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, which can be modeled as a pathfinding problem, has always been one of the most challenging stages of integrated circuits (IC) design, both in terms of complexity and time consumption. Even with state-of-the-art commercial CAD tools, complex designs can take hours and maybe days to be routed. Thus, there has been a surge of interest to use machine-learning-based approaches to tackle these difficulties. In a previous work, we have proposed a novel reinforcement learning (RL) approach for routing. Although the RL router can deliver better solutions than a baseline, it suffers from poor generalizability, that is, every problem has to be solved by training a policy from scratched. Hence, in this work, we propose a novel graph neural network (GNN) and implement it as the policy network of the RL router. This enables the agents to generalize their learned routing strategy to new, unseen designs and saves computation time.

1 Introduction

IC routing is the process of connecting different electrical contact points (pins) on a chip with metallic wires. For example, the CPU and the memory in a computer chip must be connected with wires such that they can transfer data and do computing. We can model routing as a pathfinding problem, where the chip is abstracted into a 2-dimensional grid, and the pins, defined by (x, y) coordinates, sit on the nodes of this grid. The task is then to find paths between the specified nodes by traversing through the grid edges. In addition, each edge has a feature called the capacity, which represents the maximum number of wires permitted to use this edge. This constraint comes from the fact that in a real physical world, each wire has some thickness, and the number of wires allowed within a chip region is finite. If the usage of an edge exceeds its capacity, the edge is overflowed, which is highly undesirable. Hence, to find a optimal routing solution, we need to determine optimal paths to connect pins to minimize the total wirelength while maintaining zero edge overflow.

A set of pins that need to be connected together is called a net, and during a routing process, there will be many nets that must be routed, they form a set called the netlist. For the rest of this proposal, we will use the term nodes interchangiablly with pins.

In our previous work, we proposed an multi-agent RL router to solve the IC routing problem, where each agent is responsible for routing one net, and they take actions concurrently. We designed the agents to be homogeneous and fully-cooperative, and hence, they are governed by one super policy. The RL router was trained using proximal policy optimization (PPO) (REF), and was evaluated using benchmarks. The results showed the RL router can significantly outperform an A* baseline, however, the policy trained for one benchmark does not generalize to other unseen although very similar problems. After extensive analysis, we conclude the reason is because the policy neural network, consists of several full-connected (FC) layer, cannot learn rich information from the state encodings and thus has poor generalizability.

As opposed to a multi-layer perceptron (MLP), GNNs can generalize well when dealing with graphical data (REF), which is the case for routing problem that can be modeled as a grid graph. Therefore, in this project we propose a novel message-passing GNN architecture, which will be embedded into our developed RL router as the policy network. The GNN will take the routing grid graph from the environment as the state input and output actions. The aim is to enhance the router, such that it can generalize trained policies to unseen but similar problems, and hence, achieve significant improvement in terms of computation time.

2 Related work

Many works in the literature have verified the effectiveness of using GNN as the policy/value network of a RL model to promote generalization, given the environment that the agent is trying to solve can be modeled as a graph. However, to the best of our knowledge, no work has been directly done in terms of using RL as an approach of IC routing, and is combined with a GNN for generalizability.

[1] proposed an RL IC router that is trained by deep Q-network (DQN), i.e., using a feed-forward NN as the value network of the RL agent. However it suffers from the same limitation as our previous work, the results show no signs of generalization. In the work of [2], the authors proposed a GNN as the value network of an RL agent trained by DQN, the solution achieved good generalization in the field of network routing, which is a similar problem as IC routing. This work is the main aspiration of our proposal. Similarly, [3] and [4] reported similar concepts of implementing GNN as the policy network of an RL model, where the problem can be abstracted into a graph. Additionally, [5] and [6] used GNN as the encoder layer to their RL policy and value networks.

3 Methodology

In this proposal, we confine the model discussion to a very simply case of IC routing. This is to ensure that a prototype can be swiftly developed. As the project progresses, more complexities will be added such that the model becomes more compact.

3.1 Modeling IC routing as a grid graph

Figure 1 shows a simple IC routing problem with just one net modeled as a grid graph. The gird has a size of 4x4, edge capacity of 1, and a net consists of two pins located at (0,0) and (3,3). We can observe that this graph has 16 nodes in total, and let us assume the RL agent starts at node (0,0), its target sits at (3,3). This graph is characterized by a node feature matrix $\mathbf{X}$, an edge feature matrix $\mathbf{E}$, and the adjacency matrix $\mathbf{A}$. Matrix $\mathbf{X}$ is of dimension 16x2, with each row representing a node's binary feature vector. The first element denotes the agent's presence, while the second indicates the existence of a target at the node. Both $\mathbf{E}$ and $\mathbf{A}$ are 16x16 matrices, with $\mathbf{E}$ encapsulating the scalar capacities of the edges.

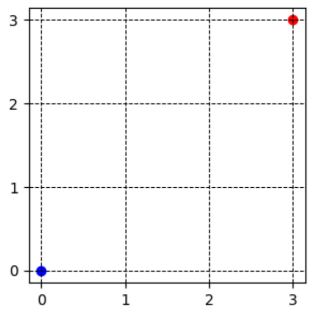

Figure 1: The grid graph of a simple IC routing problem.

3.2 GNN architectural design

The GNN will utilize the message passing mechanism. It takes $\mathbf{X}$, $\mathbf{E}$, and $\mathbf{A}$ from the RL environment as input, and outputs the probability distribution of all the actions. The message network will be implemented by MLP, the messages will be aggregated by element-wise summation, and the node states will be updated by a gated recurrent unit (GRU). After the messaging passing stage, the node states will be feed into another MLP ended with a softmax layer to produce action probability distribution.

Architecture of GNN can't reply on the graph topology. Like #nodes. We want the GNN to generalize to graphs of any size.

3.3 RL policy training

The proposed GNN will be embedded into our RL router as its policy network, which will be trained using PPO with RLlib. The parameters of the GNN will be updated by stochastic gradient descent, guided by the value feedback from the value network.

3.4 Project timeline

- Extended literature review (5 days)
- Designing the GNN prototype, combing with the RL router, and training on a simple benchmark (14 days)
- Testing the updated RL agent on benchmarks both similar to and different from the training benchmark (1 day)
- Analyzing the results and getting a measure of the generalizability (3 days)
- (bonus) Scaling up the model and design to support multi-agent concurrent routing (7 days)
- Writing the project report (14 days)

4 Experiments

We will implement a baseline MLP policy network, and train it with a simple benchmark.

4.1 Comparing training efficiency

In this experiment, we will train the GNN equipped router as well as the baseline router, both using a same benchmark. We will compare their plots of reward versus training iteration. The goal is to see how well the GNN equipped router trains comparing to the baseline. According to [4], we expect the new design will have similar or even better training efficiency.

4.2 Comparing zero-shot performance

In this experiment, we will use the two zero-shot policies obtained during the previous experiment (one from the new design, one from the baseline) to run inference on several unseen benchmarks, which are netlists with canvas sizes, number of nets, and total number of pins per net similar to that of the training benchmark. We would then compare the results in terms of rewards, and solution qualities. We expect the new design would outperform the baseline.

4.3 Comparing fine-tuning performance

For this experiment, we will first run zero-shot inference on benchmarks with different number of nets, chip canvas size, and number of pins per net than the training benchmark. The goal is to see how well can the GNN scale to more complicated problems. We expect the zero-shot performance to be less sound than the results of the previous experiment, and hence, it is likely the policies need fine-tuning. We will fine-tune the polices for the same number of iterations, and compare the results in terms of rewards, and solution qualities. We expect the new design to do better than the baseline.

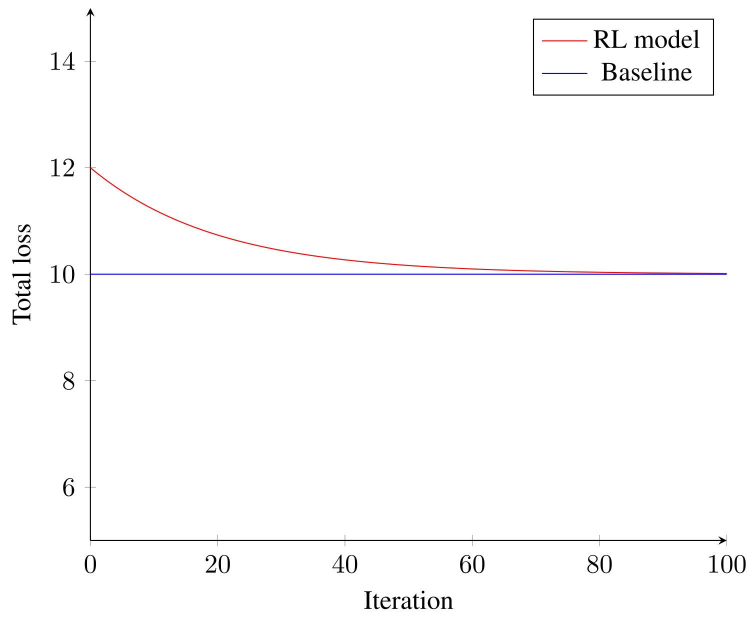

Figure 2: Placeholder figure for total loss vs. time. Not related to any aforementioned experiments.

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] 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.

[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] 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.

[4] 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/

[5] 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.

[6] 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.