum_{n \in \mathcal{N}} wl_n \times \alpha_n \tag{17}$$

where α_n represents the weight of each net. We will accept the solution if the optimized MP model can be solved while satisfying all the constraints. However, if the model fails to be solved successfully, we must backtrack to the preprocessing stage and relax the grouping results. At the same time, we will enlarge the chip's movable space *th* to adjust the position of the overall module. This relaxation step is intended to expand the solvable solution space, increasing the likelihood of finding a feasible solution.

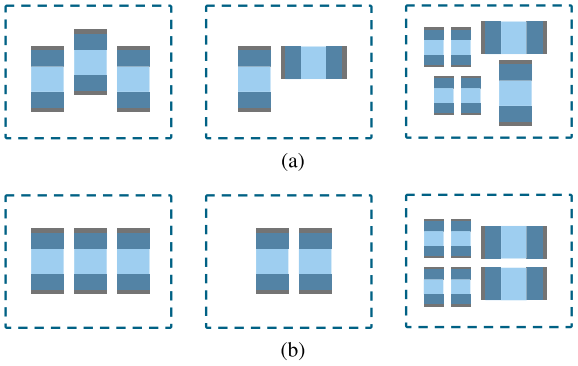


Fig. 12. Illustration of some unaligned components. (a) Before fine-tuning; (b) After fine-tuning.

Algorithme 3: Alignment-Aware Fine-Tuning

Input : A vector of components $\boldsymbol{V}$, a matrix of grids $\mathcal{G}$.
Output : A vector of updated location of components $\boldsymbol{D}$

```
repeat
    Define Alignment List
    Align ← checkAlignment(V, G);
    for blklist ∈ Align do
        Define adjustment List Candidate;
        for block ∈ blklist do
            if block.isFixed or block.isFirstPriority then
                Continue;
            Candidate.append(block);
        Adjust blocks Candidate;
        if checkOverlap(V) then
            backtrack();
        else
            block.isFixed ← TRUE;
    D ← V;
until allFixed(V) or allZeros(Align) or Timeout;
return D;
```

E. Alignment-Aware Fine-Tuning

Following legalization, we obtained a detailed placement solution that met all hard constraints. However, as illustrated in Fig. 12(a), numerous components remained unaligned, complicating the manufacturing process. To address this issue, we adopted a localized approach to alignment optimization. We divided the placement region into a grid structure and evaluated alignment within each grid cell. We assessed every component located within a grid cell's alignment relationship with other components in the same cell. We quantified misalignment using a metric misalignment count (MC), which measures the count of misaligned component pairs across all grid cells. Our objective in this stage was to minimize MC, thereby enhancing alignment quality and improving the overall placement results.

Algorithm 3 illustrated the proposed alignment-aware fine-tuning process, which iteratively adjusts the positions of components in a grid to achieve alignment while avoiding

TABLE I
PCB BENCHMARK STATISTICS

Designs	Design area(mil × mil)	Density	$\|\mathcal{M}\|$	$\|\mathcal{C}\|$	$\|\mathcal{P}\|$	$\|\mathcal{N}\|$
PCB1	200 × 240	0.58	1	13	35	12
PCB2	210 × 360	0.68	2	33	208	25
PCB3	220 × 1060	0.66	3	42	194	73
PCB4	415 × 415	0.70	2	53	194	55
PCB5	210 × 1190	0.73	3	98	332	85
PCB6	1160 × 280	0.60	11	121	139	112
PCB7	620 × 560	0.70	6	136	625	113
PCB8	680 × 570	0.65	15	141	404	139
PCB9	550 × 550	0.70	7	155	590	149
PCB10	585 × 585	0.71	8	193	903	163
PCB11	1230 × 950	0.81	14	308	2045	289
PCB12	1160 × 1440	0.80	12	364	2455	373
PCB13	1050 × 1030	0.73	23	367	1941	294
PCB14	1140 × 1000	0.71	15	399	1322	375
PCB15	940 × 1160	0.72	10	404	1518	266
bm1	510 × 230	0.43	1	13	77	34
bm2	410 × 220	0.31	2	46	163	54
bm3	580 × 600	0.21	2	54	233	35
bm4	860 × 720	0.25	1	57	314	63

overlaps. The algorithm takes as input a vector of components, $\boldsymbol{V}$, and a grid matrix, $\mathcal{G}$, and updates the location of the components. The process begins by initializing an alignment list and identifying components requiring alignment. The algorithm then iterates through these alignment groups, attempting to adjust the positions or orientations of components. It is important to note that the first-priority components remain fixed during this step to prevent unnecessary crossings of high-priority networks, as such crossings could negatively impact the placement quality. Once the components achieve a valid alignment without overlaps, it is marked as fixed to prevent further modifications. If any overlap is detected during this adjustment, the algorithm invokes a backtracking step to restore the previous state. We will select other components in this area for adjustment. The loop persists until one of three conditions is met: All components achieve perfect alignment (indicated by all zeros in ***Align***), all components become immovable, or the process reaches a predefined timeout threshold. The timeout threshold is set to 300 s in our experiment. At the end of each iteration, the updated locations of components, $\boldsymbol{D}$, are recorded. The algorithm achieves local alignment while preserving spacing and boundary constraints, making it highly effective for placement refinement applications.

V. EXPERIMENTAL RESULTS

We implemented our proposed placement algorithm in C++ language on a Linux server with 64GB of memory. Our MP-based model can be effectively solved by Gurobi [30]. We use testcases from the commercial domain to evaluate our placement framework. In this article, we test 19 different PCB benchmarks, including 15 testcases from industry and 4 testcases from academia, which are presented in Table I. Here, Density is computed as the sum of component areas divided by the available area within the irregular boundary. $|\mathcal{M}|$, $|\mathcal{C}|$, $|\mathcal{P}|$, and $|\mathcal{N}|$ represent the numbers of modules, components, pins, and nets, respectively. These industry benchmarks cover a wide range of PCB characteristics, including border shapes, density, and module types, with more diversity compared to academic benchmarks. Most industry benchmarks have a density of about 0.80, which is particularly high for PCB layout and

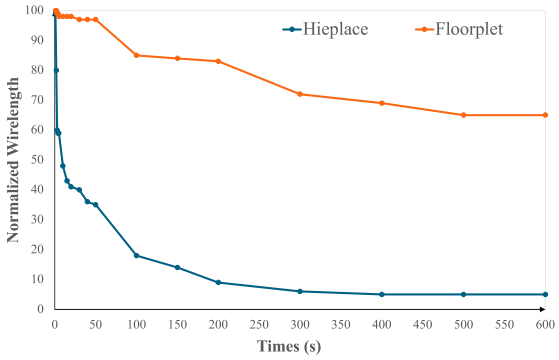


Fig. 13. Convergence curves of PCB8.

poses significant challenges. This is because our industrial PCBs come primarily from mobile phones, which require compact designs with irregular borders. This high density underscores the complexity of PCB layout and emphasizes the importance of our approach. By leveraging these real-world testcases, we aim to evaluate the performance and effectiveness of our layout framework in practical settings. In addition, we constructed four academic cases based on [31] with lower density to verify the effectiveness of our algorithm across a broader range of scenarios.

In this article, we conduct a comparative analysis between our proposed method and two established approaches, evaluating optimization across various factors. The first comparison is with the work by [25], which focuses on chiplet placement using an ILP method. To ensure a fair comparison, we incorporated our irregularity and preplaced component constraints into their method to generate feasible solutions. The second comparison involves a semi-automatic approach that combines B*tree with the SA algorithm as described by [29], supplemented with manual adjustments. In addition, we compared our method with Quilter [32], an open-source RL-based tool. Since Quilter supports constraint settings before placement, its high runtime requirements and specialization for scenarios with fewer than 100 components make it primarily suitable for academic benchmarks. Consequently, for these academic benchmarks, we specifically compared the placement results generated by our proposed method against those produced by the Quilter.

A. Convergence of Different Methods

In this section, we analyze the convergence of the proposed methods using data generated from PCB8, the benchmark consisting of 141 components. To illustrate the differences in convergence between the proposed methods, we normalize the gap between the feasible solution at each optimization step and the final optimized solution, as shown in Fig. 13. The ratio represents normalized wirelength, where a value of one indicates the initial solution and zero signifies that the weighted wirelength has reached optimal minimization and cannot be further reduced. As illustrated in Fig. 13, solutions progressively converge toward the optimal configuration over time, resulting in significantly reduced weighted wirelength values. The proposed framework demonstrates superior

TABLE II
COMPARISON OF THE PLACEMENT RESULT WITH DIFFERENT METHODS

Benchmark	[29] + Manual			[25] with Constraints				HiePlace			
	$HPWL_{all}$	Area Ratio	Cost $\times 10^3$	$HPWL_{all}$	Area Ratio	Cost	Runtime (min)	$HPWL_{all}$	Area Ratio	Cost	Runtime (min)
PCB1	656	0.99	3635	322	0.97	1546	0.03	306	0.98	1413	0.02
PCB2	812	0.97	5051	614	0.97	3323	0.07	703	0.97	3718	0.06
PCB3	5889	0.83	24969	5895	0.82	25029	30	5886	0.83	24939	5
PCB4	5001	0.98	21426	3756	0.97	11271	30	3438	0.97	9450	5
PCB5	9505	0.95	45829	7302	0.94	24123	40	7223	0.95	24044	20
PCB6	8807	0.81	17179	5662	0.81	12763	60	5222	0.82	12323	10
PCB7	9194	0.96	47327	4910	0.94	20453	120	4300	0.94	19843	10
PCB8	12789	0.93	41661	9398	0.88	18326	120	9216	0.88	18144	15
PCB9	8700	0.93	39066	5493	0.92	19515	240	4855	0.94	15961	20
PCB10	14921	0.99	50111	19014	0.97	6495	240	6252	0.97	18723	30
PCB11	64705	0.98	266521	N/A	N/A	N/A	240	22009	0.95	73462	60
PCB12	62302	0.97	181387	N/A	N/A	N/A	240	29393	0.95	62540	60
PCB13	102661	0.92	336814	N/A	N/A	N/A	240	46424	0.91	83018	90
PCB14	47680	0.93	214675	N/A	N/A	N/A	240	22730	0.91	45445	120
PCB15	45804	0.96	159582	N/A	N/A	N/A	240	24549	0.95	81978	120
Ratio	1	1	1	0.65	0.98	0.52	1	**0.48**	0.99	**0.34**	**0.13**

convergence speed compared to [25], reaching near-optimal solutions within 10 min. In contrast, the previous method shows a substantially slower convergence rate, except for the initial stage, where no apparent changes were observed in the convergence curve.

B. Comparison of the Different Methods

The experimental results are shown in Table II, where **$HPWL_{all}$**, **Area Ratio**, **Cost**, **Runtime** represent the estimated wirelength based on the half perimeter wirelength model, the area ratio, the weighted wirelength which considers the connection priority, the count of the alignment path, and the runtime, respectively. *AreaRatio* is calculated by a sweepline algorithm, which is the ratio of the bounding box area occupied by the components to the outline area. *Cost* is calculated based on (17). Since the problem solved in this work is complicated, it is difficult for solutions to converge to the final optimized solution. Therefore, the runtime of the MP solver is limited in generating the best-effort solutions. The runtime limitation of [25] is 4 h. The number of components determines the running time of our proposed method, and even the most significant cases can achieve satisfactory results in less than 2 h. In addition, due to the need for engineer support, we did not calculate the runtime for tree-based methods.

As shown in Table II, the MP-based algorithm in [25] achieves significantly shorter wirelength than other baselines but fails to produce feasible solutions for larger benchmarks (PCB11-PCB15) within the runtime limit, revealing scalability limitations. In contrast, HiePlace consistently outperforms prior methods across most of the testcases, delivering superior solution quality with shorter wirelength, lower cost, and faster execution. On average, it reduces cost by 66% and 18% compared to the two baselines while requiring only 13% of their runtime to achieve better results. The speedup is due to our solution space reduction strategy, which ensures efficient search without compromising quality.

In addition, we compare our approach with the Quilter automated tool. We evaluated our approach against a widely used open-source tool, Quilter [32], across four academic benchmarks (bm1-bm4). Figs. 14 and 15 present the cost and runtime comparisons, respectively.

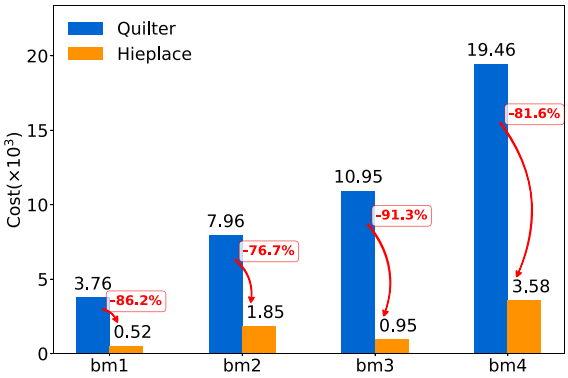


Fig. 14. Comparison of cost between Quilter [32] and Hieplace.

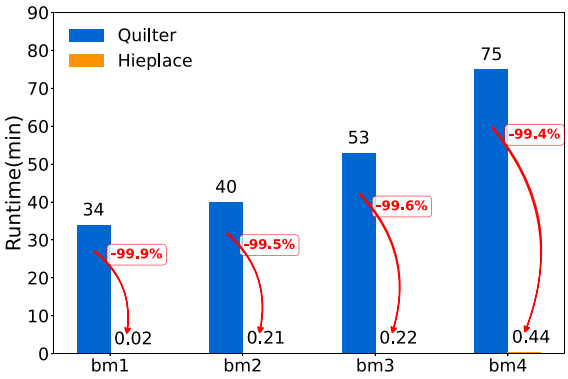


Fig. 15. Comparison of runtime between Quilter [32] and Hieplace.

Fig. 14 illustrates the cost comparison between the two methods. Hieplace framework achieves significantly lower costs across all academic benchmarks, with reductions ranging from 76.7% to 91.3%. For the largest benchmark (bm4), Hieplace reduces the cost from 19.46×10^3 to just 3.58×10^3, representing an 81.6% improvement. These cost reductions directly translate to better PCB designs with improved wirelength and resource utilization.

As shown in Fig. 15, our Hieplace framework demonstrates dramatic runtime improvements across all benchmarks. The runtime reduction ranges from 99.4% to 99.9% compared to

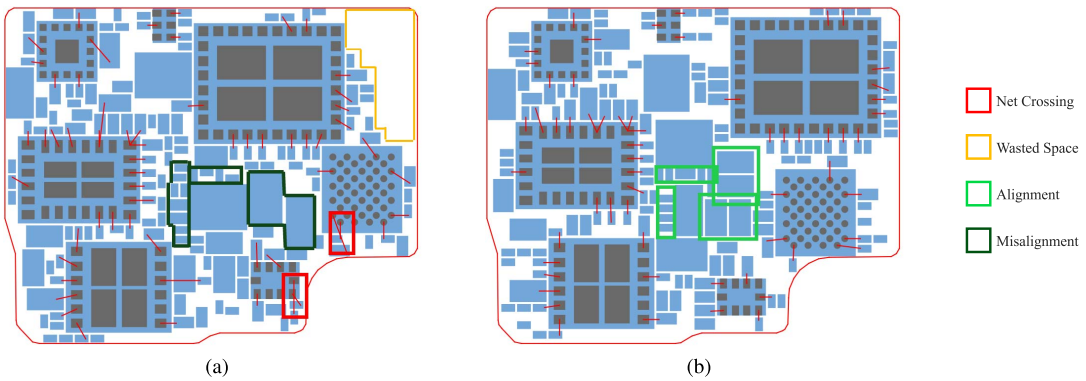


Fig. 16. Comparison of placement results on PCB9. (a) Floorplet [25]. (b) HiePlace.

TABLE III
COMPARISON OF THE ROUTABILITY METRICS

Benchmark	[25] with Constraints			HiePlace		
	$HPWL_{first}$	MC	Crossing	$HPWL_{first}$	MC	Crossing
PCB1	198	2	0	198	2	0
PCB2	335	12	0	301	10	0
PCB3	2126	8	5	2117	7	2
PCB4	912	17	3	835	9	0
PCB5	2018	29	9	1947	13	2
PCB6	1259	31	4	1176	5	0
PCB7	2460	37	2	2033	11	2
PCB8	2106	42	2	1406	30	0
PCB9	1528	69	2	1378	32	0
PCB10	2690	59	4	1691	45	0
Ratio	1	1	1	**0.84**	**0.54**	**0.20**

Quilter. Specifically, while Quilter requires 34–75 min for the benchmarks, Hieplace completes the same tasks in just a few seconds. This exceptional speedup demonstrates the efficiency of our approach, making it particularly suitable for more dense and large-scale industrial applications.

The consistent performance improvements across various benchmarks demonstrate the robustness and effectiveness of our approach. The combined advantages of dramatically reduced runtime and significantly lower costs make Hieplace particularly valuable for complex industrial PCB design scenarios where both efficiency and quality are critical considerations.

C. Comparison of the Routability

In this section, we compare our approach with Floorplet on key metrics affecting routability. The comparison is presented in Table III, where $HPWL_{first}$, MC, and Crossing represent the estimated wirelength for first-priority components and core chips, the count of misaligned regions, and the number of net crossings between the chip and the first-priority components. We compared the solution on the first 10 PCB designs, which have feasible placement results. The results indicate that our method effectively minimizes net crossings and reduces wirelength. As shown in Table III, compared to the baseline, our approach achieves an average reduction of 16% in wirelength and 46% in misaligned regions while effectively minimizing the impact of line crossings. This shows that our method has a significant effect on the layout of first-priority components. At the same time, in terms of regularity, our method has a 46% reduction in nonaligned area comparison, which means that our layout results meet the alignment requirements in more areas. As a result, we substantially improve the routability of the chip. This improvement is primarily attributed to the reduction in the number of net crossings and wirelength, as well as in terms of regularity.

Fig. 16 shows the results of PCB9 using different algorithms. Our method offers clear advantages in wirelength and routability. For example, the highest-priority net, represented by the red line, is directly connected without crossings, resulting in a shorter wirelength. In contrast, as shown in the red rectangle in Fig. 16(a), the net has multiple crossings, a longer wirelength, and wasted space, as shown in the orange region, which negatively impacts routability. In comparison, our method achieves better area utilization and improved modularity. Furthermore, as the green region indicates, our approach greatly enhances alignment, with more components placed regularly. This improvement reduces manufacturing complexity and increases efficiency. Overall, the solution generated by HiePlace outperforms previous methods in multiple aspects.

VI. DISCUSSION

While our current approach significantly improves wirelength and efficiency of PCB placement while satisfying multiple constraints, several important aspects of PCB design remain unaddressed. The most critical among these are routability evaluation and double-sided layout.

Regarding routability evaluation, we currently view it as a superposition of different indicators, such as net intersection and alignment. In the future, we plan to develop a comprehensive routability model based on various features to guide our layout process and quantify routability metrics more effectively. In addition, we could implement quick routing feasibility checks on key networks after each layout decision and incorporate this feedback into the reward calculation.

For double-sided layouts, our framework could be extended to treat the PCB as two interconnected planes, expanding the state space to include component placement on either the front or back sides. The reward function would require additional weights to account for interlayer connection costs, with particular attention to minimizing the number of vias traversing the PCB. The enhanced reliability offered by double-sided layouts represents another important

consideration. Furthermore, additional aspects, such as arbitrary component rotation angles and thermal and mechanical considerations, could be incorporated into subsequent layout optimizations.

VII. Conclusion

In this article, we presented HiePlace, an efficient MP-based placement framework tailored for PCBs. We addressed the challenges posed by diverse constraints and the unique characteristics inherent to PCB designs. Our framework introduced two innovative acceleration techniques: DP for first-priority component placement and a relaxation algorithm that minimizes Boolean variables while effectively reducing the solution space. Experimental results demonstrated significant improvements in both runtime and wirelength reduction compared to existing methods. HiePlace also successfully optimized PCB-specific design metrics, validating its effectiveness in automating PCB placement while satisfying complex design requirements.

References

[1] Y. Lin, S. Dhar, W. Li, H. Ren, B. Khailany, and D. Z. Pan, "DREAMPlace: Deep learning toolkit-enabled GPU acceleration for modern VLSI placement," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, 2019, pp. 1–6.

[2] X. He, Y. Wang, Y. Guo, and E. F. Young, "Ripple 2.0: Improved movement of cells in routability-driven placement," *ACM Trans. Design Autom. Electron. Syst.*, vol. 22, no. 1, pp. 1–26, 2016.

[3] Z. Cai et al., "PUFFER: A routability-driven placement framework via cell padding with multiple features and strategy exploration," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, 2023, pp. 1–6.

[4] Y. Pu et al., "IncreMacro: Incremental macro placement refinement," in *Proc. ACM Int. Symp. Phys. Design (ISPD)*, 2024, pp. 169–176.

[5] 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. IEEE/ACM Asia South Pacific Design Autom. Conf. (ASPDAC)*, 2021, pp. 170–175.

[6] D. Zhang, Q. Ren, and D. Su, "A novel authentication methodology to detect counterfeit PCB using PCB trace-based ring oscillator," *IEEE Access*, pp. 28525–28539, 2021.

[7] Y.-H. Cheng, D.-W. Huang, W.-K. Mak, and T.-C. Wang, "A practical detailed placement algorithm under multi-cell spacing constraints," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2018, pp. 1–8.

[8] J.-M. Lin and Z.-X. Hung, "UFO: Unified convex optimization algorithms for fixed-outline Floorplanning considering pre-placed modules," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 7, pp. 1034–1044, Jul. 2011.

[9] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen, and Y.-W. Chang, "A high-quality mixed-size analytical placer considering preplaced blocks and density constraints," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, 2006, pp. 187–192.

[10] C.-H. Chiou, C.-H. Chang, S.-T. Chen, and Y.-W. Chang, "Circular-contour-based obstacle-aware macro placement," in *Proc. IEEE/ACM Asia South Pacific Design Autom. Conf. (ASPDAC)*, 2016, pp. 172–177.

[11] R. Liu, X. Hong, S. Dong, Y. Cai, and J. Gu, "VLSI/PCB placement with predefined coordinate alignment constraint based on sequence pair," in *Proc. IEEE Int. Conf. ASIC (ASICON)*, 2001, pp. 167–170.

[12] Y. Sheng, A. Takahashi, and S. Ueno, "Relay-race algorithm: A novel heuristic approach to VLSI/PCB placement," in *Proc. IEEE Annu. Symp. VLSI (ISVLSI)*, 2011, pp. 96–101.

[13] D. Bertsimas and J. Tsitsiklis, "Simulated annealing," *Stat. Sci.*, vol. 8, no. 1, pp. 10–15, 1993.

[14] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu, "B*-trees: A new representation for non-slicing floorplans," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, 2000, pp. 458–463.

[15] A. Vidal-Obiols, J. Cortadella, J. Petit, M. Galceran-Oms, and F. Martorell, "Multilevel dataflow-driven macro placement guided by RTL structure and analytical methods," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 12, pp. 2542–2555, Dec. 2021.

[16] J. Chen, Y.-W. Chang, and Y.-C. Huang, "Analytical placement considering the electron-beam fogging effect," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 3, pp. 560–573, Mar. 2021.

[17] F. S. Ismail, R. Yusof, and M. Khalid, "Optimization of electronics component placement design on PCB using self organizing genetic algorithm (SOGA)," *J. Intell. Manuf.*, vol. 23, pp. 883–895, Jun. 2012.

[18] N. Quinn and M. Breuer, "A forced directed component placement procedure for printed circuit boards," *IEEE Trans. Circuits Syst.*, vol. 26, no. 6, pp. 377–388, Jun. 1979.

[19] Y. Lai, Y. Mu, and P. Luo, "MaskPlace: Fast chip placement via reinforced visual representation learning," in *Proc. Annu. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2022, pp. 24019–24030.

[20] R. Cheng, X. Lyu, Y. Li, J. Ye, J. Hao, and J. Yan, "The policy-gradient placement and generative routing neural networks for chip design," in *Proc. Annu. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2022, pp. 26350–26362.

[21] S. Yue et al., "Scalability and generalization of circuit training for chip floorplanning," in *Proc. ACM Int. Symp. Phys. Design (ISPD)*, 2022, pp. 65–70.

[22] R. Cheng and J. Yan, "On joint learning for solving placement and routing in chip design," in *Proc. Annu. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2021, pp. 16508–16519.

[23] Y. Lai, J. Liu, Z. Tang, B. Wang, J. Hao, and P. Luo, "ChiPFormer: Transferable chip placement via offline decision transformer," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2023, pp. 18346–18364.

[24] Z. Zhuang, B. Yu, K.-Y. Chao, and T.-Y. Ho, "Multi-package co-design for chiplet integration," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2022, pp. 1–9.

[25] S. Chen et al., "Floorplet: Performance-aware floorplan framework for chiplet integration," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 43, no. 6, pp. 1638–1649, Jun. 2024.

[26] M.-C. Wu and Y.-W. Chang, "Placement with alignment and performance constraints using the B*-tree representation," in *Proc. IEEE Int. Conf. Comput. Design (ICCD)*, 2004, pp. 568–571.

[27] S. Chen, S. Dong, X. Hong, Y. Ma, and C.-K. Cheng, "VLSI block placement with alignment constraints," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 53, no. 8, pp. 622–626, Aug. 2006.

[28] C. Luo, M. F. Anjos, and A. Vannelli, "Large-scale fixed-outline floorplanning design using convex optimization techniques," in *Proc. IEEE/ACM Asia South Pacific Design Autom. Conf. (ASPDAC)*, 2008, pp. 198–203.

[29] T.-C. Chen and Y.-W. Chang, "Modern floorplanning based on B*-tree and fast simulated annealing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 4, pp. 637–650, Apr. 2006.

[30] "Gurobi optimizer," 2024. [Online]. Available: http://www.gurobi.com/.

[31] "ASPDAC-submission-PCB-layout/PCBBenchmarks." 2020. [Online]. Available: https://github.com/aspdac-submission-pcb-layout/PCBBenchmarks/tree/master

[32] "Quilter 2025," 2025. [Online]. Available: https://www.quilter.ai/.

Shanyi Li received the B.Eng. degree from Tsinghua University, Beijing, China, in 2019. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, under the supervision of Prof. Tsung-Yi Ho.

His current research interest is electronic design automation (EDA), especially EDA for 2.5-D/3-D packaging, and PCB design.

Zhen Zhuang received the M.Eng. and B.Eng. degrees from Fuzhou University, Fuzhou, China, in 2021 and 2018, respectively, and the Ph.D. degree from The Chinese University of Hong Kong, Hong Kong in 2025, under the supervision of Prof. Tsung-Yi Ho.

His current research interest is electronic design automation, especially EDA for advanced packaging, and 3-D IC.

Dr. Zhuang was a recipient of three ICCAD/ISPD contest awards.

Mingyu Liu is currently a Senior Technical Expert with Huawei Cloud Computing Technologies Company Ltd., Guiyang, China. His research interests include the integration of integrated circuit design and cloud computing technologies, focusing on advancing industrial software innovation in cloud environments.

Dr. Liu is also a Senior Member of China Computer Federation (CCF) and serves as an Executive Committee Member for the CCF Technical Committee on Integrated Circuit Design.

Weihua Sheng received the B.Eng. degree from Tsinghua University, Beijing, China, in 2000, and the M.S. and Ph.D. degrees from RWTH Aachen University, Aachen, Germany, in 2003 and 2010, respectively.

He is currently with Huawei Research Center, Hong Kong. He has spearheaded the technology development on multicore programming tools with RWTH Aachen University from 2007 to 2013, which later turned into the foundation of Silexica. His early career roots are in the electronic design automation industry. He has a proven record of successful consultation and collaboration with top-tier technology companies on multicore design tools.

Bei Yu (Senior Member, IEEE) received the Ph.D. degree from The University of Texas at Austin, Austin, TX, USA, in 2014.

He is currently a Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

Prof. Yu received eleven Best Paper Awards from ICCAD 2024 & 2021 & 2013, IEEE TSM 2022, DATE 2022, ASPDAC 2021 & 2012, ICTAI 2019, Integration, the VLSI Journal in 2018, ISPD 2017, SPIE Advanced Lithography Conference 2016, six ICCAD/ISPD contest awards, the IEEE CEDA Ernest S. Kuh Early Career Award in 2021, the DAC Under-40 Innovator Award in 2024, and the Hong Kong RGC Research Fellowship Scheme Award in 2024. He has served as the TPC Chair for *ACM/IEEE Workshop on Machine Learning* for CAD in 2019, International Symposium of EDA in 2025, and in many journal editorial boards and conference committees.

Tsung-Yi Ho received the Ph.D. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2005.

He is a Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong. His research interests include design automation and test for microfluidic biochips and nanometer-integrated circuits.

Prof. Ho has been the recipient of the Invitational Fellowship of the Japan Society for the Promotion of Science, the Humboldt Research Fellowship by the Alexander von Humboldt Foundation, and the Hans Fischer Fellow by the Institute of Advanced Study of the Technical University of Munich. He was a recipient of the Best Paper Awards at the VLSI Test Symposium in 2013 and IEEE TCAD in 2015. He currently serves as an Associate Editors for the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, *ACM Transactions on Design Automation of Electronic Systems*, *ACM Journal on Emerging Technologies in Computing Systems*, *ACM Transactions on Embedded Computing Systems*, and IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, and the Technical Program Committees of DAC, ICCAD, DATE, and ASP-DAC.