LayNet: Layout Size Prediction for Memory Design Using Graph Neural Networks in Early Design Stage

Hye Rim Ji[1,2*], Jong Seong Kim[2], Jung Yun Choi[2] and Jee Hyong Lee[1†]

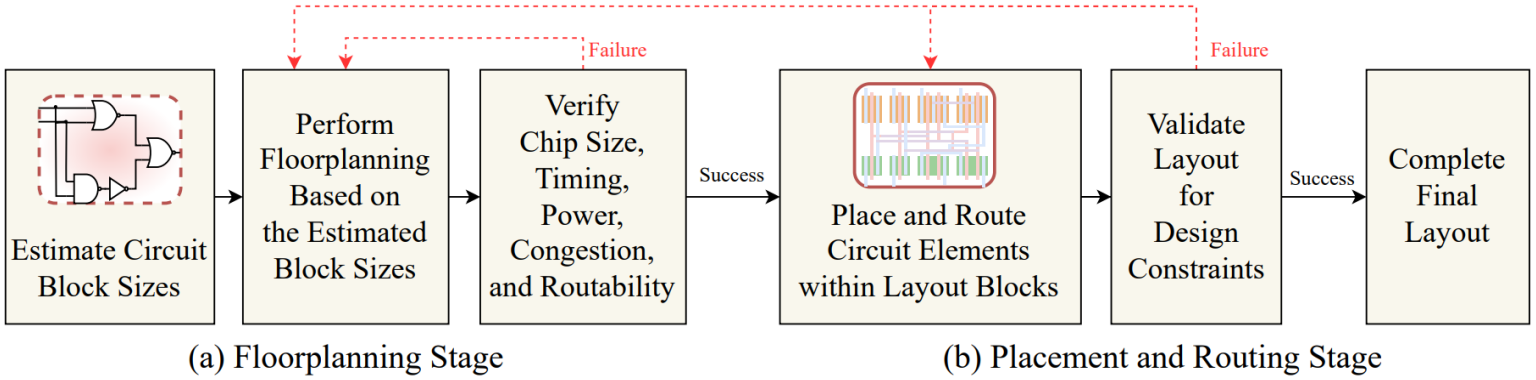
[1] *Sungkyunkwan University, Suwon-si, Gyeonggi-do, Korea*
[2] *Samsung Electronics Co, Ltd., Hwaseong-si, Gyeonggi-do, Korea*

Outline

- **Motivation**
- **Difficulty in Predicting a Layout Size**
- **Layout Size Prediction Model**
- **Efficient Techniques**
- **Experiments**
- **Conclusion**

Motivation

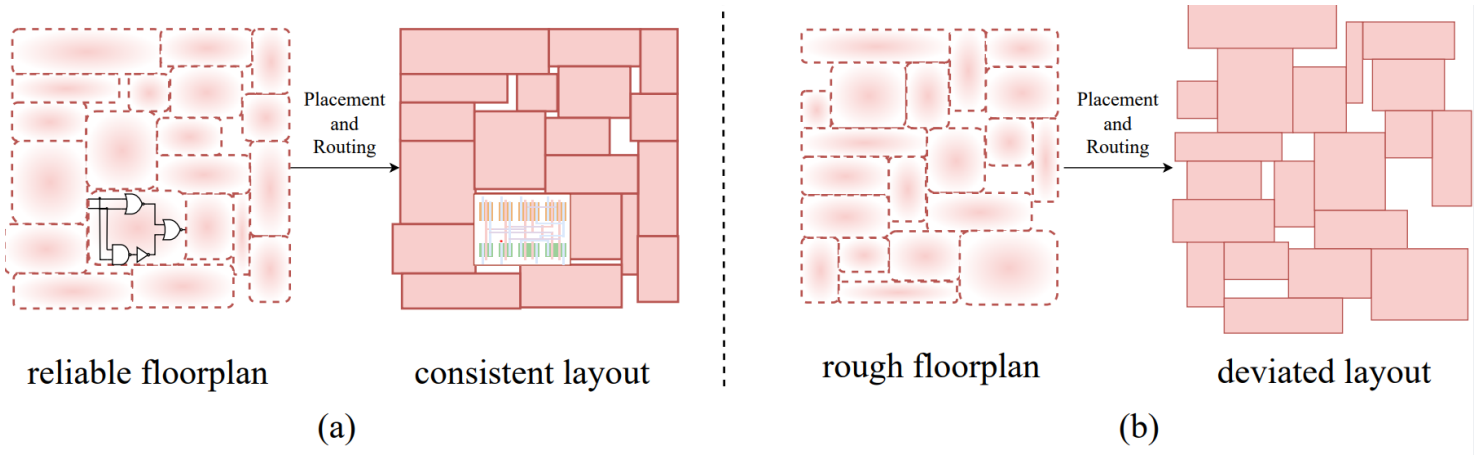
- Layout Design Flow in Full Custom Design



- In the floorplanning stage, engineers generate a floorplan with approximated block sizes based on their experience and intuition.
- After the engineers generate a feasible floorplan, they perform placement and routing to generate a final layout.
- If the final layout does not meet the design constraints, the engineer must return to the previous stages.
- This iterative design process takes up most of the memory design cycle and costs.

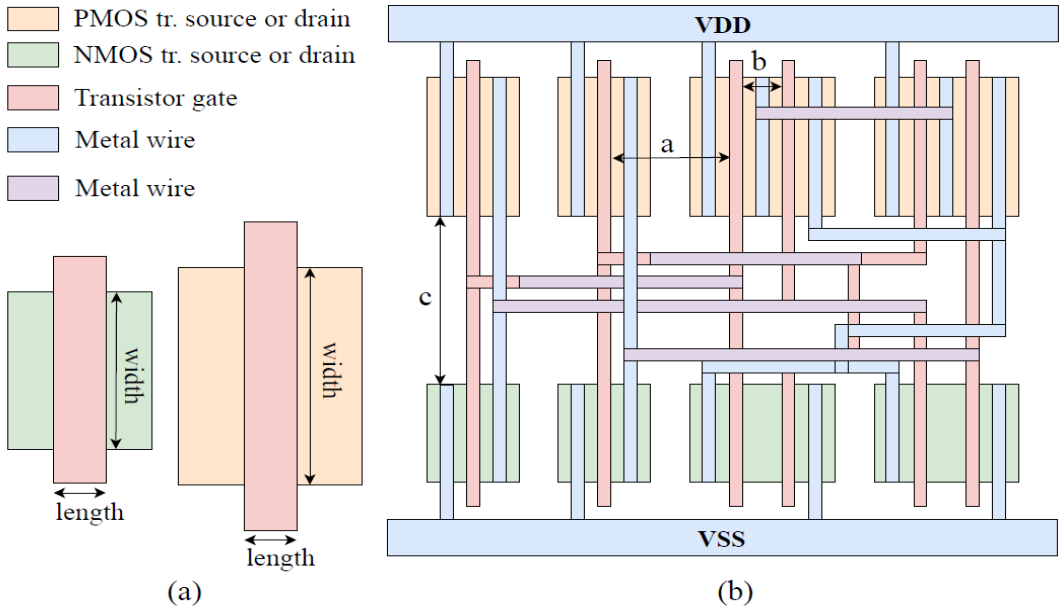
Motivation

- Comparing Layout Variations Based on Block Size Prediction Accuracy



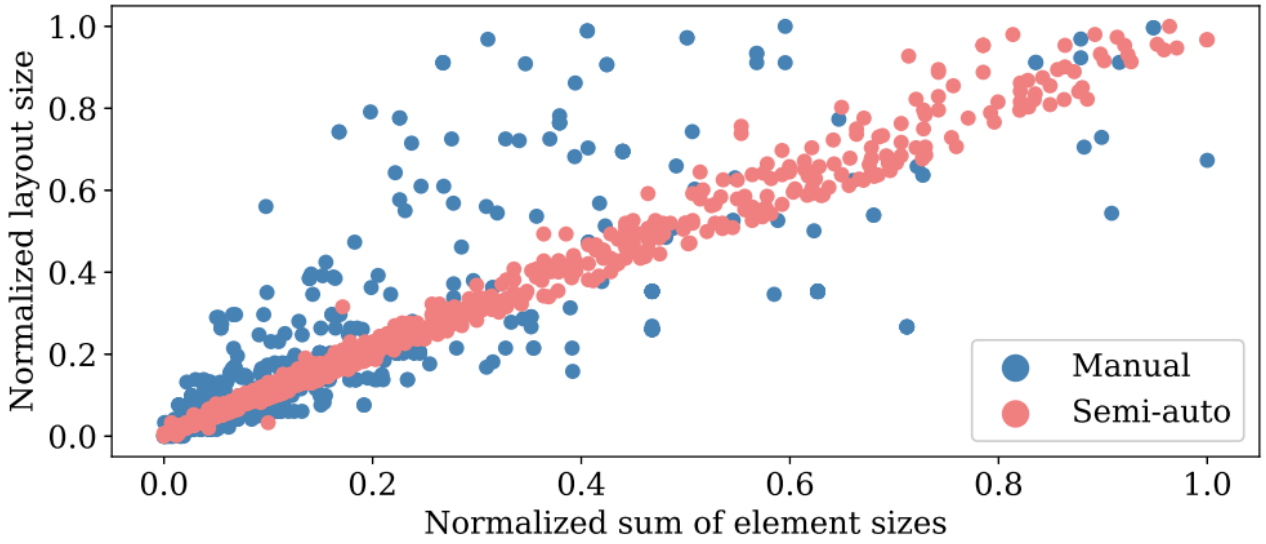
- If the sizes of circuit blocks are accurately predicted, engineers can generate a more reliable floorplan, generate a consistent layout, and reduce the number of design iterations.
- In this paper, we focus on predicting the exact block layout sizes of circuit blocks for a reliable floorplan in the early design stage.

Difficulty in Predicting a Layout Size



- Layout size: each element size + space size between adjacent elements
- The size of circuit elements is mathematically predictable, but the spaces are not.

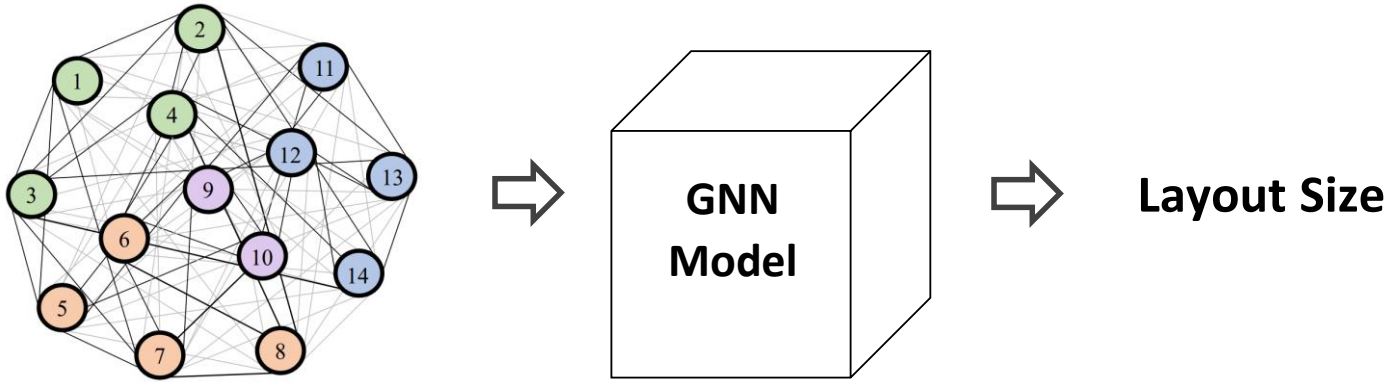
Difficulty in Predicting a Layout Size



- The wide distribution for manually generated layouts highlights the difficulty in predicting layout size based solely on element size sums.
- This highlights the importance of considering not only the sizes of the elements, but also the spaces between adjacent elements for accurate layout size prediction.

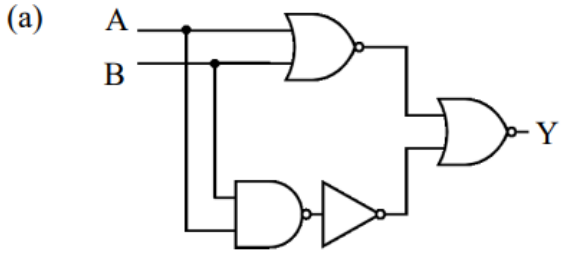
Layout Size Prediction Model

- LayNet

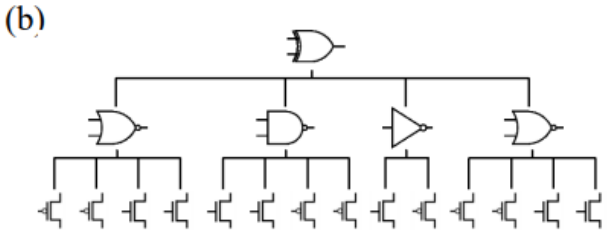


Convert a Circuit to a Weighted Graph

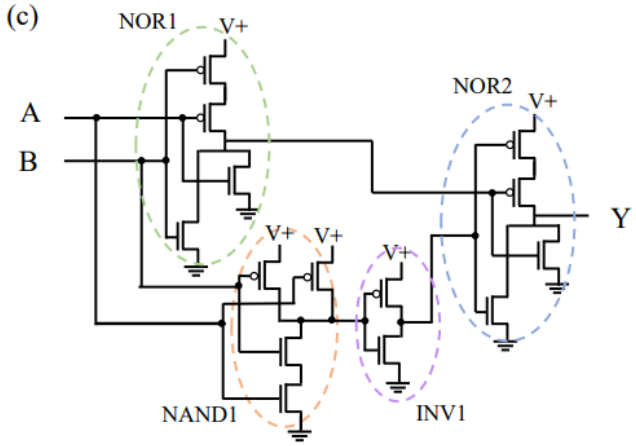
- To express the circuit design as a graph,
  a circuit designed in a hierarchical structure is converted into an element-based circuit.



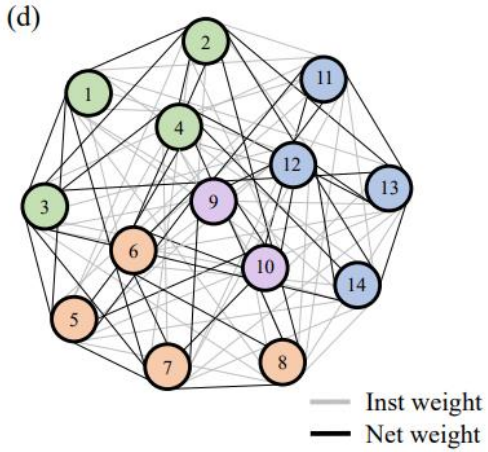
Instance-based XOR circuit



Hierarchical tree of the XOR circuit



Flattened element-based XOR circuit



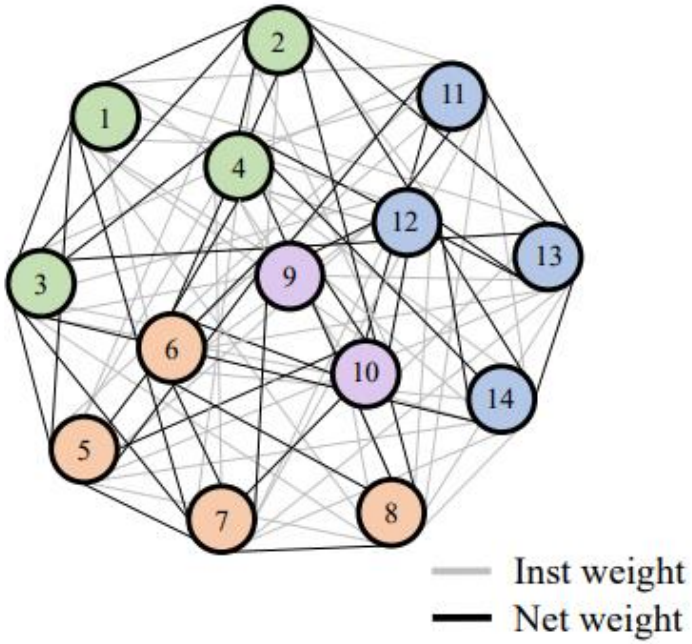
Weighted circuit graph

Convert a Circuit to a Weighted Graph

- Node & Node Features

Node: Each element of the element based design

Features: the element type, fingers, number of input pins, number of output pins, number of in-out pins, and size information of the element

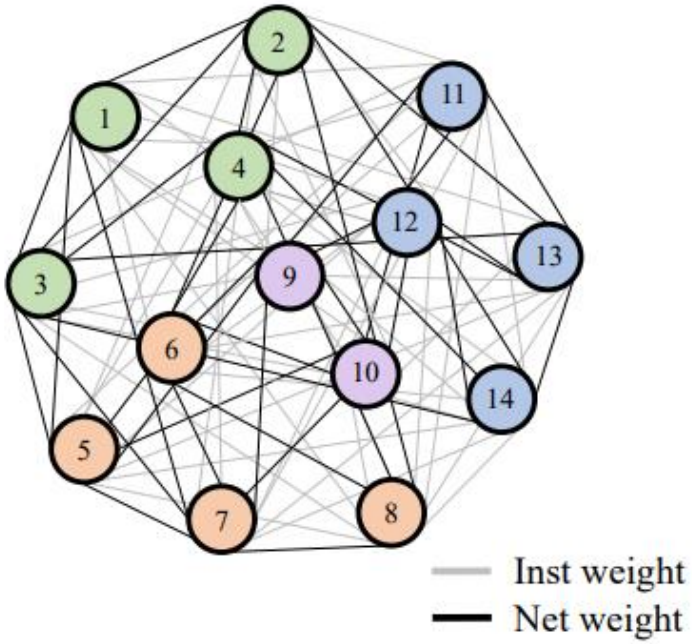


Convert a Circuit to a Weighted Graph

- Edge & Edge Weights

Edge: The degree to which two nodes are located close to Each other on the layout

Weight: Net Weight & Instance Weight



$$\mathcal{W}(n_1, n_2) = \alpha * \mathcal{W}_{\text{inst}} + (1 - \alpha) * \mathcal{W}_{\text{net}}$$

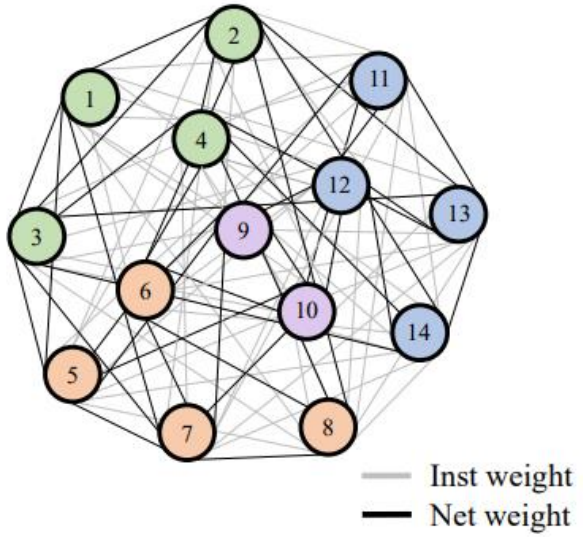
Convert a Circuit to a Weighted Graph

- 1) Net Weight

The possibility that elements will be adjacent to each other in a layout

Elements connected to a common signal ↓

Tendency for the elements to be each other ↑



$$\mathcal{W}_{\text{net}}(n_1, n_2) = \sum_{s \in S_{n1,n2}} \frac{1}{N_s}$$

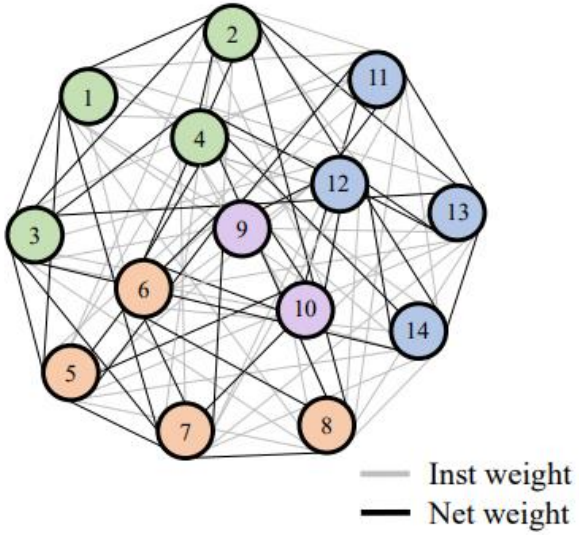
Convert a Circuit to a Weighted Graph

- 2) Instance Weight

  The possibility that elements will be adjacent to each other in the layout even though they do not share any signals

  The number of elements in an instance ↓

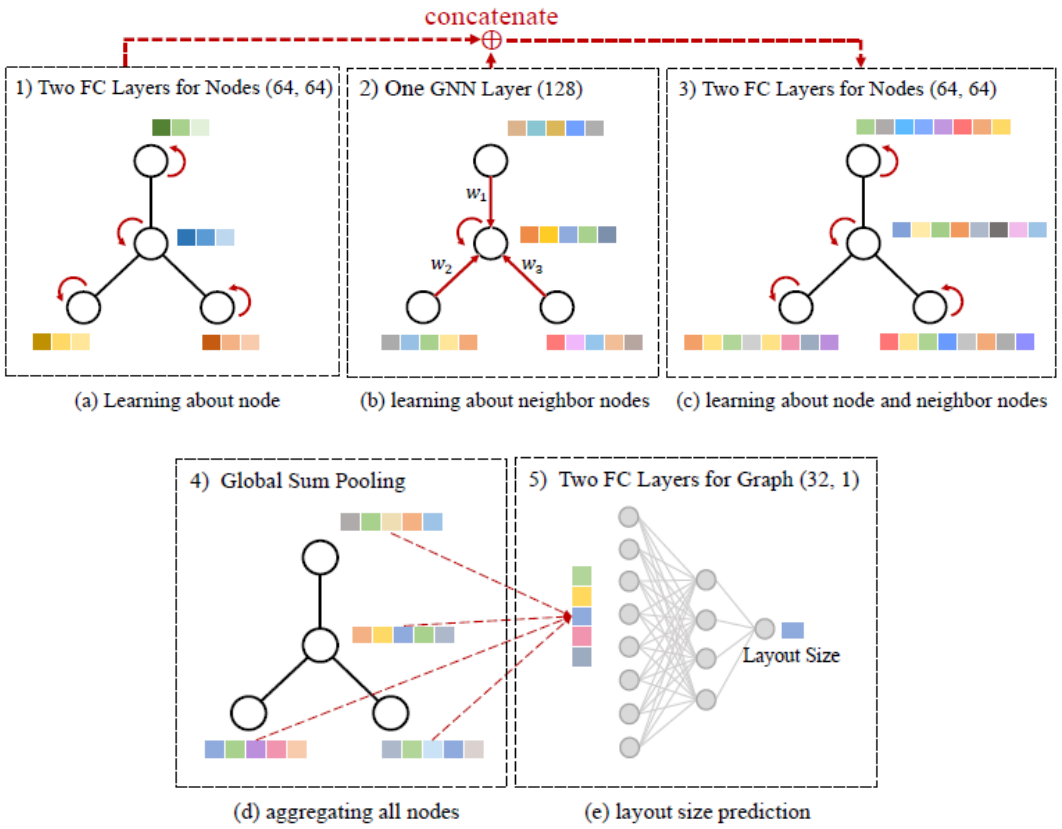
  Tendency for the elements to be placed adjacent to each other ↑



$$\mathcal{W}_{inst}(n_1, n_2) = \frac{1}{\mathrm{M_{n1,n2}}}$$

Layout Size Prediction Model

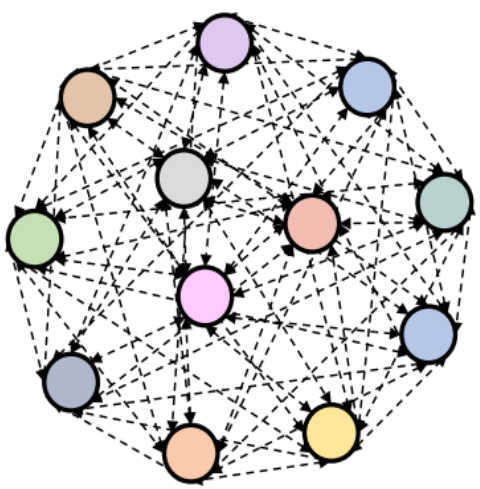
- LayNet Architecture



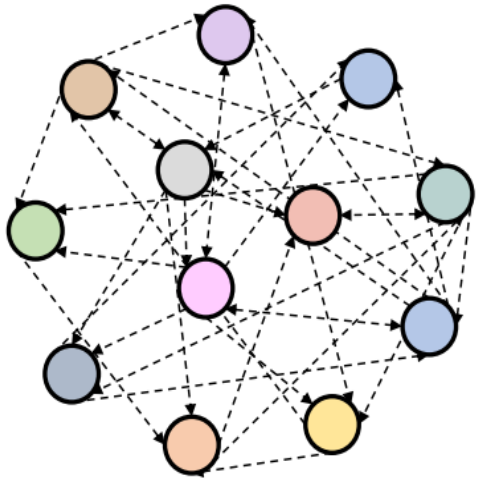
Efficient Techniques

1) Edge Selection: the number of edges ↓

- This reduces the number of edges from N 2 to kN, effectively reducing the memory requirement.



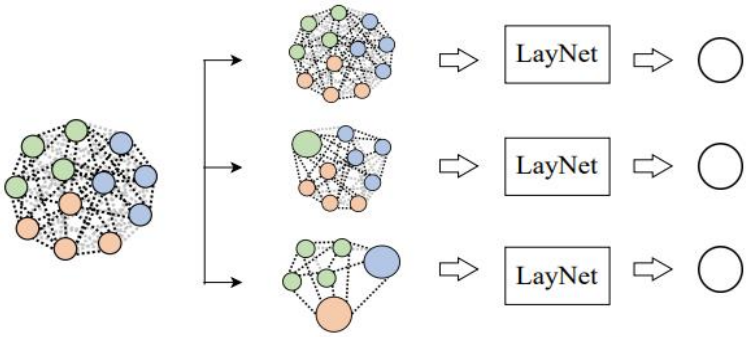
(a) All Connected Edges



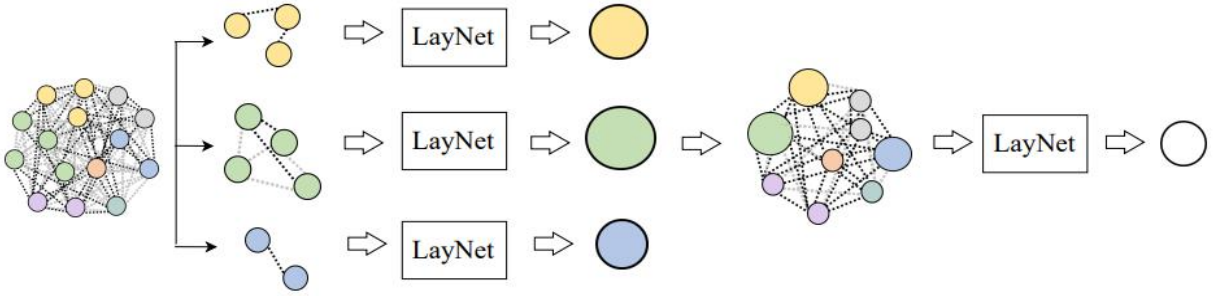
(b) Top K Weighted Edges

Efficient Techniques

2) Hierarchical Graph Learning: the number of nodes ↓



(a) Training on augmented hierarchical graphs



(b) Inference on reduced graph generated from subgraph predictions

Experiments

- DataSet: 6300 circuit and layout pairs (DRAM)

Circuit Size (# of Nodes)	Number of Circuits	
	Manually-generated	Semi-auto-generated
1-10	2094	685
11-100	1807	770
101-1000	489	392
1001-10000	49	14
Total	4439	1861

- Performance:

$$\text{MAPE} = \frac{100}{\text{n}} \sum_{k=1}^{n} \left| \frac{y_k - \hat{y}_k}{y_k} \right|$$

Experiments

- Comparison with Conventional Regression Models

Model	Manually-generated layouts			
	R^2	MAPE	stdev	Best HyperParams.
MLP	0.89	37.90%	159.13	$l = (128, 256, 256, 128)$
Linear	0.93	60.42%	106.81	-
SVM (linear)	0.83	21.81%	87.08	$C = 60000$, $\epsilon = 0.01$
RF	0.87	15.36%	46.50	$depth = 15$, $\#est = 500$
LayNet	**0.98**	**7.16%**	**13.21**	G.SAGE-mean, $\alpha = 0.7$

Model	Semi-auto-generated layouts			
	R^2	MAPE	stdev	Best HyperParams.
MLP	0.98	14.32%	10.60	$l = (256, 512, 256)$
Linear	**0.99**	9.17%	8.83	-
SVM (linear)	0.98	6.41%	7.03	$C = 1000000$, $\epsilon = 0.1$
RF	0.98	9.09%	12.24	$depth = 9$, $\#est = 50$
LayNet	**0.99**	**3.81%**	**6.28**	GCN, $\alpha = 0.3$

Experiments

- Performance Comparison of LayNet Using Difference Types of Circuit Graphs

Edge Conn.	Edge Weight	Model	Manual MAPE	Manual stdev	Semi-auto MAPE	Semi-auto stdev
net conn.	unweighted	G.SAGE-max	9.15%	21.58	4.20%	5.53
		G.SAGE-mean	9.17%	35.23	4.33%	5.40
		GCN	10.95%	35.95	4.40%	6.55
	trained	GAT	9.84%	19.46	4.30%	5.88
	w_{net}	G.SAGE-max	9.73%	33.47	4.87%	22.40
		G.SAGE-mean	9.44%	28.34	5.27%	20.00
		GCN	9.97%	35.23	4.52%	16.99
all conn.	unweighted	G.SAGE-max	10.63%	21.96	4.14%	5.67
		G.SAGE-mean	9.03%	42.87	4.24%	6.90
		GCN	9.27%	29.82	4.51%	5.62
	trained	GAT	10.19%	21.13	4.43%	6.77
	$\alpha\ w_{inst}$ + $(1-\alpha)\ w_{net}$	G.SAGE-max	8.45%	20.74	4.08%	6.37
		G.SAGE-mean	**7.15%**	14.50	4.05%	**5.13**
		GCN	7.16%	**13.21**	**3.81%**	6.28

Note: $\alpha = 0.7$ for manual layout; $\alpha = 0.3$ for semi-automatic layout.

Experiments

- Average Performance Comparison of Edge Selection and Hierarchical Graph Learning for Large Circuit Blocks

Manually-generated layouts

Edge Selection	All Connection Edges				Top 100 Weighted Edges				Top 50 Weighted Edges			
Node Reduction	0%	18.30%	45.60%	71.23%	0%	18.30%	45.60%	71.23%	0%	18.30%	45.60%	71.23%
MAPE	**4.18%**	4.28%	4.01%	5.87%	3.91%	4.01%	3.60%	4.49%	4.23%	3.70%	4.06%	**5.78%**
Inference TIme	**100.00%**	67.27%	23.57%	12.56%	38.41%	35.29%	8.62%	7.63%	30.76%	27.19%	7.79%	**7.07%**
Memory Usage	**100.00%**	71.88%	39.90%	16.05%	4.82%	3.93%	2.62%	1.38%	2.46%	2.06%	1.34%	**0.71%**

Semi-auto-generated layouts

Edge Selection	All Connection Edges				Top 100 Weighted Edges				Top 50 Weighted Edges			
Node Reduction	0%	21.57%	55.08%	73.35%	0%	21.57%	55.08%	73.35%	0%	21.57%	55.08%	73.35%
MAPE	**2.65%**	2.88%	3.35%	3.59%	2.35%	3.88%	3.57%	2.52%	2.77%	4.21%	2.73%	**2.36%**
Inference Time	**100.00%**	23.82%	16.46%	15.03%	67.33%	4.84%	4.11%	3.65%	44.38%	4.50%	4.00%	**2.64%**
Memory Usage	**100.00%**	79.25%	14.43%	7.56%	31.05%	2.44%	1.40%	0.82%	1.58%	1.24%	0.71%	**0.42%**

Note: The above table shows the results of experiments performed with LayNet using GCN.

Conclusion

- We proposed LayNet to predict the block layout size for a reliable floorplan in early design stage.
- We constructed circuit graphs with edge weights for adjacency between elements in the layout, and predicted block sizes with graph neural networks.
- The experiments demonstrated that LayNet was a powerful model for learning the element sizes and spaces.
- We also used edge selection and hierarchical graph learning to efficiently predict large circuit blocks.

Q & A

Layout Size Prediction Model

- LayNet Architecture