GNN as Policy Network for Deep Reinforcement Learning Generalization

Roozmehr Jalilian, Shidi Xi
Department of Electrical & Computer Engineering
The University of British Columbia
{roozmehr.jalilian,xsd99}@ece.ubc.ca
Vancouver, BC V6T 1Z4

Abstract

Routing in integrated circuits (IC) design is often modeled as a pathfinding problem and has consistently represented one of the most challenging aspects due to its complexity and time-consuming nature. Even with state-of-the-art commercial CAD tools, routing complex designs can take hours, if not days. Consequently, there is a growing interest in applying machine learning-based approaches to overcome these challenges. In previous work, we introduced a novel reinforcement learning (RL) approach for routing. While this RL router outperformed the baseline, it was limited by poor generalizability, requiring a new policy to be trained from scratch for each problem. In this project, we aim to address this limitation by introducing a novel graph neural network (GNN) as the policy network for the RL router. This enhancement allows the agents to generalize their learned routing policies to new, unseen designs, reducing the computation time significantly.

1 Introduction

IC routing involves connecting distinct electrical contact points (pins) on a chip using metallic wires. This process is crucial for the functionality of computer chips, as it enables data transfer and computing activities between components like the CPU and memory. We model routing as a pathfinding problem, representing the chip as a 2-dimensional grid. Pins, identified by their (x, y) coordinates, are placed on the grid's nodes. The objective is to establish paths connecting these nodes while navigating the grid edges. Each edge possesses a capacity feature, indicating the maximum number of permissible wires due to the physical constraints of wire thickness and finite space within a chip region. An optimal routing solution seeks to minimize total wire length and avoid edge overflow (usage of an edge exceeding its capacity), ensuring all connections are efficient and within capacity.

Two important terminologies here are net and netlist. A net refers to a set of pins requiring connection, and a netlist is a set comprises multiple nets needing routing during the process. And for clarity, we will use the terms nodes and pins interchangeably throughout this proposal.

In prior research, we introduced a multi-agent RL router to address the IC routing challenge. Each agent, tasked with routing one individual net, operated concurrently under a shared super policy, as we designed the agents to be homogeneous and fully-cooperative. Trained with proximal policy optimization (PPO) [1] and evaluated against benchmarks, this RL router demonstrated superior performance compared to the A* baseline. However, a limitation emerged: the policy, though effective for specific benchmarks, lacked generalizability for other, albeit similar, problems. We attributed this shortfall to the policy neural network's architecture, comprising several fully connected (FC) layers, which struggled to extract rich information from the state encodings.

Contrastingly, graph neural networks (GNNs) exhibit robust generalization capabilities when processing graphical data [2, 3], aligning with the routing problem's grid graph model. In this project, we introduce a novel message-passing GNN architecture integrated into the RL router's policy network. The GNN accepts the routing grid graph from the environment as state input and determines actions accordingly. This innovation aims to bolster the RL router's ability to generalize trained policy to problems not encountered during training, promising substantial reductions in computation time.

2 Related work

Numerous studies in existing literature have verified the effectiveness of employing Graph Neural Networks (GNNs) within the policy/value network of Reinforcement Learning (RL) models to enhance generalization, especially when the agent's environment is graph-representable. However, to our knowledge, no existing study has directly explored the use of GNN equipped RL for IC routing.

[4] introduced an RL IC router trained via a Deep Q-Network (DQN), utilizing a multi-layer perceptron (MLP) as the RL agent's value network. However, similar to our previous work, it showed a lack of generalization. Contrastingly, [2] incorporated a GNN as the value network of an RL agent, also trained by DQN. This approach manifested notable generalization capabilities in network routing—a field bearing similarities to IC routing—and serves as the principal inspiration for our current proposal. In parallel, both [5] and [3] echoed the integration of GNNs with RL models wherein the operational problems are graphs. Furthermore, [6] and [7] adopted GNNs as encoder layers for their respective RL policy and value networks.

3 Methodology

In this proposal, we limit our discussion to a simplified case of IC routing to facilitate the rapid development of a prototype. As the project advances, we will introduce additional complexities to refine and compact the model.

3.1 Modeling IC routing as a grid graph

Figure 1 illustrates a simplified IC routing problem, represented as a grid graph with a single net. The grid, sized 4x4 and with an edge capacity of 1, contains a net composed of two pins positioned at coordinates (0,0) and (3,3). The grid comprises 16 nodes in total. Assuming the RL agent initiates its path at node (0,0), its destination is fixed at (3,3). The grid is defined by a node feature matrix $\mathbf{X}$, an edge feature matrix $\mathbf{E}$, and the adjacency matrix $\mathbf{A}$.

Matrix $\mathbf{X}$, with dimensions of 16x2, consists of rows that represent the binary feature vectors of nodes. The first element of each row signifies the presence of the RL agent, while the second element indicates whether a target resides at that specific node. Both $\mathbf{E}$ and $\mathbf{A}$ are 16x16 matrices. In particular, $\mathbf{E}$ encapsulates the scalar capacities of the edges connecting the nodes.

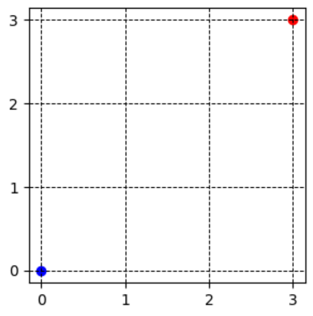

Figure 1: The grid graph of a simple IC routing problem.

3.2 GNN architectural design

The GNN employs a message-passing mechanism, processing inputs of matrices $\mathbf{X}$, $\mathbf{E}$, and $\mathbf{A}$ from the RL environment and yielding the probability distribution of all potential actions. An MLP implements the message network, with messages being aggregated through element-wise summation. Node states are updated using a Gated Recurrent Unit (GRU).

Following the message-passing phase, the updated node states are fed into another MLP, culminating in a softmax layer that generates the action probability distribution.

The GNN architecture cannot reply on the graph topology, e.g., the number of nodes. We want the GNN to generalize to graphs of any size.

3.3 RL policy training

We plan to integrate the proposed GNN into our RL router, serving as its policy network. The training will be conducted using PPO with RLlib. The GNN's parameters will be refined through stochastic gradient descent, guided by feedback from the value network.

3.4 Project Timeline

- Conduct an extended literature review to gather comprehensive insights and foundational knowledge (5 days).
- Develop the GNN prototype, integrate it with the RL router, and initiate training on a preliminary benchmark (14 days).
- Evaluate the enhanced RL router's performance on a variety of benchmarks, including those analogous and distinct from the training set (1 day).
- Analyze the outcomes to assess the model's generalization capabilities and performance metrics, making figures and tables (3 days).
- (Bonus) Expand the model and its design to accommodate multi-agent concurrent routing, enhancing scalability and efficiency (7 days).
- Writing up the report (14 days).

4 Experiments

We will construct a baseline using an MLP policy network and train it on a simple benchmark to facilitate a comparative analysis.

4.1 Comparing Training Efficiency

This experiment aims to compare the training efficiency of the GNN-equipped router with the baseline router, both trained on the same benchmark. We will assess their performance by examining the plots of reward versus training iterations. The objective is to evaluate the training efficiency of the GNN-equipped router in comparison to the baseline. As suggested by [3], we anticipate that the new design will demonstrate comparable, if not superior, training efficiency.

4.2 Comparing Zero-Shot Performance

We will employ the zero-shot policies derived from the previous experiment to perform inference on several unseen benchmarks. These benchmarks will feature netlists with canvas sizes, numbers of nets, and total numbers of pins per net that resemble the training benchmark. The comparison criteria will include rewards and routing solution quality (wire length, overflow). We predict that the new GNN-equipped design will surpass the baseline in performance.

4.3 Comparing Fine-Tuning Performance

This experiment is designed to evaluate the scalability of the GNN to more complex problems. We will initiate zero-shot inference on benchmarks characterized by different numbers of nets, chip canvas sizes, and numbers of pins per net than the training benchmark. We anticipate a decrease in zero-shot performance compared to the results of subsection 4.2, indicating a likely need for policy fine-tuning.

Both the new design and baseline policies will undergo fine-tuning for an equal number of iterations. We will compare their performances based on rewards and solution quality metrics. Our expectation leans towards the new GNN-equipped design demonstrating superior adaptability and performance after fine-tuning.

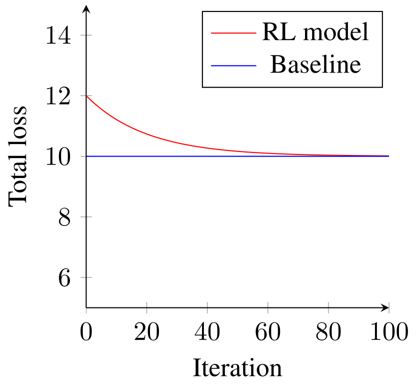

Figure 2: Placeholder figure for total loss vs. time. Not related to any aforementioned experiments.

Table 1: Placeholder table for the experiments. Not related to any aforementioned experiments.

Benchmark specs				Results		
Name	Canvas size	# of nets	Avg. # of pins per net	Runtime (s)	Max. memory usage (MB)	Total wirelength
test1	8×8	5	2.5	1.25	250	100
test2	6×6	3	1	0.60	150	70

References

[1] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 7 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[2] P. Almasan, J. Suárez-Varela, K. Rusek, P. Barlet-Ros, and A. Cabellos-Aparicio, "Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case," *Computer Communications*, vol. 196, pp. 184–194, 12 2022.

[3] T. Wang, R. Liao, J. Ba, and S. Fidler, "Nervenet: Learning structured policy with graph neural networks," 2018. [Online]. Available: http://www.cs.toronto.edu/

[4] H. Liao, W. Zhang, X. Dong, B. Poczos, K. Shimada, and L. B. Kara, "A deep reinforcement learning approach for global routing," *Journal of Mechanical Design, Transactions of the ASME*, vol. 142, 6 2020.

[5] H. Chen, K. C. Hsu, W. J. Turner, P. H. Wei, K. Zhu, D. Z. Pan, and H. Ren, "Reinforcement learning guided detailed routing for custom circuits." Association for Computing Machinery, 3 2023, pp. 26–34.

[6] A. Mirhoseini, A. Goldie, M. Yazgan, J. W. Jiang, E. Songhori, S. Wang, Y. J. Lee, E. Johnson, O. Pathak, A. Nazi, J. Pak, A. Tong, K. Srinivasa, W. Hang, E. Tuncer, Q. V. Le, J. Laudon, R. Ho, R. Carpenter, and J. Dean, "A graph placement methodology for fast chip design," *Nature*, vol. 594, pp. 207–212, 6 2021.

[7] S. Yue, E. M. Songhori, J. W. Jiang, T. Boyd, A. Goldie, A. Mirhoseini, and S. Guadarrama, "Scalability and generalization of circuit training for chip floorplanning." Association for Computing Machinery, 4 2022, pp. 65–70.