Graph of Circuits with GNNs for Exploring the Optimal Design Space

Authors:
Aditya .H. Shahane, Swapna Manjiri, Dr.Ankesh Jain, Dr.Sandeep Kumar
Indian Institute of Technology Delhi

37th *Neural Information Processing Systems (NeurIPS)- 2023*

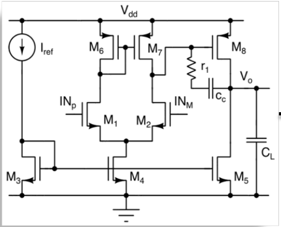
Outline

Circuit Designing Optimization

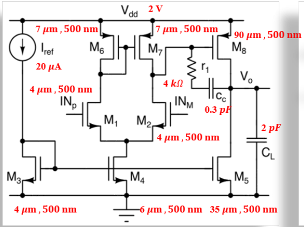
- ***Pre-layout design*** is a key stage in Analog circuit design flow.
- It can be represented as Parameter-to-Specification (P2S) problem.

The goal is:

- Find optimal design **parameters** (e.g: Transistor widths, lengths, reference current and voltages etc) to meet desired circuit **specifications** (e.g: Gain, Unity Gain Bandwidth, Gain Margin, Phase Margin etc)



Pre-Layout Schematic



max **F.O.M**

s.t Gain $\geq$ g1
UGB $\geq$ g2
GM $\geq$ g3
PM $\geq$ g4
Noise $\leq$ g5
Power $\leq$ g6

Parameter-to-Specification (P2S) problem

Figure 1: Circuit Design Optimization framework

Conventional Design Flow

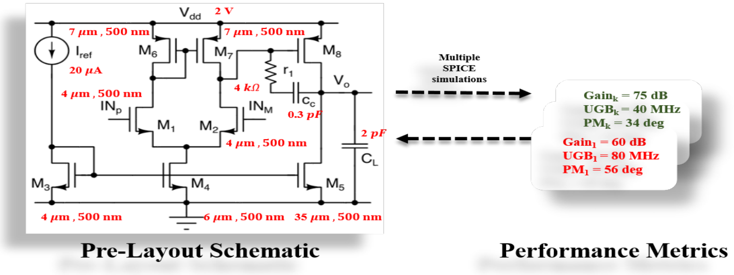


Figure 2: Iterative process until Convergence is met

Contributions

- We introduce Graph of Circuits Explorer, a graph representation of circuit instances incorporating feature and label information.
- Graph of Circuits Explorer-Graph Surrogate Model (**GCX-GSM**), incorporating semi-supervised learning was proposed which acts as a proxy to **SPICE** simulations.
- GCX-GSM is further integrated with two newly introduced optimization algorithms:
  - Efficient Analog Sizing via Constrained Optimization (**EASCO**)
  - Analog Sizing through Real-Time Online Graphs (**ASTROG**)

  to obtain the most optimal design parameters.
- We further propose a comprehensive technique of feature generation which facilitates knowledge transfer across different technology nodes and topologies.

Outline

Traditional Machine Learning and Deep Learning Frameworks

- *A Bayesian Optmization Framework for Analog Circuit Optimization*: BO-EI [1]
- *An Efficient Analog Circuit Sizing Method based on Machine Learning assisted Global Optimization*: ESSAB [2]
- *Differential evolution and swarm intelligence techniques for analog circuit synthesis.*[3]

Limitations:

- High Time-complexity.
- Reliance on extensive labeled datasets.

Introduction to Graph representation of Circuits

What is a Graph?

- A graph is a data structure used to store semantic information with the help of structure formed using nodes and edges.
- A graph with features is denoted by $G = (V, E, W, X, Y)$ where $V = \{v_1, v_2, \ldots, v_n\}$ is the vertex set, $E \subseteq V \times V$ is the edge set and W is the adjacency (weight) matrix.
- $X \in \mathbb{R}^{n \times d} = [X_1, X_2, ..., X_n]^T$ and $Y \in \mathbb{R}^{n \times m} = [Y_1, Y_2, ..., Y_n]^T$ are the feature matrix and label matrix respectively.
- $X_i \in \mathbb{R}^d$ and $Y_i \in \mathbb{R}^m$ are feature and label vectors for node i in graph G.

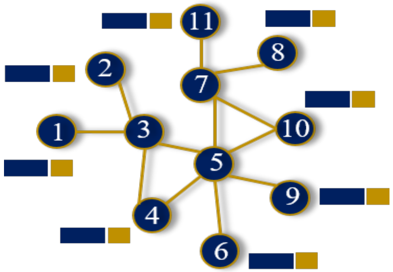


Figure 3: Graph representation

Introduction to Graph representation of Circuits

Graph representation of Analog and Mixed signal circuits

- Most common approach: Represent circuit's topology as a graph.
- Circuit components like resistors, capacitors etc are **nodes** and circuit wirings are **edges** [4] (ParaGraph).
- To this [5] (Pre-training GNNs) suggested each device terminal be considered node which makes edges featureless.
- This results in a **heterogeneous graph** and a corresponding graph-regression problem.
- Set of n graphs denoted by $\{G_1, G_2, \ldots, G_n\}$, a function f maps these graphs to corresponding performance metric (Y_i), where $Y_i = f\left(G_i\right).$

Introduction to Graph representation of Circuits

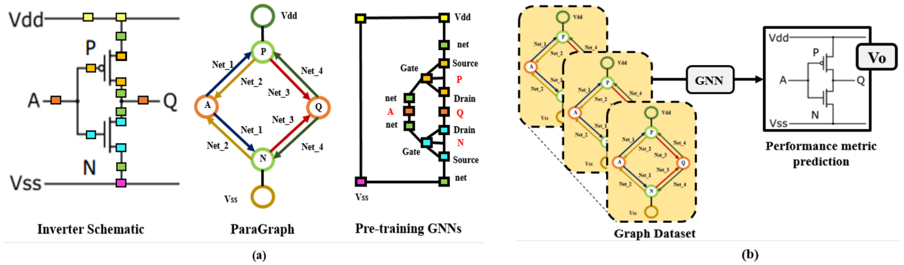


Figure 4: The figure (a) depicts two distinct methods for representing circuits as a graph. In [4] circuit components are depicted as nodes while nets carry edge-based information. To this, [5] suggested explicitly representing device terminals as nodes to avoid ambiguity when the device has identical net connections between two or more terminals. Meanwhile, in figure (b), we can see the training framework of ParaGraph [4] method. For the example circuit used, multiple graphs and corresponding label vectors are used for training, and prediction on the unlabelled graph is performed.

Motivation

- As the circuit size increases, the graph becomes densely interconnected, leading to a large number of nodes and edges, making it challenging to comprehend the overall behavior of the circuit.
- Creating a large labelled graph dataset is time-consuming and computationally expensive, which hinders effective GNN training.
- This framework lacks inter-graph connections, which reduces its reliability.

Motivation

- Graph regression problems cannot be applied to semi-supervised frameworks; reliance on labelled datasets is inevitable.
- Training over multiple graphs (n) with N nodes and E edges when passed through L layer GNN makes the process slow, and the high time-complexity $O(n * LN^2E)$ and space-complexity $O(n * N^2L)$ further adds to the computational burden.
- Transferring knowledge across different topologies becomes challenging due to varying graph sizes.

Outline

Problem Formulation

Given $n = l + u$ ***circuit instances and their parameters*** $X = (X_l, X_u)$ ***along with labels of only*** l ***number of circuit instances as,*** Y_l***, how can we predict the labels of the remaining*** u ***number of circuit instances*** Y_u***?***

- The circuit performance parameters or the labels Y_i are obtained using expensive SPICE simulation; large set of labelled data is a luxury.
- Limited labelled data where $l < u$ is a significant hurdle in learning models for EDA.
- Graph-based Semi-supervised learning attempts to address this by learning from both labelled as well as unlabelled data.

Problem Formulation

- Consider sets $T_l = \{(X_i, Y_i)\}_{i=1}^{l} \in \mathcal{X} \times \mathcal{Y}$ and $T_u = \{(X_i)\}_{i=l+1}^{l+u} \in \mathcal{X}$, where $\mathcal{X}$ is the domain set and $\mathcal{Y}$ is the label set.
- Suppose the elements of the set T_l be sampled from probability distribution $\mathbb{P}$ over the ordered pair $\mathcal{X} \times \mathcal{Y}$. Samples of the domain set $X_i \in \mathcal{X}$ will follow the marginal distribution $\mathbb{P}_{\mathcal{X}}$ of $\mathbb{P}$.
- For the limited labelled data scenario, the goal of a learning algorithm is to estimate conditional distribution $\mathbb{P}(Y|X)$ using both the labelled and unlabelled data as the training data.
- The guiding principle is that if two data points from the domain set are close in the intrinsic distribution $\mathbb{P}_{\mathcal{X}}$ then the corresponding conditional distributions should be similar. For the circuit prediction, it means that given two circuit instances with similar features $X_i \sim X_j$ then the corresponding label vectors should also be similar $Y_i \sim Y_j$.

Proposed Framework for Semi-Supervised Learning on Graph of Circuits

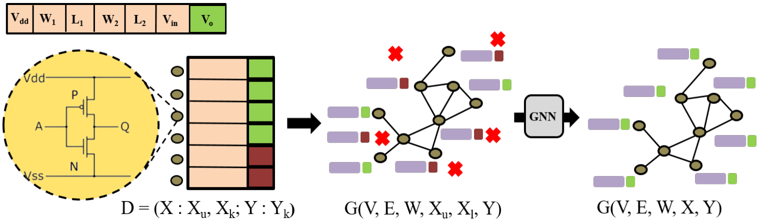
- Obtaining feature-aware geometrical representation; Construct a graph of circuits using circuit features $X = (X_l, X_u)$ in the form of an adjacency matrix A.
- Using the graph adjacency matrix, feature matrix, and available set of circuit performance matrix together, i.e., $G(X, A, Y_l)$ with the graph neural network pipeline to build a machine learning model.
- Additionally, incorporating available label information resulting in label informed Graph of Circuits.
- Predicting the performance parameters Y_u corresponding to X_u's.

Proposed Framework for Semi-Supervised Learning on Graph of Circuits

Algorithm 1 GCX(GSM) assisted Optimization

```
Input: Feature matrix $(X_l, X_u)$, label matrix $(Y_l)$
Output: Optimal design $(X_i^*)$
1. $G(V, E, W) \longleftarrow$ Learning Graph of Circuits Using $(X_l, X_u)$
2. $G\left(V, \tilde{E}, \tilde{W}\right) \longleftarrow$ Label informed Graph of Circuits using $(X_l, X_u, Y_l)$
3. $Y_u \longleftarrow$ Label prediction of unlabeled circuits using GNN/KGR with
   $G(V, E, W, X_l, X_u, Y_l)$; Select the most efficient surrogate model
4. GCX(GNN) $\longleftarrow G(V, E, W, X_u, X_l, Y_l)$ + GNN; surrogate model
5. $X_i^* \longleftarrow$ GCX(GNN) + EASCO/ASTROG
```

Proposed Framework for Semi-Supervised Learning on Graph of Circuits



V_{dd} | W_1 | L_1 | W_2 | L_2 | V_{in} | V_o
Vdd
P
A
Q
N
Vss
GNN
$D = (X : X_u, X_k; Y : Y_k)$
$G(V, E, W, X_u, X_l, Y)$
$G(V, E, W, X, Y)$

Figure 5: **Graph of Circuits Explorer (GCX)**: The dataset (D) contains both labeled (X_l, Y_l) and unlabeled (X_u) samples, each circuit is a node of the graph and corresponding feature vector of the associated node serves as node in the graph; the graph is learned (weighted adjacency-W) using the suggested formulation; and the learned graph is passed through a GSM for label prediction of unlabeled nodes. This forms a **GCX(GSM)** surrogate model. For the above circuit $X_{li} = (V_{dd}, W1, L1, W2, L2, V_{in})$ and $Y_{li} = Vo$.

How to Learn Graph of Circuits to Encode Circuit Similarity

- Feature matrix of circuit instances $X = [X_1, X_2, \dots, X_n]^T$ where X_i is the feature of i^{th} circuit which corresponds to i^{th} node of a graph.
- Underlying assumption is that the signal residing on the graph changes smoothly between connected nodes [6].
- If two circuit instances X_i and X_j have similar features, Euclidean distance $\|X_i - X_j\|^2$ should be smaller; w_{ij} should be large.
- The Dirichlet energy (DE) is used for quantifying the smoothness of the graph signals which is defined as:

$$\text{DE} = \frac{1}{2} \sum_{i,j} w_{ij} \|X_i - X_j\|^2$$

- The lower value of Dirichlet energy indicates a desirable configuration [6, 7].

How to Learn Graph of Circuits to Encode Circuit Similarity

- Nodes connected with stronger weights indicate similar features and performance indices, facilitating knowledge transfer to new circuit instances.
- Finally, we learn a weighted adjacency matrix W by minimizing the Dirichlet energy combined with other sparsity regularization terms which entails solving the following optimization problem:

$$w := \arg \min_{w \in \ w_m} \frac{1}{2} \sum_{i,j} w_{ij} \left\| X_i - X_j \right\|^2 - \alpha 1^{\top} \log \left(Sw \right) + \beta \|w\|_2^2$$

- where $\alpha > 0$ and $\beta \geq 0$ controlling the properties of the resultant graph, $\|w\|_2^2$ ensures the graph is sparse, while $1^{\top} \log \left(Sw \right)$ ensures that every node will have some connections.

How to Learn Graph of Circuits to Encode Circuit Similarity

- To enhance the graph structure, we propose the incorporation of additional information, specifically related to the labeled samples.
- Identify the prominent features (X_{imp}) corresponding to the labeled instances. A new feature vector, denoted as $\tilde{X}_i = [X_{impi}, Y_i]$ created, where Y_i represents the label of the i^{th} circuit instance.
- The new Dirichlet's energy represented by $\tilde{DE}$, can be expressed as follows:

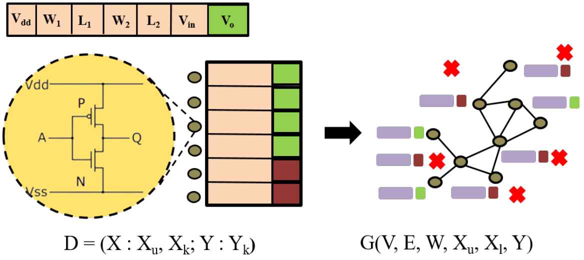
$$\tilde{DE} = \sum_{i,j} \tilde{w_{ij}} ||\tilde{X}_i - \tilde{X}_j||^2$$

- The newly obtained weighted adjacency can be represented as :

$$w_{LIG} = w + \tilde{w}$$

- w_{LIG} represents label-informed graph (LIG), w represents feature based weighted adjacency matrix and $\tilde{w}$ represents the weighted adjacency matrix derived from the additional information.

How to Learn Graph of Circuits to Encode Circuit Similarity



V_{dd} W_1 L_1 W_2 L_2 V_{in} V_o
Vdd
P
A
Q
N
Vss
$D = (X : X_u, X_k; Y : Y_k)$
$G(V, E, W, X_u, X_l, Y)$

Figure 6: **Graph of Circuits**: n points are uniformly sampled across the d-dimensional space; l points are simulated where $l << n$. Graph is learnt using the formulation discussed previously.

Homophily Analysis

Homophily in graphs is based on similarity between connected node pairs, where two nodes are considered similar if they share similar node label. Homophily ratio is defined based on [8]. While there exists multiple metrics to quantify homophily, we adopt assortativity as it is easily generalized to nodes and edges.

- **Label based Homophily**
  - Given a graph $G(V, E)$ and node label vector y, the edge homophily ratio is defined as fraction of edges that connects the nodes with similar labels.

$$h\left(G, \{y_i; i \in V\}\right) = \frac{1}{|E|} \sum_{(j,k) \in E} 1\left(y_j \sim y_k\right)$$

  - Where $|E|$ is number of edges in graph and 1 is the indicator function. A graph is typically considered highly homophilous when $h(.)$ is typically large (typically, $0.5 \leq h(.) \leq 1$).

Homophily Analysis

- **Global Node Label Assortativity**
  - Global assortativity measures the overall tendency of nodes with similar characteristics to be connected to each other in a network.
  - It provides an understanding of the network's general structural pattern.
  - The global node assortativity can be defined as:

$$r_n^{\text{global}} = \frac{\sum_i e_{ii} - \sum_i a_i b_i}{1 - \sum_i a_i b_i}$$

  - where, e_{ij} is the fraction of the edges corresponding to similar/dissimilar quantized distance ($\|y_i - y_j\|^2$) among the circuit instances, $a_i = \sum_j e_{ij}$, $b_i = \sum_i e_{ij}$.
  - When nodes with similar features tend to connect with each other, $r_n^{\text{global}} \to 1$, the graph exhibits strong homophily.

Homophily Analysis

- **Local Node Label Assortativity**
  - Local assortativity examines the tendency of nodes to form connections with similar nodes in their immediate neighborhood (i.e., their 1-hop neighbors).
  - Allows for a more fine-grained analysis of the network, as it considers node characteristics within smaller subgraphs or neighborhoods.
  - The local node label assortativity can be defined as :

$$r_n^{\text{local}}(u) = \frac{|\{v\} : v \in N(u) \wedge y_u \sim y_v|}{|N(u)|}; u \in V$$

  - where, $|N(u)|$ denotes the 1-hop neighborhood of node u, y_u denotes quantized label vector of node u.

Graph of Circuits Explorer-Graph Neural Networks (GCX-GNNs)

- After learning a graph using the feature vector description provided, the graph undergoes label propagation through the GNN architecture.
- In this process, each node represents a circuit instance and the graph consists of both labelled and unlabelled nodes.
- During each layer of the GNN, the nodes interact through message passing, and feature aggregation is carried out.

Graph of Circuits Explorer-Graph Neural Networks (GCX-GNNs)

- The unlabelled circuit instance acquires information from its labelled neighbors in the form of embeddings.
- The embeddings are iteratively updated across the L layers of the GNN, ultimately leading to the prediction of a label for the unlabelled circuit instance (node).
- The formulation for the embedding update can be understood from the equation given below:

$$h_u^{(k)} = \sigma\left(W_{\textsf{self}}^{(k)} h_u^{(k-1)} + W_{\textsf{neigh}}^{(k)} \sum_{v \in N_u} h_v^{(k-1)} + b^{(k)}\right)$$

- $h_u^{(k-1)} \in \mathbb{R}^{d^{(k-1)}}$: Node embeddings of target node u at layer k. $W_{\textsf{self}}^{(k)}, W_{\textsf{neigh}}^{(k)} \in \mathbb{R}^{d^{(k)} \times d^{(k-1)}}$: are the learnable parameters, $b^{(k)} \in \mathbb{R}^{d^{(k)}}$: represents the bias term and σ: represents elementwise non-linearity (e.g., a tanh or ReLU).

Graph of Circuits-Graph Neural Networks (GCX-GNN)

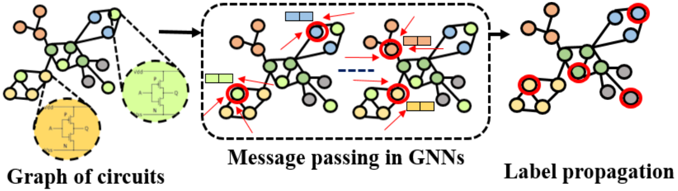


Figure 7: **Message passing in GNNs:** Graph is passed through the GNN architecture; nodes interact with each other via message passing and feature aggregation for target node is carried out. Unlabeled nodes gather information from its labeled neighborhood in form of embeddings. Embeddings updated over L layers is used for label prediction. To better understand GNN architectures, refer: [9, 10, 11].

Graph of Circuits Explorer-Kernalized Graph Regression (GCX-KGR)

- The graph signal at any time instant k is defined by a vector $Y_k = [Y_{k,1}, Y_{k,2}, \ldots, Y_{k,n}]^T$ with $Y_{k,n} \in \mathbb{R}$ being the signal value at node n.
- The graph Laplacian is associated with total variation metric $\nu(Y)$ of a signal Y:

$$\nu(Y) = Y^{\mathrm{T}}\mathbf{L}Y \quad = \sum_{i<l} A_{i,l}\left(Y_i - Y_l\right)^2$$

- The model is estimated in terms of a matrix $\mathbf{W} \in \mathbb{R}^{M\times n}$ such that:

$$Y_n = \mathbf{W}^{\mathrm{T}}\boldsymbol{\phi}\left(X_n\right)$$

- where Y_n is an estimate of the target signal t_n and $\phi : \mathbb{R}^M \to \mathbb{R}^M$ is an unknown function of the input signal.

Graph of Circuits Explorer-Kernalized Graph Regression (GCX-KGR)

- The optimal parameter matrix $\mathbf{W}$ is obtained by minimizing the cost function:

$$C(\mathbf{W}) = \sum_{n=1}^{N} \|t_n - Y_n\|_2^2 + \alpha\,\mathrm{tr}\left(\mathbf{W}^{\mathrm{T}}\mathbf{W}\right) + \beta \sum_{n=1}^{N} \nu\left(Y_n\right)$$

- This formulation works in a supervised setting; we addressed the issues with generating large labeled datasets hence convert it into a semi-supervised learning framework.

Graph of Circuits Explorer-Kernalized Graph Regression (GCX-KGR)

- Thus, given a set of l labeled circuit instances $\{(X_i, Y_i)\}_{i=1}^{l}$ and a set of u unlabeled circuit instances $\{(X_j,)\}_{j=l+1}^{l+u}$ the new formulation becomes:

$$C(\mathbf{W}) = \sum_{n=1}^{l} \|t_n - Y_n\|_2^2 + \gamma_A \operatorname{tr}\left(\mathbf{W}^{\mathrm{T}}\mathbf{W}\right) + \frac{\gamma_I}{(l+u)^2} \sum_{n=1}^{N} \nu\left(Y_n\right)$$

- where γ_A controls the complexity of the function in the input feature space and γ_I controls the complexity of the function in the transformed space (graph structure) [12, 13].

Graph of Circuits Explorer-Kernalized Graph Regression (GCX-KGR)

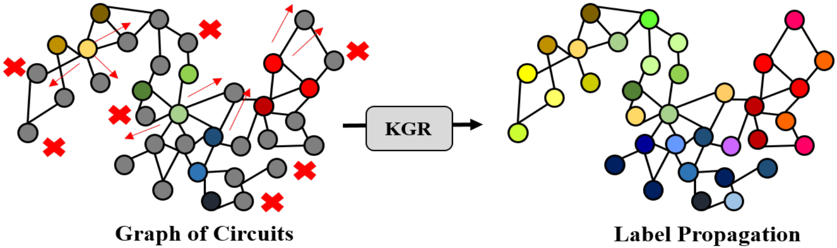


Figure 8: **Kernalized Graph Regression (KGR)**: By considering the geometry of marginal distribution, the model leverages the assumption that points in proximity have similar labels (smoothness assumption); Label propagation to nearby unlabeled samples is based on this principle.

Efficient Analog Sizing via Constrained Optimization (EASCO)

Static Optimization framework

- **Data generation phase**: n points are uniformly sampled from a d-dimensional feature space such that $X_i \in [a_i, b_i]$.
- This creates a dataset (D) such that $D \in \mathbb{R}^{n\times d}$.
- **Graph Learning**: (as previously discussed) using all the n sampled points, with a subset of these points simulated under different labeled data settings (p), such as 30%, and 50%, by adopting a semi-supervised learning framework.

Efficient Analog Sizing via Constrained Optimization (EASCO)

- The labelled points, $X_{train} \in \mathbb{R}^{p \times d}$ are used for training, while the rest, $X_{test} \in \mathbb{R}^{(n-p) \times d}$ are reserved for testing.
- In the third component, GSMs [9, 10, 11, 12, 13] are used to make predictions.
- **Constrained Optimization**: With constraints in place, the process returns a subset of the original search space $\hat{D} \in \mathbb{R}^{k \times d}$ where $k << n$.
- The objective function is optimized using Differential Evolution (DE) algorithm [14] applied to the resulting parameter space, yielding the most optimal parameters.

Efficient Analog Sizing via Constrained Optimization (EASCO)

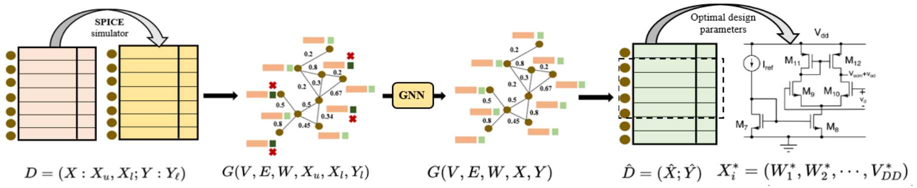

SPICE simulator
GNN
Optimal design parameters
$D = (X : X_u, X_l; Y : Y_\ell)$
$G(V, E, W, X_u, X_l, Y_l)$
$G(V, E, W, X, Y)$
$\hat{D} = (\hat{X}; \hat{Y})$
$X_i^* = (W_1^*, W_2^*, \cdots, V_{DD}^*)$
V_{dd}
I_{ref}
M11
M12
M9
M10
M7
M8

Figure 9: **GCX-EASCO algorithm**: Initially points are uniformly sampled and labels are generated based on the value of p; graph $G\,(V, E, W, X_u, X_l, Y_l)$ is learned using all the points based on features; GNNs in a semi-supervised framework learn the embeddings corresponding to labeled samples, which are used for label prediction corresponding to unlabelled nodes $G\,(V, E, W, X, Y)$. Ultimately, constrained optimization over the subset $\hat{D}$ of the original dataset gives optimum parameters X_i^*.

Analog Sizing through Real-Time Online Graphs (ASTROG)

EASCO efficiently minimized simulations once coarse boundaries were established. However, handling an increasing number of specifications made it challenging to identify these boundaries.

Dynamic Optimization Framework

- Initially, a dataset (D) is generated with α samples such that $D \in \mathbb{R}^{\alpha \times d}$.
- All designs in the database are ranked using the adopted infill sampling criterion [2], and the λ best designs are selected.
- New bounds $\tilde{X}_i = [\tilde{a}_i, \tilde{b}_i]$ are obtained based on these designs, where $\tilde{X}_i \in \mathbb{R}^{\lambda \times d}$.

Analog Sizing through Real-Time Online Graphs (ASTROG)

- The Differential Evolution (DE) algorithm [14] is used to optimize the objective function within the newly obtained bounds.
- Graph is learnt (with the previously discussed approach) by combining the features of best λ labeled points with the m unlabeled points obtained from the DE algorithm.
- The obtained graph is passed through the pre-trained GSMs and labels are predicted for the m unlabeled nodes.

Analog Sizing through Real-Time Online Graphs (ASTROG)

- After the label prediction, the $\lambda + m$ labeled points are re-ranked using the infill sampling criterion [2].
- The optimal design and its corresponding performance metric (label) are then selected from the $\lambda + m$ labeled points and incorporated into the original database.

This process is repeated until the predetermined iteration budget for each circuit is exhausted.

Analog Sizing through Real-Time Online Graphs (ASTROG)

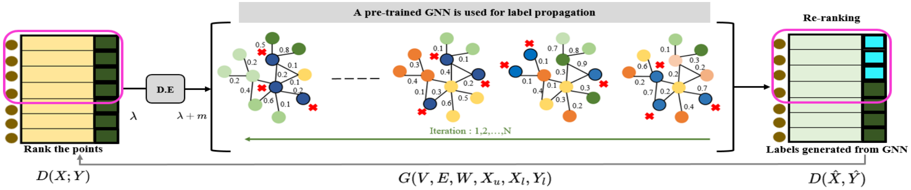


Figure 10: **GCX-ASTROG algorithm**: The process starts with sampling and labeling a small set of points (α). A novel sampling criterion guides the algorithm towards a specific feature space region. The top λ points create a population (P) for the DE algorithm, generating m unlabeled points at every iteration of ASTROG. A semi-supervised learning framework uses $\lambda + m$ points to learn a graph $G\left(V, E, W, X_u, X_l, Y_l\right)$. A pre-trained GNN propagates labels over the m points at each iteration. The best solution is integrated into the original dataset, and the process continues until the iteration budget is met.

Outline

Experiments

- Experimental results of our two proposed algorithms: **EASCO** and **ASTROG**.
- To evaluate the effectiveness of these algorithms, we chose **Two-stage OTA with Miller Compensation** which is operating mostly in a linear region, and the **Ring Oscillator (Three-stage)** which is operating mostly in a nonlinear region as our test cases.
- The Miller Compensated OTA has 11 design variables, while the Ring Oscillator has 6 design variables.
- The performance evaluation of our algorithms was conducted using multiple metrics, which will be discussed in detail later.

Experiments

Test Case-1: Amplifiers

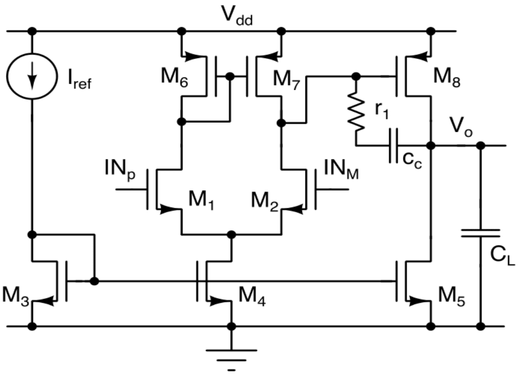
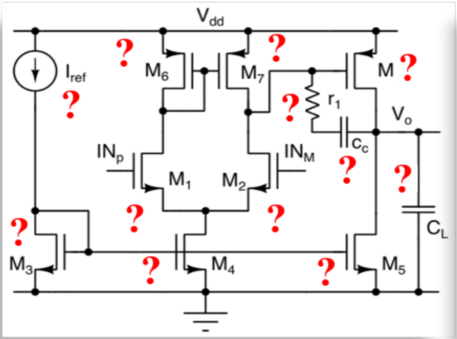
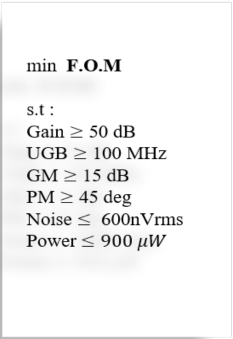


Figure 11: Schematic of Miller Compensated Two-stage OTA

Experiments



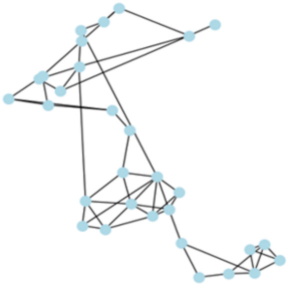

min **F.O.M**

s.t :
Gain ≥ 50 dB
UGB ≥ 100 MHz
GM ≥ 15 dB
PM ≥ 45 deg
Noise ≤ 600nVrms
Power ≤ 900 μW

Figure 12: Parameter-to-Specification (P2S) problem

Experiments

Feature based Graph Learning



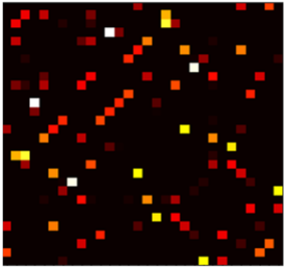


Figure 13: **Sparse Graph Learning**: Feature based Graph of Circuits (n= **30 nodes** for representation) and its corresponding heatmap.

Experiments

Homophily Analysis

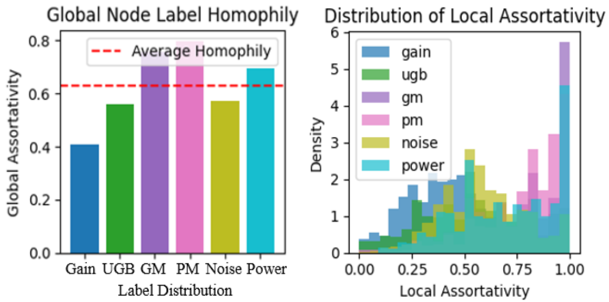
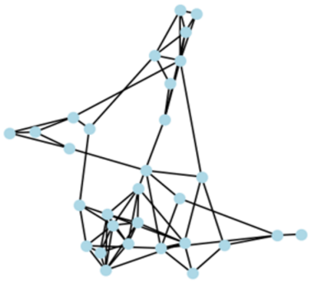


Figure 14: **Global and Local Homophily**: Homophily Analysis across each performance metric. Performed for graph with $n=$ **1k nodes**.

Experiments

Label Informed Graph Learning



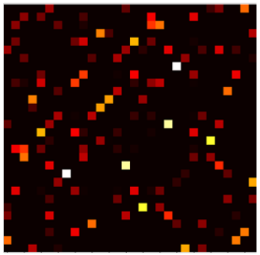


Figure 15: **Label informed Graph Learning**: Features along with label information based Graph of Circuits (n= **30 nodes** for representation) and its corresponding heatmap.

Experiments

Homophily Analysis

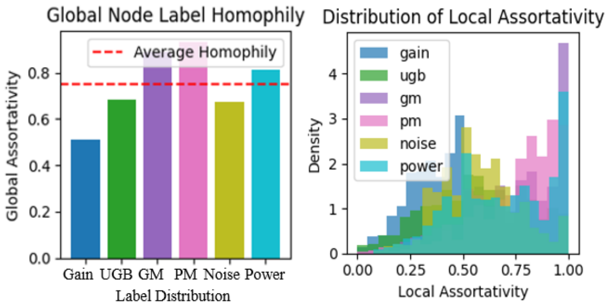


Figure 16: **Global and Local Homophily**: Homophily Analysis across each performance metric. Performed for graph with $n=$ **1k nodes**. Additional information across nodes improves homophily score.

Experiments

Feature based Graph of Circuits			
Test-Circuit	Overall	Global	Local
2-stage Amplifier	0.63	0.62	0.64
3-stage Oscillator	0.61	0.63	0.64

Table 1: Overall, Global and Local node label assortativity associated with circuits under study.

Label informed Graph of Circuits			
Test-Circuit	Overall	Global	Local
2-stage Amplifier	0.75	0.73	0.62
3-stage Oscillator	0.68	0.65	0.60

Table 2: Overall, Global and Local node label assortativity associated with circuits under study.

Experiments

Miller Compensated Two-stage OTA (R^2 scores)												
GCX(.)	Gain **(dB)**		UGB **(MHz)**		GM **(dB)**		PM **(deg)**		Noise **(μV)**		Power **(μW)**	
	p**=30%**	p**=50%**	p**=30%**	p**=50%**	p**=30%**	p**=50%**	p**=30%**	p**=50%**	p**=30%**	p**=50%**	p**=30%**	p**=50%**
(GCN)	0.17	0.22	0.36	0.43	0.52	0.81	0.70	0.81	0.35	0.64	0.42	0.61
(SAGE)	0.42	**0.60**	0.43	**0.60**	0.89	**0.91**	0.93	**0.94**	0.96	**0.97**	0.97	**0.99**
(GAT)	0.19	0.41	0.33	0.41	0.89	0.91	0.79	0.85	0.41	0.61	0.60	0.71

Table 3: R^2 scores with different GNN architectures (Included value is the mean, averaged across 5 runs)

Experiments

Model	Gain	UGB	GM	PM	Noise	Power
GCX(SAGE)-I	0.60	0.60	0.91	0.94	0.97	0.99
GCX(SAGE)-II	0.60	0.58	0.95	0.95	0.99	0.99
ESSAB [2]	**0.57**	**0.50**	**0.81**	**0.93**	**0.89**	**0.89**

Table 4: R^2 scores comparison between our best model **GCX(SAGE)-I,II** and **ESSAB** [2] surrogate model.

Model	p**=30%**	p**=50%**	p**=70%**	p**=90%**
GCX(KGR)-I	**0.20**	**0.61**	**0.65**	**0.71**
GCX(KGR)-II	**0.22**	**0.63**	**0.68**	**0.78**

Table 5: R^2 scores for **GCX(KGR)** averaged across all the performance metrics for different labeled data settings.

Experiments

The question is why did graph-based model, GCX(KGR) perform poorly?

- Traditional graph-based semi-supervised machine learning algorithm
  - Low computational overhead.
  - Global optima.
- **GCX(KGR)** suffers from inaccurate predictions in the given problem due to two main reasons:
  - Limited availability of labeled data.
  - Increasing dimensionality of the circuits.

Experiments

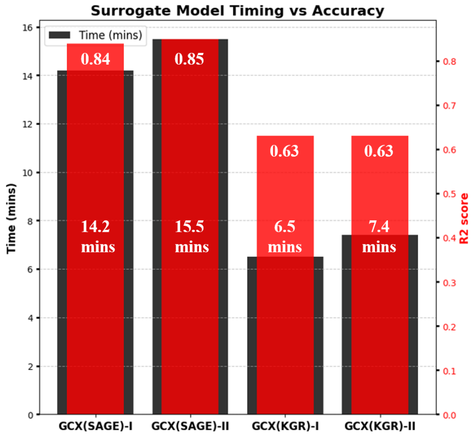


Figure 17: **Optimizing Surrogate Model Selection**: A Training Time vs. Accuracy Plot of Various Models to Identify the Most Suitable Surrogate with the Right Balance of Performance and Efficiency.

Experiments

Miller Compensated Two-stage OTA (Optimization)													
Model	Gain (dB)		UGB (MHz)		GM (dB)		PM (deg)		Noise (nV)		Power (μW)		Success
(specs)	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	(out of 10)
DE [3]	58.6	53.3	190	165	19.2	17.8	46.7	45.0	478	463	815	633	4/10
BO-EI [1]	68.3	65.6	225	190	18.9	17.8	52.0	45.6	470	462	627	505	3/10
EASCO	67.3	64.4	215	192	19.2	18.5	47.2	45.5	470	462	722	490	4/10
ASTROG	80	61.6	145	106	26	23.5	52.5	47.9	497	446	716	318	6/10

Table 6: Comparison table for different optimization strategies [1, 3] with **EASCO** and **ASTROG**

Experiments

Miller Compensated Two-stage OTA (Optimization)							
Model	Gain (dB)	UGB (MHz)	GM (dB)	PM (deg)	Noise (nV)	Power (μW)	FOM
DE [3]	58.6	190	18.35	45.02	478.7	766.3	0.040
BO-EI [1]	**68.3**	**192**	18	**52.8**	**467**	**505.3**	**0.019**
EASCO	67.3	**205**	**18.53**	46.05	**465.5**	548.7	0.022
ASTROG	**80**	112	**24.84**	**52.48**	497.3	**495.5**	**0.021**

Table 7: comparison table for best case performance metrics with corresponding **FOM** (The table displays **simulated results** corresponding to optimized parameters from different algorithms)

Experiments

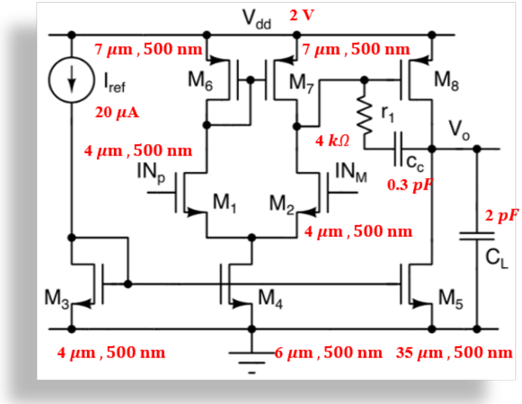


Figure 18: Optimized Input Design parameters with **ASTROG** algorithm.

Experiments

Timing Analysis

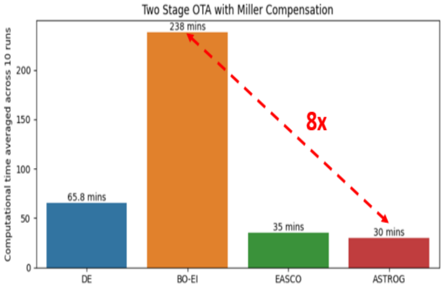


Figure 19: Computational Time comparison corresponding to different optimization algorithms for **Miller Compensated Two-stage OTA**

Experiments

Test Case-2: Oscillators

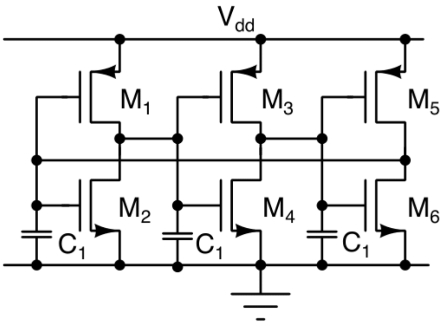
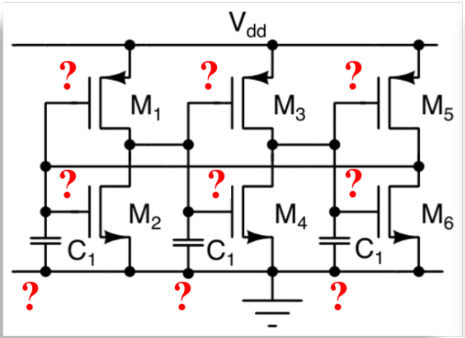
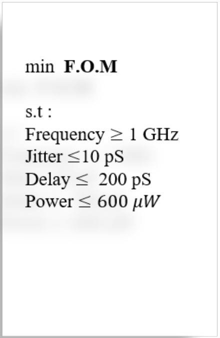


Figure 20: Schematic of Ring Oscillator (Three Stage)

Experiments




min **F.O.M**

s.t :
Frequency $\geq$ 1 GHz
Jitter $\leq$10 pS
Delay $\leq$ 200 pS
Power $\leq$ 600 μW

Figure 21: Parameter-to-Specification (P2S) problem

Experiments

Feature based Graph Learning

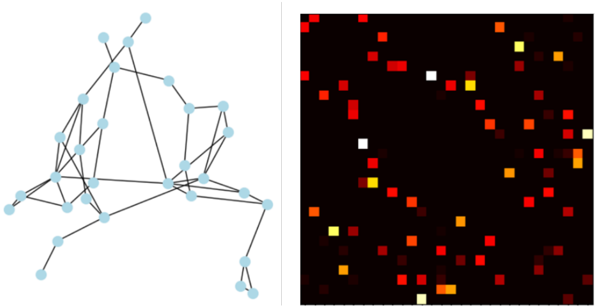


Figure 22: **Sparse Graph Learning**: Feature based Graph of Circuits (n= **30 nodes** for representation) and its corresponding heatmap.

Experiments

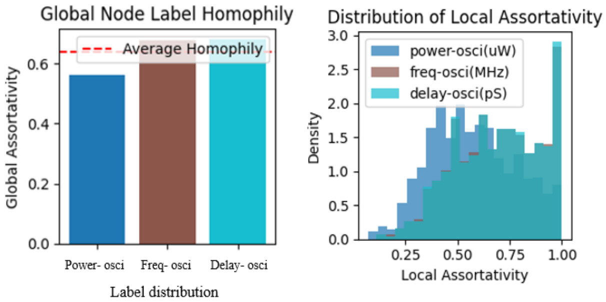


Figure 23: **Global and Local Homophily**: Homophily Analysis across each performance metric. Performed for graph with $n =$ **1k nodes**.

Experiments

Label Informed Graph Learning

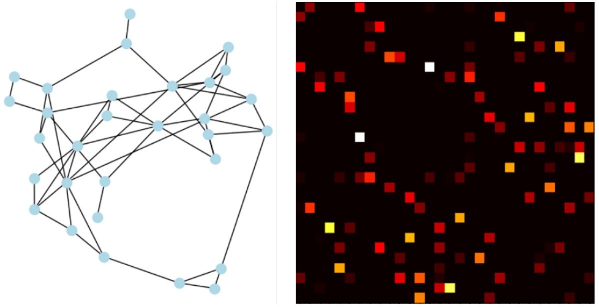


Figure 24: **Label informed Graph Learning**: Features along with label information based Graph of Circuits (n= **30 nodes** for representation) and its corresponding heatmap.

Experiments

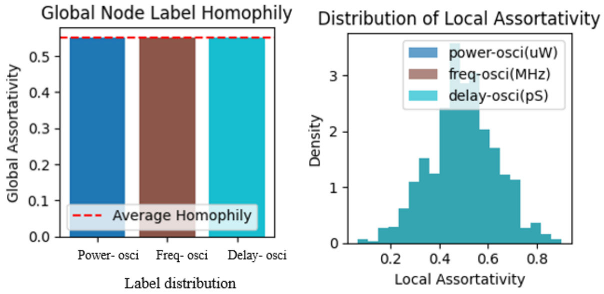


Figure 25: **Global and Local Homophily**: Homophily Analysis across each performance metric. Performed for graph with n= **1k nodes**. Additional information across nodes improves homophily score.

Experiments

Three-stage Ring Oscillator (R^2 scores)						
GCX(.)	Frequency **(MHz)**		Delay **(μS)**		Power **(μW)**	
	p=30%	p=50%	p=30%	p=50%	p=30%	p=50%
(GCN)	0.30	0.44	0.28	0.55	0.35	0.63
(SAGE)	0.45	**0.83**	0.56	**0.82**	0.84	**0.91**
(GAT)	0.41	0.50	0.34	0.58	0.36	0.58

Table 8: R^2 scores with different GNN architectures (Included value is the mean, averaged across 5 runs)

Experiments

Model	Frequency	Delay	Power
GCX(SAGE)-I	0.83	0.82	0.91
GCX(SAGE)-II	0.75	0.90	0.92
ESSAB [2]	0.89	0.71	0.88

Table 9: R^2 scores with different GNN architectures (Included value is the mean, averaged across 5 runs)

Experiments

Ring Oscillator - Three stage (Optimization)					
Model	Frequency (GHz)	rms Jitter (ps)	Delay (ps)	Power (μW)	FOM
DE [3]	1.26	0.99	131.7	**256**	-66.96
BO-EI[1]	1.22	**7e-7**	136.9	**66.4**	**-73.48**
EASCO	**1.44**	**2.2e-7**	**115.4**	335.2	**-67.94**
ASTROG	**1.74**	1.2	**95.7**	502.3	-67.27

Table 10: Comparison table for best case performance metrics with corresponding **FOM**

Experiments

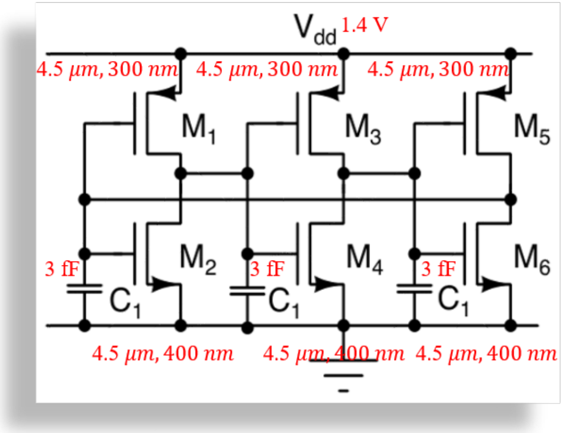


Figure 26: Optimized Input Design Parameters with **ASTROG** algorithm.

Experiments

Timing Analysis

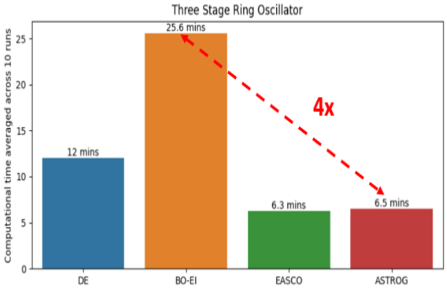


Figure 27: Computational Time comparison corresponding to different optimization algorithms for **Three-stage Ring Oscillator**

Outline

Conclusion

- This work proposed Graph of Circuits Explorer (GCX), incorporating feature and label based information.
- Integration of GCX with Graph based surrogate models (GSMs) in a semi-supervised learning framework reduced reliance on labeled data.
- We additionally introduced **EASCO** and **ASTROG** which yielded most optimal parameters.

Outline

Knowledge Transfer Across Technology and Topology

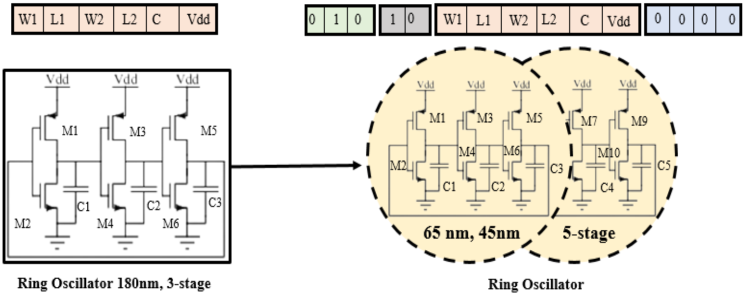


Figure 28: **Knowledge Transfer**: To enhance the feature vector, we append it with one-hot encoded technology files and circuit topology details, resulting in a new feature vector. The technology files used, 180nm, 65nm, and 45nm, are represented using 3 bits. To represent the original and higher-order topology, 2 bits are used, with an additional k bits assigned for circuit element features. The first figure shows the ring oscillator implemented with the 65nm technology file, while the second figure displays the 5-stage topology of the circuit.

Outline

[1] Shady A. Abdelaal, Ahmed Hussein, and Hassan Mostafa. A bayesian optimization framework for analog circuits optimization. In *2020 15th International Conference on Computer Engineering and Systems (ICCES)*, pages 1–4, 2020.

[2] Ahmet Faruk Budak et al. An efficient analog circuit sizing method based on machine learning assisted global optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(5):1209–1221, 2022.

[3] Samrat L. Sabat, K.Shravan Kumar, and Siba K. Udgata. Differential evolution and swarm intelligence techniques for analog circuit synthesis. In *2009 World Congress on Nature Biologically Inspired Computing (NaBIC)*, pages 469–474, 2009.

[4] Haoxing Ren, George F. Kokai, Walker J. Turner, and Ting-Sheng Ku. Paragraph: Layout parasitics and device parameter prediction using graph neural networks. In *Proceedings of the 57th ACM/EDAC/IEEE Design Automation Conference*, DAC '20. IEEE Press, 2020.

[5] Kourosh Hakhamaneshi, Marcel Nassar, Mariano Phielipp, Pieter Abbeel, and Vladimir Stojanović. Pretraining graph neural networks for few-shot analog circuit modeling and design, 2022.

[6] Vassilis Kalofolias. How to learn a graph from smooth signals, 2016.

[7] Kaixiong Zhou, Xiao Huang, Daochen Zha, Rui Chen, Li Li, Soo-Hyun Choi, and Xia Hu. Dirichlet energy constrained learning for deep graph neural networks. *Advances in Neural Information Processing Systems*, 34:21834–21846, 2021.

[8] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs, 2020.

[9] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2016.

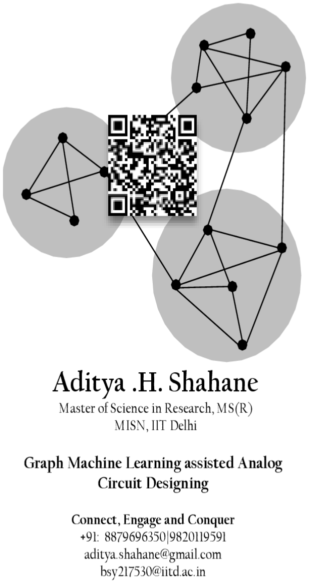
[10] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2017.

[11] Petar Veli$\underline{v}$cković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2017.

[12] Vitor Rosa Meireles Elias, Vinay Chakravarthi Gogineni, Wallace A. Martins, and Stefan Werner. Kernel regression over graphs using random fourier features. *IEEE Transactions on Signal Processing*, 70:936–949, 2022.

[13] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7(85):2399–2434, 2006.

[14] Manolis Georgioudakis and Vagelis Plevris. "a comparative study of differential evolution variants in constrained structural optimization.", (2020) Accessed November 21, 2022.



Aditya .H. Shahane

Master of Science in Research, MS(R)
MISN, IIT Delhi

Graph Machine Learning assisted Analog Circuit Designing

Connect, Engage and Conquer
+91: 8879696350|9820119591
aditya.shahane@gmail.com
bsy217530@iitd.ac.in

Figure 29: Thank you for your patience

Questions?