Transformer-based Reinforcement Learning for Net Ordering in Detailed Routing

Zhanwen Zhou[1] , **Hankz Hankui Zhuo**[2,3*] , **Jinghua Zhou**[1] and **Wushao Wen**[1]
[1]School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China
[2]National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
[3]School of Artificial Intelligence, Nanjing University, Nanjing, China
zhouzhw26@mail2.sysu.edu.cn, hankz@nju.edu.cn, zhoujh76@mail2.sysu.edu.cn,
wenwsh@mail.sysu.edu.cn

Abstract

With feature size shrinking and design complexity increasing, detailed routing has become a crucial challenge in VLSI design. Although detailed routers have been proposed to judiciously handle hard-to-access pins and various design rules, their performances are sensitive to the order of nets to be routed, especially for those sequential routers with ripup-and-reroute scheme. In the published literature, net ordering strategies mainly rely on experts' knowledge to design heuristics to guarantee their performances. In this paper, we propose a novel transformer-based reinforcement learning framework for net ordering in detailed routing, aiming at automatically gaining failure/success routing experiences and building net order policies to guide detailed routing. Our experimental results show that our framework can effectively reduce the number of design rule violations and routing cost with comparable wirelength and via count, with comparison to state-of-the-art approaches.

1 Introduction

In the realm of very large-scale integration (VLSI) design, routing presents significant challenges and has emerged as a critical bottleneck in practical applications. This is mainly due to intricate design rules and extensive solutions involved [Liu *et al.*, 2019; Chen *et al.*, 2019]. VLSI routing is generally divided into two stages: global routing and detailed routing [Zhou and Zhuo, 2024]. Global routing involves dividing the entire routing space into smaller units called global cells (GCells) and routing the nets across these GCells. This process is often guided by congestion predictions to optimize routing paths [Liu *et al.*, 2013]. Detailed routing, on the other hand, focuses on creating rectilinear wiring interconnections that adhere to design rules [Park *et al.*, 2019]. The objective is to finalize the placement of segments and vias based on global routing solutions while minimizing design rule violations (DRV), wirelength, and the number of vias.

In detailed routing, a large routing area is often divided into nonoverlapping regions to facilitate parallel routing, each of which generally has a set of *nets* to be routed. Optimizing the order of nets to be routed is a challenging task due to the large scale of nets and complicated interconnections among nets [Kahng *et al.*, 2020b; Lin *et al.*, 2021]. In this paper we focus on tackling this challenging task for detailed routing.

There have been rules-based approaches proposed to optimize the order of nets [Jia *et al.*, 2017; Chen *et al.*, 2019; Kahng *et al.*, 2020b]. In those approaches, rules are manually designed for specific objectives with respect to the number of pins in a net [Kahng *et al.*, 2020b], the region size of the global guide for a net [Chen *et al.*, 2019], the bounding box of all pins in a net [Kahng *et al.*, 2020b], or the number of DRV [Jia *et al.*, 2017]. They tend to focus on specific characteristics of the routing process, which are then hardcoded into routing approaches, making them difficult to generalize to new design rules. To address the generalization issue, Lin et al. [Lin *et al.*, 2021] propose an asynchronous reinforcement learning framework to optimize the order of nets. The order of nets in their framework is determined by features, such as sizes of routing regions, numbers of conflicted nets, and previous routing results of each net. **They fail to consider the accessibility of pins, available spaces of routing regions, and overlapping relationships between two nets, which are crucial for optimizing the order of nets. It is challenging to design effective models to represent the above-mentioned features thoroughly and use the representations to build effective models to capture the order of nets.**

In this work, we propose a novel GCNs and **T**ransformer-based **RE**inforcement learning framework for **N**et ordering in **D**etailed routing, namely `TREND`[1]. Specifically, in order to represent features thoroughly, we build a graph convolutional network (GCN) to consider complicated influence relationships of nets. In order to optimize the sequential order of nets based on the output of GCN, we design a transformer model to effectively capture the order of nets. Finally we build a pointer network to generate indices of nets one by one via considering both influences of previously generated (indices of) nets and features of currently unordered nets. Note that, similar to [Lin *et al.*, 2021], our `TREND` approach targets at detailed routing of large-scale nets derived from real chips. We are aware of learning-based works [He *et al.*, 2022;

*Corresponding author.

[1]https://github.com/xrouting/trend

Chen *et al.*, 2023; Lin *et al.*, 2024] that were recently proposed for chip routing. They are either predominantly trained and evaluated in artificially generated small grid environments, or designed for global routing.

2 Background and Problem Formulation

Routing is on a stack of metal layers, each of which has a preferred direction for routing, either horizontal or vertical. The preferred routing directions of adjacent layers are orthogonal to each other to minimize signal crosstalk. A wire segment routes along the preferred direction on the regularly spaced tracks, which are pre-defined according to the minimum width and spacing constraint of wire. Wires on adjacent metal layers can be electrically connected by vias through the crosspoint of tracks. On each track, there are a series of crosspoints viewed as vertices. The vertices and connections among them on all metal layers compose a 3D grid graph for chip routing, as shown in Figure 1. A vertex is uniquely defined by a 3D index $\langle l, t, c\rangle$, which is a tuple of layer index, track index, and crosspoint index along the track. Adjacent vertices are connected by on-track wire segment edges on the same layer, or cross-layer via edges. Over a chip, there are some obstacles that vias and wire segments should avoid to prevent short and spacing violations. In Figure 1, $net1$ has three pins (A, B, C) to connect, and the orange path is one viable routing option. Note that there can be as many as millions of nets to be routed in a real chip.

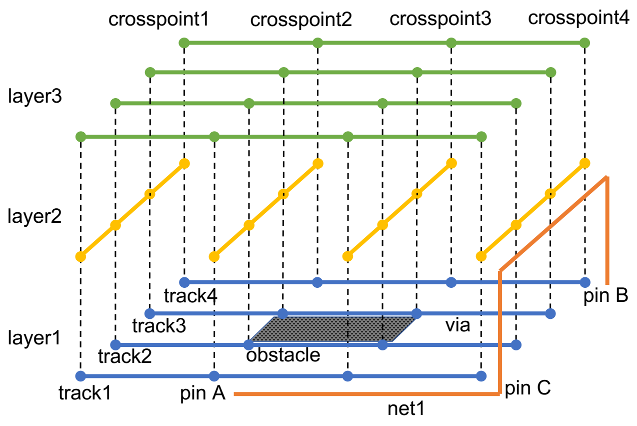


Figure 1: 3D grid graph for chip routing.

Given a placed netlist, design rules, and routing guides generated by global routing, detailed routing aims at successfully routing all nets and optimizing the weighted sum of *total wirelength*, *via count*, *non-preferred usage* (including wrong-way wires, out-of-guide and off-track wires/vias), and *DRV count* (including short, spacing, and minimum area violations) [Mantik *et al.*, 2018; Liu *et al.*, 2019]. Note that DRV are highly discouraged and suffer much more significant penalty than others.

The net ordering problem in detailed routing can be formally defined as follows. Given a set of m unrouted nets in a routing environment, numbered $1, 2, ..., m$, train a net ordering policy to generate a specific order of these net numbers, then the ordered nets will be routed sequentially. The objective is to minimize the overall wirelength, number of vias, and DRV of the routing solution. By optimizing the policy, the routing process becomes more efficient, cost-effective, and compliant with design constraints.

For example, Figure 2(a) shows a two-layer routing region with four nets. A common net ordering strategy may prioritize routing net_1 and net_2 first, using maze routing based on net size. However, this approach can result in paths that block connections for net_3 and net_4. A straightforward solution is to rip up net_1 and net_2, and reroute net_3 and net_4 first, as illustrated in Figure 2(b). This example underscores the importance of net ordering in the context of VLSI routing.

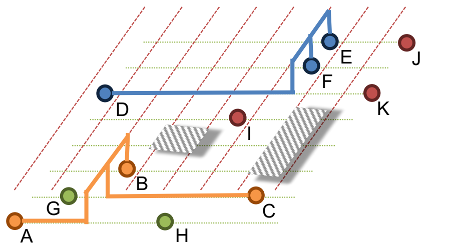
3 Our Routing Approach

3.1 Modeling

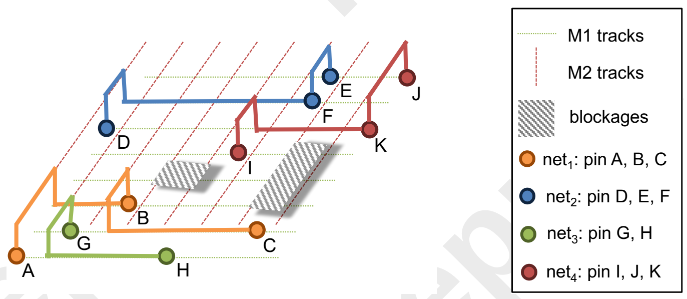
Net ordering in the context of electronic design automation can be effectively modeled using a Markov decision process (MDP), represented by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma\rangle$. By modeling net ordering as an MDP, it becomes possible to apply reinforcement learning techniques to find optimal routing strategies that efficiently connect all nets while minimizing costs and adhering to design constraints.

State Space ($\mathcal{S}$). For real chips with over thousands of nets, the routing space is extremely large. To reduce the problem size to a tractable level, a chip design is partitioned into nonoverlapping GCell-aligned regions, each of size $G \times G$ GCells, where G is a hyper-parameter. A GCell usually has 15 routing tracks on one metal layer. For each region, a nonregular-spaced 3D grid graph is built to support irregular tracks and off-track routing. The 3D grid graph is constructed by overlaying all preferred x- and y-direction grid lines (including on-track lines that align with the routing tracks, off-track lines that pass through any access point [Kahng *et al.*, 2020a] of a pin, and boundary lines) on each metal layer, and then repeating as many times as the number of metal layers along the z direction. Each state $s \in \mathcal{S}$ comprehensively represents the current routing scenario. It includes:

- A 3D grid graph, which models the physical layout of the routing area. The dimensions of the 3D grid graph are denoted as (D_x, D_y, D_z), representing the number of vertical lines, horizontal lines, and metal layers, respectively. The vertices of the 3D grid graph represent routing resource occupancy and congestion information in the routing environment. Each vertex is characterized by six properties: 1) grid coordinates range from (0, 0, 0) to $(D_x - 1, D_y - 1, D_z - 1)$; 2) physical location in the region; 3) vertex type, recording whether it is an obstacle, pin access point [Kahng *et al.*, 2020a], routable point, or nonexistent point; 4) pin number if it is an access point; 5) occupancy status; 6) occupying net identifier (if occupied).
- A netlist, which is a list of electrical connections that need to be established.
- All pins that are the connection points for the nets.
- The paths that have already been routed for the nets, providing a partial solution to the routing problem.



(a) Resource conflicts arise in the routing region.



(b) Rip up, reorder and reroute the conflicting nets.

Figure 2: Bad ordering of nets can lead to a large number of ripup-and-reroute iterations in detailed routing.

Table 1 summarizes the features of each net extracted from the state s. The first feature is the number of pins in the net. The second feature indicates the accessibility of the net, which is calculated by dividing the number of access points by the number of pins. The third feature denotes the occupancy of the routing area, which is the ratio of available vertices to total vertices in the bounding box. The fourth and fifth features are the net's 3D position and size relative to the region. The sixth feature is the count of other nets with bounding boxes overlapping the current net's bounding box.

Feature	Dim.	Description
Pins	1	Number of pins in the net.
Accessibility	1	Ratio of access point count to pin count.
Availability	1	Available / total vertices in the bounding box.
Position	3	Bottom left corner of the bounding box.
Size	3	Size ratio of the bounding box to the region.
Conflicts	1	Number of other nets that overlap with the net.

Table 1: Features of each net.

Action Space ($\mathcal{A}(s)$). This space represents all feasible routing actions at state s. These actions are state-dependent and involve ordering the nets for subsequent routing. All possible order of unrouted nets constitute the action space, which is not fixed since the number of unrouted nets for each region is not the same. To solve this problem, we build a pointer network to generate the appropriate number of nets.

Transition Model ($\mathcal{P}(s'|s,a)$). This is a probabilistic model that defines the likelihood of moving to a new state s' from the current state s after taking action a. It captures the dynamics of the routing process, including potential changes in the grid graph and netlist as routing progresses.

Reward Function ($\mathcal{R}$). In the net ordering problem, the reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ plays a crucial role. It is a bounded function that provides feedback from the routing environment after taking an action that involves routing a list of ordered nets within the grid. The reward is influenced by several factors:

- Increased Wirelength (wl): The additional wirelength incurred during routing. Minimizing wirelength is often desirable to reduce signal delay and resource usage.
- Vias ($vias$): The via count, representing vertical interlayer connections in the grid. Excessive vias increase manufacturing complexity and cost.
- Design Rule Violations ($drvs$): Violations of fabrication constraints that may cause functional failures or yield loss. The reward function penalizes DRV-inducing actions.

The routing cost is defined as the weighted sum of these metrics, as shown in Equation 1. We set the weights as $w_1 = 0.5$, $w_2 = 4$ and $w_3 = 500$ in our experiments, consistent with the ISPD-2019 detailed routing contest parameters [Liu *et al.*, 2019]. The reward R of an action a is defined as the cost difference between current state s and the new state s' after taking the action, as shown in Equation 2. α is set to the half-perimeter of the region for normalization across different regions. The initial routing cost is set to the cost of routing all nets in pre-defined order. The objective of the agent is to learn a policy to minimize the routing cost.

$$cost(s) = w_1 \cdot wl(s) + w_2 \cdot vias(s) + w_3 \cdot drvs(s) \quad (1)$$

$$R = \frac{cost(s) - cost(s')}{\alpha} \quad (2)$$

Discount Factor (γ). This parameter, usually between 0 and 1, determines the importance of future rewards. A higher value of γ places more emphasis on long-term benefits, encouraging strategies that optimize the routing solution over time.

A policy $\pi : \mathcal{S} \times \mathcal{A} \to [0,1]$ is a probabilistic mapping from states to actions. For a given state s, $\pi(a|s)$ represents the probability of selecting action a. The objective of detailed routing is to find a policy π that maximizes the expected sum of discounted rewards, denoted as $\mathcal{J}_\pi^{\mathcal{R}}$:

$$\mathcal{J}_\pi^{\mathcal{R}}(s) := \mathop{\mathbb{E}}_{a\sim\pi, s\sim\mathcal{P}} \left[\sum_t \gamma^t \mathcal{R}(s_t, a_t) \right] \quad (3)$$

This objective effectively translates to minimizing the overall wirelength, via count, and DRV count simultaneously in the routing solution.

3.2 Network Architecture

We design and implement a transformer-based reinforcement learning framework for net ordering in detailed routing, as shown in Figure 3.

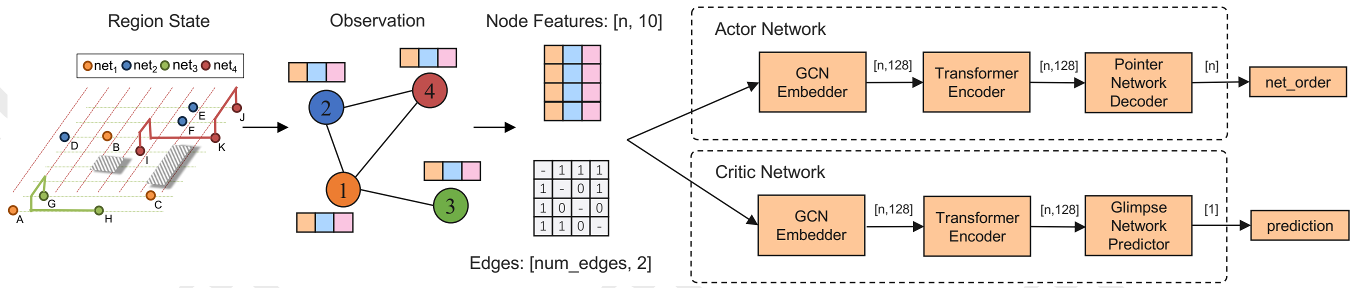


Figure 3: Network architecture of our routing approach.

Given a preordered set of n unrouted nets $N = \{net_i\}_{i=1}^{n}$ in a routing environment (ordered by baseline strategies), an undirected graph is constructed to represent the observation of the nets and their relationships within the routing region. We create one node for each net and draw an edge to connect two nets if their bounding boxes overlap. We use num_edge to represent the total number of edges in the graph. The features of these nets are defined as $X = [x_1, ..., x_n]^T$, where $X \in \mathbb{R}^{n\times 10}$, while the overlapped relationships between them are denoted as connections C, where $C \in \mathbb{Z}^{num\_edge\times 2}$. The model of our routing approach consists of an actor network and a critic network. Both networks process the state observation as input. The actor goes through three steps to generate the order of nets through a stochastic policy, while the critic establishes a performance baseline by estimating the expected routing cost of these nets based on the order recommended by the actor, providing policy gradient signals for continuous improvement. The goal of this architecture is to increase the probability of generating good orders for the nets in the routing environment.

Actor Network

The actor network is meticulously designed with three integral components: a graph convolutional network (GCN) embedder, a transformer encoder, and a pointer network decoder. Each of them plays a crucial role in generating the appropriate order of the nets.

Each routing region contains a distinct number of nets to be routed, and the overlapping relationships among these nets significantly impact the routability of the congested areas. To enhance the routing environment representation, GCN is used to map the net features and their overlapping relationships to a state embedding $M = GCN(X, C)$ where $M = [m_1, ..., m_n]^T \in \mathbb{R}^{n\times d}$ denotes the learned feature matrix with d-dimensional embeddings for n nets. d is set to 128 in our implementation. We use 2 convolutional layers with rectified linear unit (ReLU) activation to generate net embeddings. The first layer expands the feature dimension from 10 to 64, while the second layer expands it to 128 dimensions. The final expanded representation encodes both local neighboring influences and global environmental information.

To effectively capture the order of nets, the embedding M is then processed by a transformer encoder [Vaswani *et al.*, 2017] with three identical encoding blocks, each of which includes a multi-head self-attention layer and a full connected feedforward layer. We employ a residual connection around each of the two sub-layers, followed by layer normalization to stabilize the training. The final output of the transformer encoder is $E = [e_1, ..., e_n]^T \in \mathbb{R}^{n\times d}$, which contains all encodings for n nets. The dimension of the encoding for each net remains the same as d during the encoding process. Post-encoding, the sequential context of neighboring nets is aggregated to generate enhanced per-net representations.

Finally, we implement a pointer network decoder [Vinyals *et al.*, 2015] to process the encodings E and sequentially generate an ordered series of net numbers net_order. The decoder decides the subsequent net to route recurrently, that is, the previously generated series of net numbers serve as an additional input for computing the next net number. More specifically, we use a query vector of size 360 to encode the state that contains the previously generated series of net numbers. The pointer network decoder takes the encodings $E = [e_1, ..., e_n]^T$ and the query vector as inputs, and outputs the probability $P = [p_1, ..., p_n]^T$ of selecting each net to be appended to the series. It first uses vector zero as the query vector and selects a number of an unrouted net as the starting number u_1 and marks it as routed. In the following $n - 1$ time steps, the decoder computes the probabilities of selecting each net based on the encodings of these nets and a new query vector which encodes the new state. The probabilities of already-routed nets are set to 0 by a binary mask to avoid being reselected. A new net number u_t for $t = 2, ..., n$ is generated at each step, until all unrouted nets are ordered.

Critic Network

The critic network estimates the advantage value function by predicting the expected routing cost reduction achieved by the actor's net ordering policy relative to TritonRoute's baseline strategy. This advantage signal tells us how much the actor network performs better than expectation and guides the actor's policy updates through gradient ascent. Let $L(X, C, net\_order)$ denote the routing cost reduction achieved by the actor's policy, and $b(X, C)$ represent the critic's predicted baseline. The advantage value calculated by $b(X, C) - L(X, C, net\_order)$ will quantify the relative performance of the actor network for the generated order of nets net_order.

The critic network is also composed of three components: a GCN embedder, a transformer encoder, and a glimpse network predictor. The GCN embedder and transformer encoder in the critic network share identical architectures with the actor network, but employ independent parameters. They encode the features of each net and their overlapping relationships into the critic encodings $E' = [e'_1, ..., e'_n]^T \in \mathbb{R}^{n\times d}$. Then the glimpse network [Mnih *et al.*, 2014] takes the encodings E' as input, computes a weighted sum of the encodings, and outputs the predicted baseline scalar $b(X, C)$.

Parameters Update

This section presents the parameter update mechanisms for both the actor network (θ) and the critic network (ψ). For a set of n unrouted nets N with net features X and connections C in a routing environment, the expected advantage in routing cost reduction achieved by the actor-generated net ordering is defined as $J(\theta|X, C)$ in Equation (4):

$$J(\theta|X, C) = \sum_{o\in O}(b(X, C) - L(X, C, o))p_\theta(o|X, C) \quad (4)$$

O is the set of all possible orders of N. We utilize the REINFORCE algorithm [Williams, 1992] to compute the gradient of Equation (4) and update θ using stochastic gradient ascent.

Unlike the training method used in the actor network, we train the critic network to predict the routing cost reduction for the ordered nets net_order generated by the actor network, and minimize the squared error as shown in Equation (5):

$$Loss(\psi) = ||b_\psi(X, C) - L(X, C, net\_order)||_2^2 \quad (5)$$

We update ψ using stochastic gradient descent.

4 Experiment and Results

4.1 Routing Environment and Benchmarks

We design a comprehensive and adaptable routing environment, namely XRoute-Env, to facilitate the work of reinforcement learning for researchers in the field of detailed routing. It offers a suite of built-in features that streamline the process of designing and evaluating routing strategies. The environment is built upon a highly customized version of TritonRoute [Kahng *et al.*, 2022]. It is designed to simulate the entire routing process of a chip, starting with the input of design files and a routing guide. The routing engine iteratively applies ripup-and-reroute optimization, progressively refining the solution until converging to a DRC-clean detailed routing result.

To facilitate training and comparisons of different approaches in XRoute-Env, we experiment on the benchmarks from ISPD-2018 [Mantik *et al.*, 2018] and ISPD-2019 [Liu *et al.*, 2019] detailed routing contests. These two benchmark suites are established standards in detailed routing research, extensively adopted in the literature [Li *et al.*, 2019; Kahng *et al.*, 2020a; Lin *et al.*, 2021; Kahng *et al.*, 2022] for performance benchmarking. The ISPD-2019 benchmark suite enhances routing realism over ISPD-2018 by incorporating advanced design rules, resulting in test cases that better emulate commercial chip routing challenges. The characteristics of these benchmarks are summarized in Table 2. There are totally 20 test cases in 65nm, 45nm, and 32nm technology nodes derived from real chips. The largest one has up to 899404 standard cells and 895253 nets, which makes it very challenging to finish routing efficiently because of the extremely large routing space.

BM	#std	#blk	#net	#layer	Tech.
18t1	8879	0	3153	9	45nm
18t2	35913	0	36834	9	45nm
18t3	35973	4	36700	9	45nm
18t4	72094	0	72401	9	32nm
18t5	71954	0	72394	9	32nm
18t6	107919	0	107701	9	32nm
18t7	179865	16	179863	9	32nm
18t8	191987	16	179863	9	32nm
18t9	192911	0	178857	9	32nm
18t10	290386	0	182000	9	32nm
19t1	8879	0	3153	9	32nm
19t2	72094	4	72410	9	32nm
19t3	8283	4	8953	9	32nm
19t4	146442	7	151612	5	65nm
19t5	28920	6	29416	5	65nm
19t6	179881	16	179863	9	32nm
19t7	359746	16	358720	9	32nm
19t8	539611	16	537577	9	32nm
19t9	899341	16	895253	9	32nm
19t10	899404	16	895253	9	32nm

Table 2: Characteristics of the ISPD-2018 & ISPD-2019 detailed routing benchmarks (BM).

We follow the ISPD-2019 detailed routing contest and a lot of literature on detailed routing [Li *et al.*, 2019; Lin *et al.*, 2021; Kahng *et al.*, 2022] to evaluate the routing results from different routers by DRV count, total wirelength, via count and elapsed time.

4.2 Experiment Setup

We partition each benchmark into regions of size 7×7 GCells as TritonRoute [Kahng *et al.*, 2021]. We train our model first in every routing region from 18t1 for 30,000 steps and then in every routing region from 19t3 for 20,000 steps using the learned parameters. This speeds up the training process, since the experience learned from small regions can be used to route large regions. The learning rates are set to 0.00005 and 0.00002, respectively. At inference time, we greedily pick the net number with maximum probability at each step to construct the final routing order. To accelerate the routing process, we bypass regions with few overlapping nets. We calculate the average overlapping rate of a region by summing all conflicts among nets, then dividing it by the number of nets, and dividing the result by the number of nets again to normalize the value. If the average overlapping rate is smaller than a threshold (we use 0.3 in our experiments), the model will just return the original order of nets provided by XRoute-Env with the default strategy based on the number of pins and the size of each net.

We compare our approach with another RL-based net ordering approach [Lin *et al.*, 2021], namely "Lin's approach" in this paper, and the known-best academic detailed router TritonRoute [Kahng *et al.*, 2021]. Net ordering prioritization in TritonRoute is determined by pins count, area of the bounding box and net identity.

We used the same routing settings in XRoute-Env for these approaches. All routings run for 64 iterations at most, with region shift for one GCell in alternate iterations. All experiments run on a Linux server with a 40-core Intel Xeon CPU E5-2650 at 2.3 GHz and 240 GB shared memory, equipped with 2 NVIDIA GeForce Titan XP GPU with totally 24 GB video memory.

4.3 Results

Comparison on Different Net Ordering Strategies

In this section, we compare the quality of the routing solution between our routing approach, Lin's approach, and TritonRoute on the 20 benchmarks from ISPD-2018 and ISPD-2019 detailed routing contests, to evaluate the routing performance of these net ordering approaches on different chips with varying sizes and complexities. In addition to elapsed time, DRV count, wirelength and via count, we calculate the routing cost, which is defined in Equation 1, to assess the overall performance of the routing solution.

Table 3 and Table 4 demonstrate the routing results of the 20 benchmarks between our transformer-based RL approach, Lin's approach, and TritonRoute. The '-' symbol in the table indicates routing failure due to unresolved connectivity errors after the first iteration, caused by multi-pin nets with unconnected pins.

BM	Elapsed time (hh:mm:ss)		
	Our	Lin	TR
18t1	00:01:49	00:03:57	00:00:25
18t2	00:12:52	00:27:38	00:02:22
18t3	00:26:48	01:06:15	00:13:11
18t4	00:32:26	01:09:27	00:13:35
18t5	00:21:49	00:50:54	00:04:05
18t6	00:40:13	01:22:43	00:09:33
18t7	01:40:33	-	00:18:00
18t8	01:44:43	-	00:19:35
18t9	01:14:51	02:27:45	00:15:34
18t10	02:25:42	-	01:07:06
19t1	00:04:26	00:07:45	00:01:12
19t2	01:02:14	-	00:15:25
19t3	00:09:24	00:14:18	00:04:03
19t4	01:16:49	-	00:53:28
19t5	00:03:54	00:07:09	00:01:13
19t6	02:34:19	-	00:28:32
19t7	06:08:27	09:22:46	00:39:19
19t8	12:09:42	14:13:15	00:51:14
19t9	13:39:43	22:31:57	01:27:14
19t10	16:15:52	-	02:27:33

Table 3: Comparison of elapsed time between our routing approach (Our), Lin's approach (Lin) and TritonRoute (TR) with queue-based ripup-and-reroute on the ISPD-2018 & ISPD-2019 detailed routing benchmarks (BM).

Table 3 shows that our approach and Lin's approach cost more time than TritonRoute to route each benchmark, although the difference in elapsed time between these approaches is not large. This increased runtime is primarily due to the time-intensive processes of feature encoding and net order generation. These processes could potentially be optimized using off-the-shelf representation learning techniques, such as model compression and acceleration [Cheng *et al.*, 2017; Xu and McAuley, 2023].

However, the quality of the routing results is more important than the time taken to complete the routing, especially for a large chip. The routing results in Table 4 show that all three approaches successfully route every benchmark with zero design rule violation except 19t10, which is the biggest benchmark. According to the routing cost, our approach performs better than Lin's approach in almost all benchmarks except 18t5, and better than TritonRoute for large test cases, including 18t4 to 18t10, 19t4, 19t6, 19t8 to 19t10. It suggests that our framework has learned the experience to route nets with fewer wirelength and via count in complex routing environment with lots of overlapping nets. Reduced wirelength and via count directly correlate with lower manufacturing costs, making our net ordering approach effective for simultaneous performance enhancement and cost reduction in post-routing chips.

Routing Without Queue-Based Ripup-and-Reroute

Queue-based ripup-and-reroute scheme is a novel contribution in TritonRoute-WXL [Kahng *et al.*, 2022], which decreases the design rule violations and runtime by switching the routing order of conflicting nets during one iteration. In order to demonstrate the performance of net ordering, we disable the queue-based ripup-and-reroute scheme so that all unrouted nets or nets with any violations in the previous iteration will be ripped up and rerouted once and only once at next iteration. In this section, we compare our routing approach with Lin's approach and TritonRoute to route the 10 benchmarks from ISPD-2018 and first 6 benchmarks from ISPD-2019 detailed routing contest without using queue-based ripup-and-reroute flow. All routings run for 64 iterations at most. The benchmarks from 19t7 to 19t10 are too large for these approaches to finish routing in 100 hours for 64 iterations.

Table 5 demonstrates the routing results. The results show that both two RL-based approaches produce less DRV than TritonRoute in all experimental benchmarks except 19t2. It suggests that these RL-based approaches have learned the routing experience to avoid violations by ordering nets, while TrionRoute relies on the queue-based ripup-and-reroute scheme to reroute a net with violation over and over again.

We can also observe that our approach performs best in almost all benchmarks based on the routing cost metric, except 18t2, 19t2 and 19t3. The routing cost of our approach is better than that of TritonRoute in 19t3 and very close to that of TritonRoute in 18t2. It suggests that RL-based algorithms can be used for detailed routing in large-scale chips with fairly good routing results. Although this experience is learned from the regions in 18t1 and 19t3, it can be generalized to other benchmarks with much larger routing space.

5 Conclusion and Future Work

This paper presents a transformer-based reinforcement learning framework for net ordering in detailed routing, which

BM	DRV count			Wirelength (DBU)			Via count			Routing cost		
	Our	Lin	TR	Our	Lin	TR	Our	Lin	TR	Our	Lin	TR
18t1	0	0	0	**86747**	87111	86753	35573	35720	**35513**	185665.5	186435.5	**185428.5**
18t2	0	0	0	1576242	1578576	**1576206**	363658	365681	**363627**	2242753.0	2252012.0	**2242611.0**
18t3	0	0	0	1756345	1759396	**1756206**	364486	366958	**364460**	2336116.5	2347530.0	**2335943.0**
18t4	0	0	0	**2624698**	2628327	2624715	**730583**	736029	730859	**4234681.0**	4258279.5	4235793.5
18t5	0	0	0	2767567	2770337	**2767487**	907371	**906569**	907409	5013267.5	**5011444.5**	5013379.5
18t6	0	0	0	**3568829**	3576404	3568835	**1388624**	1389074	1388924	**7338910.5**	7344498.0	7340113.5
18t7	0	-	0	**6504482**	-	6504609	2270710	-	**2270704**	**12335081.0**	-	12335120.5
18t8	0	-	0	6537218	-	**6537202**	**2288881**	-	2289026	**12424133.0**	-	12424705.0
18t9	0	0	0	**5460824**	5474072	5460970	**2272300**	2275228	2272577	**11819612.0**	11837948.0	11820793.0
18t10	0	-	0	**6797185**	-	6797456	**2475983**	-	2476292	**13302524.5**	-	13303896.0
19t1	0	0	0	**63420**	63546	63432	**37136**	37340	37177	**180254.0**	181133.0	180424.0
19t2	0	-	0	2485283	-	**2485251**	805972	-	**805715**	4466529.5	-	**4465485.5**
19t3	0	0	0	82941	83177	**82874**	63775	64438	**63584**	296570.5	299340.5	**295773.0**
19t4	0	-	0	**6024472**	-	6024711	**1086714**	-	1087234	**7359092.0**	-	7361291.5
19t5	0	0	0	**951129**	953089	951172	169243	170276	**169178**	1152536.5	1157648.5	**1152298.0**
19t6	0	-	0	**6582348**	-	6582467	**1991724**	-	1991825	**11258070.0**	-	11258533.5
19t7	0	0	0	**12178450**	12204275	12178615	4511050	4541503	**4510697**	24133425.0	24268149.5	**24132095.5**
19t8	0	0	0	18726776	18757337	**18726739**	**6983678**	6993049	6984292	**37298100.0**	37350864.5	37300537.5
19t9	0	0	0	**28334694**	28383299	28335064	11584500	11598176	**11584493**	**60505347.0**	60584353.5	60505504.0
19t10	20	-	20	**28011370**	-	28011623	**11721823**	-	11722960	**60902977.0**	-	60907651.5

Table 4: Comparison of DRV count, wirelength, via count and routing cost between our routing approach (Our), Lin's approach (Lin) and TritonRoute (TR) with queue-based ripup-and-reroute on the ISPD-2018 & ISPD-2019 detailed routing benchmarks (BM).

BM	DRV count			Wirelength (DBU)			Via count			Routing cost		
	Our	Lin	TR	Our	Lin	TR	Our	Lin	TR	Our	Lin	TR
18t1	**0**	**0**	**0**	**86553**	86708	86572	**35576**	35685	35601	**185580.5**	186094.0	185690.0
18t2	**0**	**0**	**0**	**1574491**	1576548	1574538	364130	366391	**364124**	2243765.5	2253838.0	**2243765.0**
18t3	14	**3**	22	1754029	1757038	**1753984**	364701	368026	**364609**	**2342818.5**	2352123.0	2346428.0
18t4	61	**31**	89	2621258	2624536	**2621145**	733092	740539	**732761**	**4273497.0**	4289924.0	4286116.5
18t5	8	**5**	10	2764065	2766600	**2763972**	**892069**	893066	892133	**4954308.5**	4958064.0	4955518.0
18t6	53	**32**	57	3559460	3566712	**3559428**	1355160	1356002	**1355045**	**7226870.0**	7223364.0	7228394.0
18t7	**247**	-	306	6487839	-	**6487468**	2220154	-	**2219026**	**12248035.5**	-	12272838.0
18t8	**177**	-	195	6520057	-	**6519990**	2237048	-	**2236386**	**12296720.5**	-	12303039.0
18t9	175	**170**	279	**5445442**	5457025	5445468	**2221327**	2223374	2221375	**11695529.0**	11707008.5	11747734.0
18t10	**219**	-	246	6778356	-	**6778309**	2417597	-	**2417120**	**13169066.0**	-	13180634.5
19t1	8	**4**	14	63172	63220	**63148**	**36996**	37185	37103	**183570.0**	182350.0	186986.0
19t2	470	-	**173**	2478050	-	**2477779**	803797	-	**802809**	4689213.5	-	**4536625.5**
19t3	61	**54**	66	82312	82534	**82269**	63606	**63395**	63401	326080.0	**321847.0**	327738.5
19t4	**13**	-	19	6012042	-	**6012026**	1084181	-	**1084118**	**7349245.0**	-	7351985.0
19t5	**4**	**4**	5	**949142**	950705	949156	168254	170471	**168192**	**1149587.0**	1159236.5	1149846.0
19t6	**446**	-	615	6564219	-	**6563725**	1976476	-	**1973987**	**11411013.5**	-	11485310.5

Table 5: Comparison of DRV count, wirelength, via count and routing cost between our routing approach (Our), Lin's approach (Lin) and TritonRoute (TR) without queue-based ripup-and-reroute on the ISPD-2018 & ISPD-2019 detailed routing benchmarks (BM).

demonstrates superior routing quality over conventional algorithms. Nevertheless, our framework exhibits several limitations that highlight promising directions for future research:

- Experimental results indicate that while our transformer-based framework achieves significant improvements in most benchmarks, it does not consistently surpass TritonRoute's heuristic-based approach across all test cases. We will focus on two research directions: (1) refining exploration precision via adaptive net selection constraints during routing, and (2) improving solution quality and routing convergence through scaled-up training with extended episode counts, prolonged duration and extended region coverage.

- The inference module (learned offline) in `TREND` is time-consuming for large benchmarks, which could be optimized with novel representation learning approaches in the future, such as action model learning [Zhuo *et al.*, 2011; Jin *et al.*, 2022] in planning community for better capturing the logical relations among routing actions.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 62076263).

References

[Chen *et al.*, 2019] Gengjie Chen, Chak-Wa Pui, Haocheng Li, and Evangeline F. Y. Young. Dr. cu: Detailed routing by sparse grid graph and minimum-area-captured path search. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(9):1902–1915, 2019.

[Chen *et al.*, 2023] Hao Chen, Kai-Chieh Hsu, Walker J. Turner, Po-Hsuan Wei, Keren Zhu, David Z. Pan, and Haoxing Ren. Reinforcement learning guided detailed routing for custom circuits. In David G. Chinnery and Iris Hui-Ru Jiang, editors, *Proceedings of the 2023 International Symposium on Physical Design, ISPD 2023, Virtual Event, USA, March 26-29, 2023*, pages 26–34. ACM, 2023.

[Cheng *et al.*, 2017] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *CoRR*, abs/1710.09282, 2017.

[He *et al.*, 2022] Youbiao He, Hebi Li, Tian Jin, and Forrest Sheng Bao. Circuit routing using monte carlo tree search and deep reinforcement learning. In *2022 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, pages 1–5, 2022.

[Jia *et al.*, 2017] Xiaotao Jia, Yici Cai, Qiang Zhou, and Bei Yu. A multicommodity flow-based detailed router with efficient acceleration techniques. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(1):217–230, 2017.

[Jin *et al.*, 2022] Mu Jin, Zhihao Ma, Kebing Jin, Hankz Hankui Zhuo, Chen Chen, and Chao Yu. Creativity of AI: automatic symbolic option discovery for facilitating deep reinforcement learning. In *AAAI*, pages 7042–7050. AAAI Press, 2022.

[Kahng *et al.*, 2020a] Andrew B. Kahng, Lutong Wang, and B. Xu. The tao of PAO: Anatomy of a pin access oracle for detailed routing. In *2020 57th ACM/IEEE design automation conference (DAC)*, pages 1–6. IEEE, 2020.

[Kahng *et al.*, 2020b] Andrew B. Kahng, Lutong Wang, and Bangqi Xu. The tao of PAO: anatomy of a pin access oracle for detailed routing. In *57th ACM/IEEE Design Automation Conference, DAC 2020, San Francisco, CA, USA, July 20-24, 2020*, pages 1–6. IEEE, 2020.

[Kahng *et al.*, 2021] Andrew B. Kahng, Lutong Wang, and Bangqi Xu. Tritonroute: The open-source detailed router. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 40(3):547–559, 2021.

[Kahng *et al.*, 2022] Andrew B. Kahng, Lutong Wang, and Bangqi Xu. Tritonroute-WXL: The Open-Source Router With Integrated DRC Engine. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(4):1076–1089, 4 2022.

[Li *et al.*, 2019] Haocheng Li, Gengjie Chen, Bentian Jiang, Jingsong Chen, and Evangeline FY Young. Dr. cu 2.0: A scalable detailed routing framework with correct-by-construction design rule satisfaction. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–7, Westminster, CO, USA, 2019. IEEE.

[Lin *et al.*, 2021] Yibo Lin, Tong Qu, Zongqing Lu, Yajuan Su, and Yayi Wei. Asynchronous reinforcement learning framework and knowledge transfer for net-order exploration in detailed routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(9):3132–3142, 2021.

[Lin *et al.*, 2024] Zhenkun Lin, Genggeng Liu, Xing Huang, Yibo Lin, Jixin Zhang, Wen-Hao Liu, and Ting-Chi Wang. A unified deep reinforcement learning approach for constructing rectilinear and octilinear steiner minimum tree. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1–1, 2024.

[Liu *et al.*, 2013] Wen-Hao Liu, Wei-Chun Kao, Yih-Lang Li, and Kai-Yuan Chao. NCTU-GR 2.0: Multithreaded collision-aware global routing with bounded-length maze routing. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 32(5):709–722, 2013.

[Liu *et al.*, 2019] Wen-Hao Liu, Stefanus Mantik, Wing-Kai Chow, Yixiao Ding, Amin Farshidi, and Gracieli Posser. Ispd 2019 initial detailed routing contest and benchmark with advanced routing rules. In *Proceedings of the 2019 International Symposium on Physical Design*, ISPD '19, page 147151, New York, NY, USA, 2019. Association for Computing Machinery.

[Mantik *et al.*, 2018] Stefanus Mantik, Gracieli Posser, Wing-Kai Chow, Yixiao Ding, and Wen-Hao Liu. Ispd 2018 initial detailed routing contest and benchmarks. In *Proceedings of the 2018 International Symposium on Physical Design*, ISPD '18, page 140143, New York, NY, USA, 2018. Association for Computing Machinery.

[Mnih *et al.*, 2014] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. *Advances in neural information processing systems*, 27, 2014.

[Park *et al.*, 2019] Dongwon Park, Ilgweon Kang, Yeseong Kim, Sicun Gao, Bill Lin, and Chung-Kuan Cheng. ROAD: routability analysis and diagnosis framework based on SAT techniques. In Ismail Bustany and William Swartz, editors, *Proceedings of the 2019 International Symposium on Physical Design, ISPD 2019, San Francisco, CA, USA, April 14-17, 2019*, pages 65–72. ACM, 2019.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 60006010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[Vinyals *et al.*, 2015] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, page 26922700, Cambridge, MA, USA, 2015. MIT Press.

[Williams, 1992] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(34):229256, May 1992.

[Xu and McAuley, 2023] Canwen Xu and Julian J. McAuley. A survey on model compression and acceleration for pretrained language models. In *Thirty-Seventh AAAI Conference on Artificial Intelligence*, pages 10566–10575, 2023.

[Zhou and Zhuo, 2024] Zhanwen Zhou and Hankz Hankui Zhuo. Survey on intelligent routing approaches for chips. *Acta Automatica Sinica*, 50(9):1671–1703, 2024.

[Zhuo *et al.*, 2011] Hankz Hankui Zhuo, Qiang Yang, Rong Pan, and Lei Li. Cross-domain action-model acquisition for planning via web search. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 21, pages 298–305, 2011.