Hybrid Detour Refinement Strategy for Package Substrate Routing

Ding-Hsun Lin[†] Tsubasa Koyama[†] Yu-Jen Chen[†]
Keng-Tuan Chang[§] Chih-Yi Huang[§] Chen-Chao Wang[§] Tsung-Yi Ho[‡]
Department of Computer Science, National Tsing Hua University, Taiwan[†]
Product Design Division, Advanced Semiconductor Engineering, Inc., Taiwan[§]
Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong[‡]

ABSTRACT

Advanced packaging technologies have gained significant importance in recent years due to rapid technological advancements. In these designs, substrate routing plays a critical role in ensuring proper functioning and performance. Although existing automatic routing tools are available to assist designers in solving routing problems, they often produce suboptimal results or design rule violations (DRVs) when dealing with the complex constraints and specifications found in industrial designs. As a consequence, designers must dedicate weeks to optimizing and correcting the results generated by these automatic routing tools. In this work, a hybrid detour refinement strategy that combines rule-based and deep learning (DL)-based approaches is proposed to address these challenges. The aim is to reduce detours and improve area distribution in the auto-routing results of industrial Flip-Chip Ball Grid Array (FCBGA) substrate designs while decreasing the time required for modifications. Experimental results demonstrate that the proposed methods effectively enhance both detour reduction and area distribution, achieving an average improvement of 43% and 33%, respectively, compared to the auto-routed design. Furthermore, the modification time is also drastically reduced from weeks to minutes.

KEYWORDS

Advanced Packaging, Package Substrate Routing, Routing Refinement, Deep Learning

1 INTRODUCTION

In recent years, advanced packaging technologies have become essential in meeting the demands of modern electronics, driven by rapid technological advancements. In these designs, substrate routing plays a vital role in ensuring the functionality and performance of package designs. It involves two sub-stages: escape routing and area routing. In escape routing, the nets are routed from the internal pins of the chip to the chip boundary. Subsequently, area routing is employed to connect these boundary points to the target pins. However, gaps may exist between these two sub-stages, and certain existing approaches [1–3] address these issues by routing these sub-stages concurrently. In addition, some previous works have incorporated artificial intelligence (AI) techniques, such as deep learning (DL) and reinforcement learning (RL), to achieve improved routing outcomes [4, 5]. These AI-based approaches have shown significant enhancements in efficiency, accuracy, and adaptability, which are increasingly critical in modern complex chip designs.

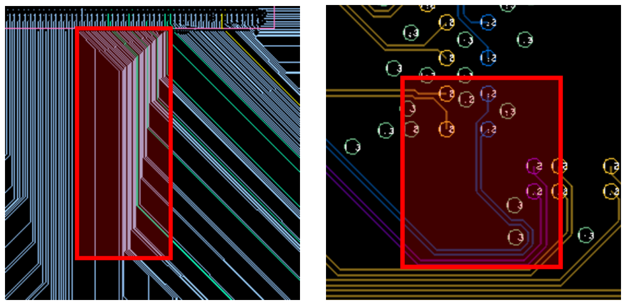
(a) (b)

Figure 1: Visualization of the (a) dense routing area and the (b) detoured nets.

Although existing works and automatic routing tools assist designers in quickly addressing routing problems, they often produce suboptimal results due to the complexity of constraints and specifications in industrial designs. For instance, dense routing areas and detours in the layout need to be modified to enhance performance, as shown in Figure 1(a) and (b), respectively. Consequently, designers frequently spend several weeks manually adjusting the initial routing results to meet specific design requirements. Thus, reducing the time required for manual modifications is a critical challenge throughout the entire design cycle.

Several previous works have specifically addressed the refinement of initial routing results [6–8]. However, it is worth noting that many of the previous works mentioned, such as [1–5] and [7], primarily employ grid-based algorithms. While these algorithms have shown effectiveness in certain cases, they can be time-consuming when applied to large industrial scenarios. The router proposed in [6] achieves favorable area distribution within a shorter runtime. However, it reroutes the initial results before the diffusion process, which still requires considerable time. Moreover, while it alleviates congestion, it also increases the occurrence of detours. In the proposed rerouting framework in [7], it aims to achieve a manual-like design that reduces detours and improves routing congestion to

ASPDAC '25, January 20–23, 2025, Tokyo, Japan
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0635-6/25/01
https://doi.org/10.1145/3658617.3697732

some extent. However, because the processes are performed locally within fixed areas, they cannot interact with adjacent areas to solve problems on a global scale. As for the methodology proposed by [8], it successfully refines detours and dense routing areas in auto-routed designs. Nonetheless, they result in a less effective process and inferior performance in addressing detours. They proposed the *DNN Guided Routing Method*, which utilizes a neural network to predict the routing direction, either clockwise or counterclockwise, to refine the detoured nets. Unfortunately, they lack feasibility in solving complicated detours with limited routing area. Moreover, they may exacerbate the congestion in dense routing areas, significantly hindering the effectiveness of subsequent optimization.

In this work, we proposed a novel hybrid routing refinement approach that utilizes rule-based and DL-based methods to replace the *DNN Guided Routing Method* proposed in [8]. In addition, different from the previous approach [8], we remove the iterative execution process, significantly reducing the execution time for refining detours. As shown in Figure 2, we follow [8] for the previous stages before detour refinement. After detecting all potential detours, we first refine the significant detours using the rule-based *Riverside Routing Method* by rerouting along the surrounding segments. After completing the preliminary refinement, to achieve refinement of the remaining detours, we introduce the Decision Transformer (DT) [9] RL strategy for routing refinement, the DL-based *Decision Transformer Routing Method*. Different from the conventional RL strategy [10], DT can be trained directly using existing training data (i.e., offline training), eliminating the excessive time spent by conventional RL in the process of exploration and exploitation. Moreover, DT can explore past actions through self-attention and combine prior experience to determine future actions, which is not achievable by conventional RL as it only evaluates based on reward values. Additionally, by considering multiple actions, the direction and the length, simultaneously in each step, the proposed method successfully increases the routability compared to previous work. Lastly, we introduce some post-processing techniques to assist in optimizing the results of the DT, ensuring that the modified outcomes align more closely with those achieved through manual modification. Our contributions can be summarized as follows:

- We utilize a multifaceted approach, incorporating both rule-based and DL-based methods for detour refinement. This approach combines the precision of the rule-based method with the adaptability of the DL-based method.
- We propose a novel *Decision Transformer Routing Method* and utilize several gridless methods to refine the auto-routed results, enabling it to handle large industrial cases.
- To the best of our knowledge, this is the first work combining the Decision Transformer into the field of circuit routing, and it also exhibits remarkable flexibility and reliability.
- Experimental results demonstrate significant improvements in both detour reduction and area distribution compared to the baseline auto-routed design, and the modification time is drastically reduced to minutes.

The rest of the paper is organized as follows. Section 2 provides preliminaries. Section 3 outlines the details of the proposed detour refinement approach. Section 4 presents the experimental results. Finally, Section 5 provides the concluding remarks for this work.

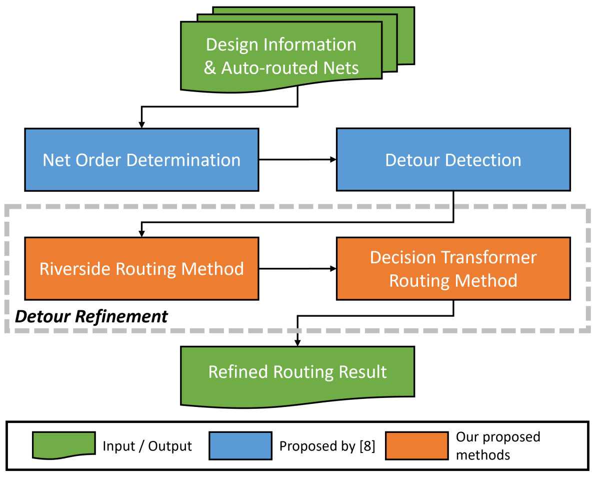

Figure 2: Overall flow of the proposed refinement.

2 PRELIMINARIES

In this section, the problem formulation is first given, followed by a description of the proposed refinement flow.

2.1 Problem Formulation

Given an auto-routed design comprising netlist, pin locations, pin sizes, and segment details, the main objective is to refine the detours and dense routing areas while satisfying specific constraints. These constraints include avoiding an increase in wirelength, preventing shorts (both net-to-net and pin-to-net shorts) and sharp angles.

2.2 Overall Refinement Flow

Figure 2 illustrates the proposed refinement flow. We follow previous work [8] to detect all potential detours from the original auto-routed design. After obtaining all detoured segments, we further apply the proposed rule-based *Riverside Routing Method* and the DL-based *Decision Transformer Routing Method* to refine all detours. Finally, the refined routing result is obtained.

3 DETOUR REFINEMENT

To facilitate effective detour modification, we first employ the *Riverside Routing Method* to all detour segments, providing sufficient space for subsequent refinement. The *Riverside Routing Method* demonstrates remarkable effectiveness when applied to a specific shape of detours designated as significant detours, as shown in Figure 3(a). These detours are characterized by over five continuous turns in the same direction. Although the remaining detours, named slight detours, as presented in Figure 3(b) and (c), are also refined by the *Riverside Routing Method*, they are not properly solved. Consequently, we propose the *Decision Transformer Routing Method* to complete the entire refinement process.

3.1 Riverside Routing Method

Targeting the significant detours, we propose the *Riverside Routing Method* to ensure smooth and high-quality optimization. Figure 4 illustrates an example of the *Riverside Routing Method*. The algorithm

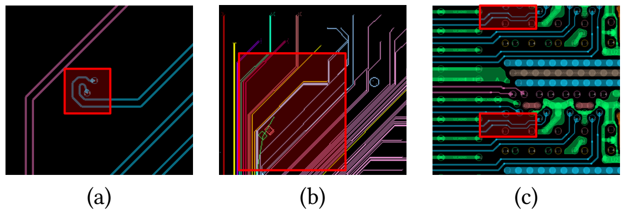

Figure 3: Different types of detours highlighted in red squares. (a) Significant detours. (b) Slight detours within a large area. (c) Slight detours within a small area.

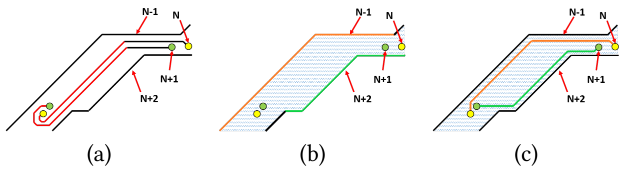

Figure 4: Illustration of the *Riverside Routing Method.* (a) The red lines indicate the significant detoured segments, the numbers N-1, N, N+1, and N+2 represent the net order of each net, and the yellow and green dots represent the pins of net N and net N+1, respectively. (b) Removing net N and net N+1 and extracting segments from net N-1 and net N+2. (c) The result after using *Riverside Routing Method.*

starts with removing the nets that are involved in detected detours (Figure 4(a)). Next, we identify the adjacent nets of the removed nets, and extract their segments. For example, in Figure 4(b), the orange and the green segments are the extracted segments. After the extraction, we reposition them closer to the pins of the removed nets while adhering to a 135-degree angle constraint. Subsequently, the pins of the removed nets are rerouted along neighboring nets, much like a river flowing along its border. Figure 4(c) illustrates an example where net N is rerouted along N-1 (indicated by the orange line), and net N+1 is rerouted along N+2 (indicated by the green line). By employing this approach, significant detours are effectively handled, ensuring connectivity and adherence to design constraints.

3.2 Decision Transformer Routing Method

After efficiently modifying all significant detours using the *Riverside Routing Method*, we proceed to address the remaining slight detours. In this stage, we adopt DT as our main routing framework. First, we introduce the preparation process for the training data. Next, we discuss the setting of the RL routing environment for model interaction. We then introduce our proposed DT architecture to deal with the routing problem on the gridless designs. Eventually, we propose some post-processing techniques to improve the routability.

3.2.1 Training Data Preparation. The training data is extracted from the manually-modified design provided by the Advanced Semiconductor Engineering (ASE) Group. Initially, we exclude the segments involved in the snake shape that are routed to meet the length-matching constraint [11], as these nets fall outside the purview of

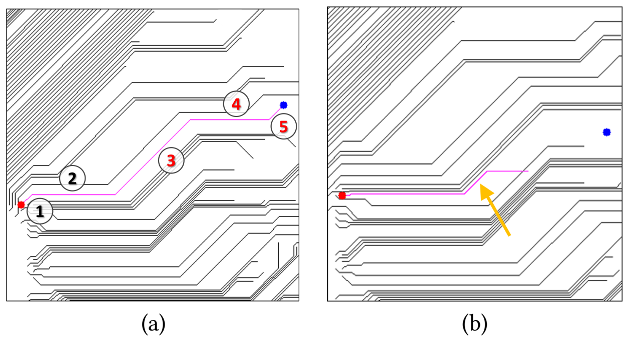

Figure 5: Image visualization while handling the Training Data Preparation. (a) The original net with five segments, which are labeled from 1 to 5. Extract segment 3 to segment 5 to form new data. (b) Randomly shorten the pointed segment to generate failure data.

our rerouting goals. For the remaining nets, we perform a random continuous segments extracting algorithm to obtain augmented training data. As shown in Figure 5(a), an original net containing five segments allows us to randomly select multiple consecutive segments to form new training data (e.g., segment 3 to segment 5). In addition to considering successful cases, we also incorporate failure cases into the training data to enhance the model's robustness. As shown in Figure 5(b), we randomly lengthen or shorten segments within a net to create failure cases. In summary, after initially removing segments with special purposes, the number of nets and segments is 837 and 9,036, respectively. Subsequently, by extracting partial consecutive segments for augmentation and adjusting their lengths to generate failure cases, the overall number of nets and segments increases to 118,017 and 552,201, respectively.

3.2.2 RL Routing Environment. State, action, and reward are essential elements in the RL routing environment, each of which will be explained separately below.

State Space: In our RL environment, we utilize images to represent the status at each time step. We extract a larger square region centered on the target net as the routable area. Besides the target net itself, the surrounding segments, which are treated as obstacles, are also encompassed in the state image. Therefore, each state image comprises four distinct components: the current point, the end points of the target net, the routed segments, and the obstacles. For better recognition, we assign different colors to each component.

Action Space: Since our approach focuses on the gridless routing problem, the proposed method utilizes a hybrid action space to combine discrete action and action with continuous value. As demonstrated in Figure 6, for each segment, we decompose it into two actions: a discrete value representing the direction and a continuous value representing the length. Due to the constraints of substrate routing, the routing direction can only be multiples of 45 degrees, which denotes a total of 8 possible directions, labeled from 0 to 7, as shown in Figure 6(a). As for the length, we assume the side length of the squared state image as 1 and normalize the length of the segments within the image. For example, as shown in Figure 6(b), the net starts from the green point and terminates at the blue

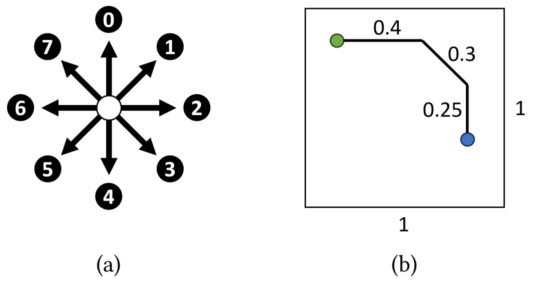

Figure 6: Action space of the RL routing environment. (a) Eight possible routing directions. (b) Demonstration of representing the net as a list of actions. The net starts at the green point and ends at the blue point. The segments of the net can be expressed as [(2, 0.4), (3, 0.3), (4, 0.25)].

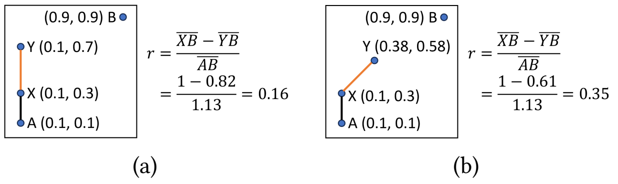
(0.9, 0.9) B
Y (0.1, 0.7)
X (0.1, 0.3)
A (0.1, 0.1)
$r = \frac{\overline{XB} - \overline{YB}}{\overline{AB}} = \frac{1 - 0.82}{1.13} = 0.16$
(a)
(0.9, 0.9) B
Y (0.38, 0.58)
X (0.1, 0.3)
A (0.1, 0.1)
$r = \frac{\overline{XB} - \overline{YB}}{\overline{AB}} = \frac{1 - 0.61}{1.13} = 0.35$
(b)

Figure 7: Application of the reward function. In both scenarios, the segments $\overline{XY}$ have equal lengths but different directions. (a) The segment with direction 0 has a reward of 0.16. (b) The segment with direction 1 has a reward of 0.35.

point. The first segment is going to the right, which corresponds to a direction of 2, and its length would be 0.4 units relative to the side length of the image. Thus, the action of the segment could be represented as (2, 0.4). Following the same definition, the actions of these three segments could be written as [(2, 0.4), (3, 0.3), (4, 0.25)].

Reward Function: After each action is taken, we evaluate its contribution to the goal. Let the start and the end point of the target net be denoted as A and B, respectively. After each action, we obtain a new segment with the start point X and the end point Y. We define the reward function as follows:

$$r(A, B, X, Y) = \frac{\overline{XB} - \overline{YB}}{\overline{AB}} + \begin{cases} V, & \text{if success} \\ -V, & \text{if failure} \\ 0, & \text{otherwise.} \end{cases}$$

For the denominator, we compute the Euclidean distance of the target net, which serves as a normalization method to facilitate comparisons between different nets. The numerator is calculated as the difference between the Euclidean distances from the start and end points of the segment to the end point of the entire net, which reflects the contribution to the goal given by the current action. Moreover, our reward function is tailored to effectively capture the impact of direction. As shown in Figure 7, despite having the almost identical length of the segment in case (a) and case (b), our reward function yields a higher reward in case (b), since it is much closer to the target end point B. Eventually, if the routing succeeds, which means the distance between the end point Y of the segment and the end point B of the target net falls within a small margin, we add an additional reward V to the reward function. Conversely, while the routing fails, which indicates that the segment either intersects with existing segments or nets, or the end point Y of the segment lies outside the routable area, we assign an additional negative reward $-V$.

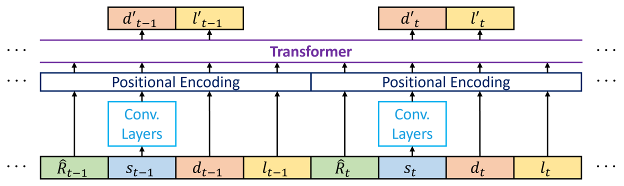

Figure 8: The proposed DT architecture and the input sequence, including returns-to-go $\widehat{R}$, states s, direction actions d, and length actions l.

3.2.3 *DT Model Architecture.*

DT is a novel approach to RL that leverages sequence modeling techniques. We follow existing approach [9] to treat the decision-making process as a sequence of states s, actions a, and returns-to-go $\widehat{R}_t = \sum_{t'=t}^{T} r_{t'}$, which is the sum of all future rewards r from the current time step t to the future time step T, and utilizes a transformer architecture [12] to learn the optimal policy. However, the existing DT uses different image sizes as input states and can only handle a single action, not multiple actions. Therefore, as shown in Figure 8, we propose a new DT model. We adjust multiple convolution layers in order to match the sizes of the input state images. In addition, we replace the single action with two actions: discrete action (direction d) and continuous action (length l). By making these changes, we can utilize the DT architecture to handle hybrid actions.

3.2.4 *Training & Testing Process.*

During the training stage, we feed the full sequence with all time steps, which includes states, actions, and returns-to-go, into the model. Subsequently, the model predicts two actions for each time step. To optimize the model, we apply cross-entropy loss and mean squared error for discrete action and continuous action, respectively. Finally, we utilize a tunable hyperparameter to balance the two losses.

As for the testing stage, different from the training process, in which we have the information of all time steps, the model can only predict action for the current time step based on historical states, actions, and returns-to-go. Thus, we iteratively predict the action until the environment terminates. Besides, since the proposed DT architecture sequentially generates two actions, we can validate the legality of the first direction action before deciding on the second length action. Considering the constraints of substrate routing in this paper, sharp angles are prohibited, so the angle between two segments should be 135 degrees. Therefore, after the DT predicts the direction and length, we first judge the legality of the direction. If it is valid, we directly use the predicted direction and length as the result of the current time step. Otherwise, we select the legal direction with the highest probability as the new direction. Subsequently, we put this new direction back into the DT to predict

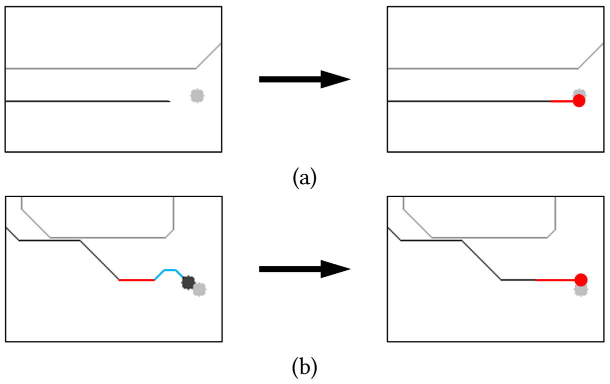

Figure 9: Visualization of early termination. (a) By adjusting the length of the last segment, the routing can connect and finish earlier. (b) An example demonstrates the contrast before and after implementing early termination. This technique helps avoid unnecessary bends (colored in blue) by extending the fourth-to-last segment (colored in red).

the corresponding length. Consequently, we successfully meet the constraint and increase the accuracy of the DT model.

3.2.5 Post-processing Techniques. We propose three post-processing techniques to improve the success rate and efficiency of the DT routing refinement while avoiding the generation of excessive meaningless segments.

Test-time Image Rotation: During the Training Data Preparation, we refrained from using image rotation for data augmentation because the nature of substrate routing inherently involves routing in various directions. However, in the testing process, we incorporate image rotation to enable the model to explore more possibilities and record successful routing results. We rotate the image clockwise at four angles: 0, 90, 180, and 270 degrees, and conduct routing tasks for each rotation. If there are multiple successful results, we prioritize the one with fewer bends or shorter overall routing lengths.

Early Termination: To enhance routing efficiency during DT routing, after predicting a new segment, we check if it allows early termination. For each predicted segment, we adjust its length to have the minimum distance between its end point and the end point of the net. If the routing is successful by using the adjusted length, we directly apply it to achieve the routing goal. Conversely, we continue the subsequent routing process with the original model-predicted length. As shown in Figure 9(a), the predicted segment in the model correctly routes toward the end point of the net, but the predicted length is insufficient. By modifying the length of the segment, the routing can be successful without continuing the subsequent routing process. Moreover, this method also reduces meaningless bends and the number of segments, as shown in Figure 9(b). Without using this technique, the model generates a trapezoidal routing at the end of the routing. With early termination, our approach extends the length of the fourth-to-last segment (colored in red) to reach the goal while reducing the trapezoidal segments (colored in blue) from the previous solution.

Minimum Adjustment: In this paper, if the distance between the end point of the last segment and the end point of the net is within a small distance, we consider it a successful connection. In practice, to physically connect the net to the pin, we can always achieve the goal by adjusting the length of the last segment and adding an additional segment if necessary.

Table 1: Benchmark Statistics

Case	# Nets	Chip Size (μm^2)	Substrate Size (μm^2)
Case1	308	11165 $*$ 3462	29600 $*$ 29600
Case2	115	12709 $*$ 11153	50268 $*$ 53002
Case3	296	22317 $*$ 27116	65180 $*$ 65380
Case4	195	9833 $*$ 8478	24610 $*$ 24610
Case5	92	10327 $*$ 8256	30795 $*$ 31200

4 EXPERIMENTAL RESULTS

In this section, we present the experimental results of the overall refinement results and the detour optimization on five distinct industrial cases with their specification reported in Table 1. We further conduct an ablation study of the RL approach to demonstrate the significant contribution given by the proposed DT model. All experiments are conducted using Python on an Intel Core i9-10940X 3.3GHz Linux workstation with NVIDIA GeForce RTX 3090 GPUs.

4.1 Overall Refinement Comparison

Table 2 presents the comparison of the refined results between the proposed approach and the baseline [8]. We also report the metrics of the original auto-routed designs, which are accomplished by Cadence Allegro, to highlight the importance of doing refinement. In order to ensure a fair comparison, we follow the previous work [8] to apply the *Area Optimization* and the *Redundant Bends Reduction* after our refinement flow. The effectiveness of the refinement is evaluated based on two metrics: the number of segments and the number of dense areas. To determine the number of dense areas, the original design is first divided into smaller regions. If the density of net area within a region exceeds 70%, it is classified as a dense area. A lower number of segments indicates fewer detours and redundant bends, and a lower number of dense areas indicates a more evenly distributed routing area. From the table, we can observe that the proposed refinement achieved an average detour reduction of 43% and improved area distribution by 33% compared to the auto-routed design, without any wirelength increment. Additionally, the time required for refinement significantly reduces from weeks to minutes. Comparing to the baseline [8], our method outperformed the previous approach in terms of segments, dense areas, and runtime. In conclusion, the comprehensive comparison of all metrics shows that the proposed method accomplishes the goal of routing refinement effectively.

4.2 Detour Optimization Comparison

Table 3 presents the comparison in the detour optimization stage, where **DNN**, **RS**, and **DT** refer to *DNN Guided Routing Method* proposed in [8], *Riverside Routing Method*, and *Decision Transformer*

Table 2: Refinement results for original auto-route, baseline [8], and the proposed method.

Case	Wirelength (μm)			# Segments			# Dense Areas			Runtime (sec.)	
	Auto-route	[8]	Ours	Auto-route	[8]	Ours	Auto-route	[8]	Ours	[8]	Ours
Case1	3600014	3598431	3590668	1993	1089	1150	1457	547	448	426	340
Case2	2450662	2430165	2422667	1443	672	570	1417	1239	1032	873	353
Case3	6141820	6108453	6063996	3646	1894	1676	2839	2582	2310	3594	1682
Case4	1962701	1949748	1943949	1819	1264	1111	948	563	487	962	665
Case5	910054	904697	901835	1029	981	824	547	531	529	475	378
Comparisons	1.00	0.99	0.99	1.00	0.64	0.57	1.00	0.74	0.67	1.00	0.63

Table 3: Performance Comparison in Detour Optimization

Case	# Segments		Runtime (sec.)	
	DNN	RS+DT	DNN	RS+DT
Case1	1531	1520	48	52
Case2	1157	1085	103	41
Case3	3416	3305	264	163
Case4	1576	1341	107	63
Case5	997	887	57	24
Comparisons	1.00	0.93	1.00	0.62

Routing Method, respectively. As observed in the table, the combination of **RS** and **DT** (**RS**+**DT**) outperforms **DNN** alone in solving detours, achieving a 7% reduction in the number of segments and a 38% decrease in runtime. This improvement is primarily due to the ability of **RS** to initially simplify significant detours into slight detours. Additionally, the flexible routing method of **DT** reduces the time required for **DNN** to establish auxiliary points when prediction errors occur.

4.3 Ablation Study of the RL Approach

To demonstrate the strength of DT, we perform an ablation study between the conventional RL, HyAR [10], and our proposed model in the complex RL routing environment. We select two architectures, HyAR-TD3 and HyAR-DDPG, which have optimal performance for fair comparison. Table 4 presents the comparison between two HyAR models and our DT model. In our experiments, we perform a local refinement test to evaluate their routability by using test cases where the target net contains five or fewer segments. From the table, two HyAR approaches obtain inferior averages, especially the HyAR-DDPG which is only about 25%. Although HyAR-TD3 achieves a comparable success rate in test cases with only one segment, its success rate degrades significantly to less than 30% as the number of segments increases. This is due to the training process of the conventional RL approach, which requires extensive iterations and optimization. However, in industrial situations, achieving successful attempts is challenging due to the highly complex routing environments. With our proposed DT model, leveraging efficient offline training, our average success rate exceeded 70%. Through this ablation study, the proposed method demonstrates its applicability and efficiency for use in industrial scenarios.

Table 4: Comparison of the success rate for the conventional RL [10] and the proposed DT method.

Model	Success Rate					
	# Segments					Average
	1	2	3	4	5	
HyAR-DDPG	38%	25%	23%	24%	17%	25.4%
HyAR-TD3	71%	64%	46%	31%	28%	48%
DT (Ours)	70%	89%	80%	70%	64%	74.6%

5 CONCLUSION

In conclusion, this paper proposed a hybrid detour refinement strategy that combines rule-based and DL-based approaches to achieve well-distributed and detour-reduced FCBGA substrate routing. We introduced *Riverside Routing Method* to enhance the effectiveness of modifications for significant detours. Moreover, we introduced *Decision Transformer Routing Method*, which leveraged the strengths of DT to achieve a more efficient solution for routing refinement. With the advantage of allowing offline training and long-tracking time steps, the proposed solution successfully improves the refinement performance with fewer execution runtimes. Experimental results demonstrate that our proposed method significantly reduced the number of detours and improved the area distribution on large-scale industrial gridless FCBGA substrate cases.

REFERENCES

[1] Lin et al., "A Complete PCB Routing Methodology with Concurrent Hierarchical Routing," ACM/IEEE DAC, 2021.
[2] Lin et al., "A Unified Printed Circuit Board Routing Algorithm With Complicated Constraints and Differential Pairs," ASP-DAC, 2021.
[3] Chi et al., "Practical Substrate Design Considering Symmetrical and Shielding Routes," DATE, 2022.
[4] He et al., "Circuit Routing Using Monte Carlo Tree Search and Deep Reinforcement Learning," VLSI-DAT, 2022.
[5] Yeh et al., "DPRoute: Deep Learning Framework for Package Routing," ASP-DAC, 2023.
[6] Liu et al., "Effective Congestion Reduction for IC Package Substrate Routing," ACM TODAES, 2010.
[7] Yeh et al., "Substrate Signal Routing Solution Exploration for High-Density Packages with Machine Learning," VLSI-DAT, 2022.
[8] Huang et al., "Deep Learning based Refinement for Package Substrate Routing," IEEE ECTC, 2023.
[9] Chen et al., "Decision transformer: Reinforcement learning via sequence modeling," NeurIPS, 2021.
[10] Li et al., "HyAR: Addressing Discrete-Continuous Action Reinforcement Learning via Hybrid Action Representation," ICLR, 2022.
[11] Lee et al., "Delay-Matching Routing for Advanced Packages," IEEE/ACM ICCAD, 2023.
[12] Vaswani et al., "Attention Is All You Need," NeurIPS, 2017.