Congestion Modeling Quarter 1 Project

Pranav Rebala
University of California San Diego

Lindsey Kostas
Qualcomm

Abstract

Increasing complexity in chips means that it is more difficult to understand where congested areas in netlists are before the physical placement. Since congestion is a bottleneck for the efficiency of the chip, we want to create a model that can predict where the congestion is early in the design process. Using a machine learning approach, we can create a representation of our chip that can be trained on to generalize to future chips. We reimplement a paper that creates a new model architecture which includes directed hyperedges and virtual nodes to better capture long range interactions. We also compare metrics between the models to evaluate how good the reimplementation is.

Code: https://github.com/prebala123/capstone-project

1 Introduction

Chip design is a very difficult process because of the size and complexity of interactions between different cells in chips. The goal is to optimize several key metrics including power consumption, performance, and chip size to reduce cost and improve efficiency. However, reducing the area of the chip has a tradeoff of increasing congestion in certain areas where there are a lot of cells and wires. This leads to difficulty in physical creation of the chip as well as performance issues due to overheating in congested areas. Therefore, we need a way to predict congested areas before the physical placement and routing stages because physical errors are a lot more costly. Given that chips can be represented as graphs with cells representing nodes and wires representing edges, we can use graph neural networks to learn features of congestion on known chips and predict congestion on newly designed chips in the future.

Our first project involved learning about graph structures and graph algorithms using the New York City subway dataset. We created a graph network using the subway stations and analyzed the most congested stations and routes at certain dates and times. Our work gave insights into how to analyze congestion in graphs, as well as how to set up representations using node and edge features to best preserve graph information and interactions. We then applied this to chip data by reimplementing a paper on effective neural networks for circuit representations (Luo et al., 2024). Their methods involve creating a new model for directed hypergraphs called Directional Equivariant Hypergraph Neural Networks (DE-HNN) to predict congestion from circuit netlists. By using a directed hypergraph representation as well as using virtual nodes to improve long range message passing, this model better captures the properties of circuit netlists than previous models.

Our data includes has netlists for xbar and superblue chips with each one having connectivity information between its cells and nets. We also have general information about each cell type that can be used to encode some of its features. The congestion information is given as demand and capacity of regions of the cell, and is what we want to be able to predict. By using previously known congestion data for circuits, we want to create a model that can generalize to unseen circuits in the future.

2 Methods

2.1 Subway Analysis

In our first project, we analyzed the New York City subway system using graph representation techniques. We used the NetworkX package to create a directed graph to show the directions of rides between stations. We also created multiple edges between each station to represent the average ridership at different dates and times.

Our analysis mainly involved filtering node and edge features to get the specific data we were looking for. For example if we wanted to find stations that were the most congested, we filtered all of the edges by station, then added up the ridership to find the number of people passing through each station. Also if we wanted to find which routes were the most congested at certain times, we could filter the edge features by time then sort by the average number of riders. The analysis of congestion in this dataset helped us better understand how to represent data as graphs as well as what congestion looks like in graphs.

2.2 Chip Congestion

In this project we recreated the DE-HNN model to predict congestion from chip netlists. We modeled the chip as a bipartite graph in which we had sets of nodes and nets. This preserves the graph topology better and improves efficiency since large nets can create large cliques in regular graphs. We also used directed nets to preserve which nodes are drivers and sinks, since they could have different properties for prediction.

The DE-HNN model uses message passing to update node and net features through every layer. This way every node takes in information about its neighboring nodes and nets, as well as graph structure information so it can make a better prediction. We also have a hierarchy of virtual nodes to create long range connections because chip nodes can depend on other nodes not in the local neighborhood.

Node updates: We update the features of each node by aggregating the features of the nets that it is a part of. We also add in the virtual node embedding to get long range connections.

$$m^{\ell}(v) = \mathrm{MLP}_4^{\ell}(vn) + \sum_{\sigma' \in \mathcal{L}(v)} \mathrm{MLP}_1^{\ell}(M^{\ell-1}(\sigma'))$$

Net updates: We update the net features by aggregating the sink features then concatenating with the driver features. This preserves the direction which can be useful in making predictions.

$$M^{\ell}(\sigma) = \mathrm{MLP}_3^{\ell}\left[m^{\ell}(v_{\sigma}) \oplus \left(\sum_{v' \in S_{\sigma}} \mathrm{MLP}_2^{\ell}(m^{\ell}(v')) \right) \right]$$

Virtual Node updates: We update the first level of virtual nodes by aggregating all of the nodes it is connected to as well as the top level virtual node. We update the top level node by aggregating all of the first level virtual nodes.

$$m^{\ell}(vn) = \mathrm{MLP}_6^{\ell}(svn) + \sum_{v} \mathrm{MLP}_5^{\ell}(M^{\ell-1}(v))$$

$$m^{\ell}(svn) = \sum_{svn} \mathrm{MLP}_7^{\ell}(M^{\ell-1}(vn))$$

2.3 Experiments

We ran the DE-HNN implementation on both the xbar data and the superblue data to get metrics on prediction accuracy. We did cross design experiments for all data sets in order to see how the model would generalize to unseen netlists in the future. Due to the large size of the superblue datasets, only 5 of them were used in training. For both datasets one chip was used for validation and one for testing.

Our node and net features were similar to the ones used in the original paper. The node features include width, height, cell type, orientation, in degree, out degree, and positional embeddings. The net feature was the net degree. The targets that we tried to predict were classification for congested areas and regression on demand and wire length.

3 Results

Table 1: Experimental Results

Model	Demand		HPWL		Congestion	
	RMSE	MAE	RMSE	MAE	Precision	Recall
Paper DE-HNN	6.037	4.670	1.677	1.242	0.660	0.986
Xbar	7.646	5.471	2.340	1.425	0	0
Superblue	10.89	8.736	18.98	4.027	0	0

4 Discussion

The demand regression results for the reimplementation were close to the RMSE and MAE of the model in the previous research, with the xbar results being very close. The reason why our model is worse is that we didn't use all of the features implemented in the paper. Due to time constraints we only created a subset of features, which means that we could have missed out on some good predictors.

The wire length regression results were similar to the demand regression because the xbar metrics were close to original paper. However the results on the superblue chips were not as good. This is likely because this was a regression on the nets rather than the cells and we did not create as many net features as node features. In addition the virtual nodes are not connected to the nets so even less information is getting passed. An area of improvement would be to implement the rest of the net features from the paper to see if it helps the prediction.

The congestion classification of our model was very bad and could not differentiate the two classes. Since over 90% of the cells are not congested, the model predicted not congested every time to improve the accuracy. The model could not deal with the unbalanced classes and therefore could not make real predictions. One future area of improvement would be understanding why our implementation is not able to deal with this the same way as the paper's and finding a way to improve it.