Detailed routing using GNN-based multi-agent reinforcement learning

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 multi-agent 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 your 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.

TODO: RL approach, poor generalizability, propose GNN as policy network, should probably trim down the wording for routing, as the focus for this project is about designing GNN to achieve RL generalizability.

2 Related work

TODO: Mention 1 or more analytical routers, and mention some routers that also incorporate RL and/or GNNs. At the end, mention that no other work has used a multi-agent RL approach which also involves a GNN (no one even has used vanilla multi-agent for that matter). I suggest you also include a very brief outline of RL and GNNs in here, before going into detail for the papers that have used them.

Preprint. Under review.

3 Methodology

TODO: Briefly explain what is multi-agent RL and its advantages for our given problem. Also mention the paper which used a GNN as the policy network for the RL agent [1]. Also discuss the overall architecture of the GNN that we're going to use, and mention which libraries we're using (i.e., PyTorch, PyG, and RLLib). Adding a figure depicting the routing problem in a grid would also be very helpful.
Also, don't forget to come up with a rough timeline and replace the 'X' values below!

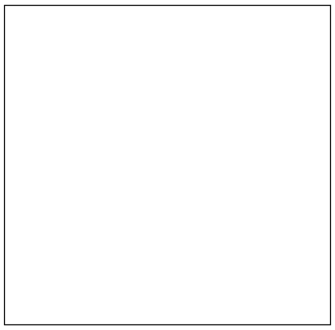

Figure 1: A simple routing scheme

Here is a list of tasks that needs to be accomplished in order:

- Extended literature review (X days)
- Getting familiar with the required Python libraries (X days)
- Finding appropriate benchmarks and converting them into the same format (X days)
- Implementing the RL agent (X days)
- Training the RL agent and tuning hyperparameters (X days)
- Testing the RL agent on benchmarks both similar to and different from the training designs (X days)
- Creating figures and tables (X hours)
- Writing the project report (X days)

4 Experiments

As stated in Section 3, our goal is to develop a model that not only routes different nets simultaneously, but is also able to generalize to unseen designs based on its learned policy.

4.1 Testing on benchmarks with similar circuit size to the training data

The goal of this experiment is to ensure that the model has learned a good policy based on its training data. We would test the model on circuits with canvas sizes, number of nets, and total number of pins per net similar to that of the training benchmarks. We would then compare the results to other routers in terms of runtime, memory usage, and solution quality.

4.2 Testing on benchmarks with different circuit size than the training data

We aim to test the *generalizability* of the model in this experiment, by applying to designs with different number of nets, chip canvas size, and number of pins per net. However, we're won't use benchmarks that are *significantly* different from the training designs, as the complexity of routing increases exponentially within very large designs. The chip canvas size for the test designs won't be much larger than the training designs, and we're more interested in observing how our model handles *different net topologies* in similarly-sized circuits.

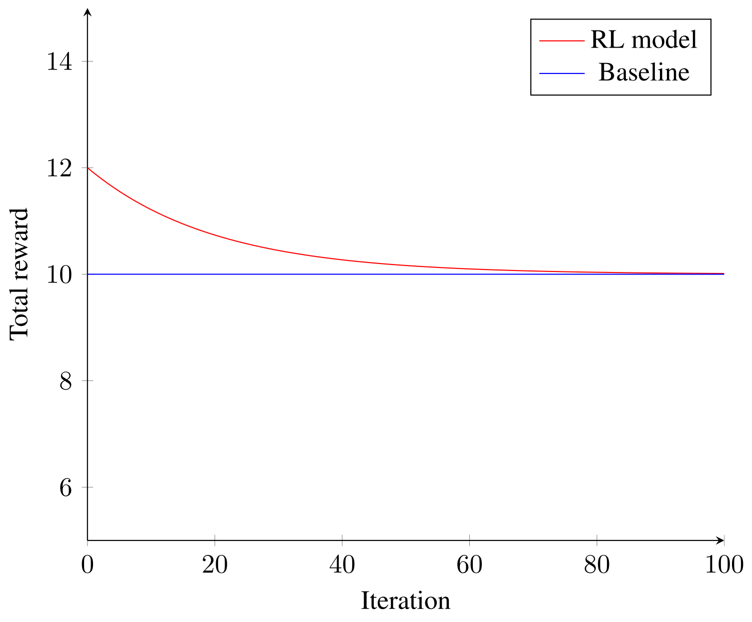

Figure 2: Placeholder figure for total reward vs. time

Table 1: Placeholder table for the 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] 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.