A Circuit Attention Network-Based Actor-Critic Learning Approach to Robust Analog Transistor Sizing

Yaguang Li*, Yishuang Lin†, Meghna Madhusudan‡, Arvind Sharma‡, Sachin Sapatnekar‡, Ramesh Harjani‡, Jiang Hu*†
*Dept. of Electrical and Computer Engineering, Texas A&M University
†Dept. of Computer Science and Engineering, Texas A&M University
‡Dept. of Electrical and Computer Engineering, University of Minnesota

***Abstract*—Analog integrated circuit design is highly complex and its automation is a long-standing challenge. We present a reinforcement learning approach to automatic transistor sizing, a key step in determining analog circuit performance. A circuit attention network technique is developed to capture the impact of transistor sizing on circuit performance in an actor-critic learning framework. Our approach also includes a stochastic technique for addressing layout effect, another important factor affecting performance. Compared to Bayesian optimization (BO) and Graph Convolutional Network-based reinforcement learning (GCN-RL), two state-of-the-art methods, the proposed approach significantly improves robustness against layout uncertainty while achieving better post-layout performance. BO and GCN-RL can be enhanced with our stochastic technique to reach solution quality similar to ours, but still suffer from a much slower convergence rate. Moreover, the knowledge transfer in our approach is more effective than that in GCN-RL.**

I. Introduction

Analog Integrated Circuits (ICs) are essential components in many electronic systems, including wireless communication, biomedical instruments and sensing systems. Their designs are intrinsically complex and hence require highly specialized expertise as well as tremendous effort. Unlike digital ICs, whose designs are largely automated, analog design automation has long been an unsolved challenge despite decades of research endeavor. Many conventional techniques failed as they lack an efficient means for capturing complex circuit characteristics. By contrast, machine learning is capable of knowledge extraction and reuse. Therefore, it provides an alternative angle and potential for overcoming the challenge. This work is to make use of reinforcement learning for automatic transistor sizing, which is a key design step in determining analog circuit performance.

An early work on analog transistor sizing [1] is a rule-based approach, whose performance control is very coarse-grained. A posynomial performance model is developed in [2] for guiding analog sizing. However, analog circuit behavior is so complex that a posynomial model is often inadequate. An evolutionary algorithm-based sizing method was proposed in [3]. A recently popular approach is Bayesian Optimization (BO) [4], where circuit performance is described by a Gaussian process and simulation-based sampling is required. Later, reinforcement learning approaches are explored [5], [6]. Their agent kernel is deep neural network, which is not as effective as graph neural network [7] as the underlying structure of an analog circuit is a graph. A state-of-the-art approach is GCN (Graph Convolutional Network)-based reinforcement learning method [8].

Many previous works [1]–[5], [8] mean to perform sizing for schematic designs prior to layout. However, layout has a large impact to circuit performance as illustrated by Fig. 1 for a 5T-OTA (5-Transistor Operational Transconductance Amplifier) design. The two crosses indicate the gain and phase margin (PM) of two sizing solutions A and B. After different layout designs of A (B), red square (blue triangle) solutions are obtained. The layout effect is not only large but also has considerable variations, which are due to different designers in manual designs and different tool parameter settings in automatic designs. When post-layout circuit performance fails to meet specifications, the circuit needs to be resized. It is common that designers struggle with multiple sizing-layout iterations, which cost a huge amount of time and effort.

Some previous works incorporate layout, parasitic extraction and post-layout simulation with transistor sizing [6], [9], [10] to address layout effect. However, both layout design and parasitic extraction can be very time consuming. Moreover, post-layout circuit simulation takes much longer time than schematic simulation due to drastic increase of circuit elements after parasitic extraction. Even a simplified realization of this process [10] can greatly exacerbate computation cost for transistor sizing. A template-based parasitic estimation technique is proposed in [11]. However, a template is applicable to limited cases and difficult to be generalized. Recently, machine learning-based parasitic prediction techniques are developed [12], [13]. Although such prediction can avoid layout design cost, it does not handle layout uncertainty as the prediction is targeted to an average effect.

Our analog transistor sizing approach is built upon an actor-critic learning framework – DDPG (Deep Deterministic Policy Gradient) [14]. This work makes the following contributions.

1) We develop Circuit Attention Network (CAN), which is a customized graph neural network and serves as the actor and critic to capture the impact of sizing on performance in DDPG. This distinguishes itself from other reinforcement learning-based sizing methods [5], [6], [8], where off-the-shelf (graph) neural network models are adopted. Simulation results confirm the effectiveness of this customized network. Additionally, CAN allows knowledge transfer among different topologies of the same type of circuits and the transfer is more effective than the GCN-based technique [8].
2) We propose a stochastic technique to mitigate layout effect during transistor sizing from an angle different from previous works. It attempts to make a sizing solution robust against layout uncertainty, e.g., it would choose solution B rather than A in Fig. 1. As such, it can avoid expensive layout and extraction, which are employed in most previous works. It is also be applied as a complement to the parasitic prediction-based approach [12], [13].

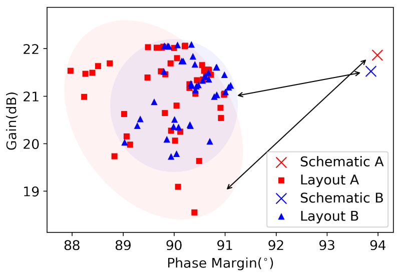

Fig. 1: Layout induced performance degradation and variations for 5T-OTA.

3) The proposed approach is validated through post-layout simulations on several designs of 12nm technology. The results show significant improvement on circuit performance and robustness against layout uncertainty compared to Bayesian optimization (BO) [4] and GCN-based reinforcement learning (GCN-RL) [8], two state-of-the-art methods. On average, CAN-RL has 10.6% circuit performance improvement over GCN-RL for default layouts. BO and GCN-RL can be enhanced with our stochastic technique to achieve solution quality similar to ours, but still suffer from a much slower convergence rate. When all the three methods apply the stochastic technique, our approach is 3.1× faster than GCN-RL and 5.6× faster than BO.

The rest of this paper is organized as follows. The analog transistor sizing formulation is provided in Section II. Our approach is described in Section III. Experimental results are shown in Section IV. Conclusions and future research are discussed in Section V.

II. Problem Formulation

Consider an analog IC, which contains N transistors. The sizes of these transistors are denoted by a vector of positive integer variables $\boldsymbol{x} = [x_1, x_2, \ldots, x_N] \in \mathbb{N}^N$. Please note that analog transistor sizes can be either discrete or continuous and we focus on discrete sizing in this work. Usually, a circuit is evaluated by multiple **performance metrics**, such as gain, bandwidth, phase margin, etc, which are denoted by $\boldsymbol{f} = [f_1, f_2, \ldots, f_M] \in \mathbb{R}^M$ and depend on the values of $\boldsymbol{x}$. Each performance metric f_j has a **specification** or design target ϕ_j. Then, the metrics can be partitioned into two sets: Π^+ (Π^-) is the set of metrics preferred to be greater (less) than ϕ_j, such as gain and bandwidth (delay and offset). To make results of different metrics comparable, **normalized performance metrics** are defined as

$$q_j = \begin{cases} \min(\frac{f_j}{\phi_j}, 1), & \text{for } f_j \in \Pi^+ \\ \min(\frac{\phi_i}{f_j}, 1), & \text{for } f_j \in \Pi^- \end{cases}, j = 1, 2, \ldots, M \tag{1}$$

To assess the overall performance of a circuit, a **Figure Of Merit** $FOM = \sum_{j=1}^{M} w_j q_j$ is evaluated, where w_j indicates weighting factors and $\sum_{j=1}^{M} w_j = 1$. Performance optimization means to maximize the FOM. The FOM definition here is similar to those in previous works [5], [8].

In this work, we propose a new figure of merit as an important ingredient for our stochastic technique to address layout effect in transistor sizing. Unlike previous works, where the performance of a sizing solution is regarded deterministic, we treat normalized metric q_j as a random variable to account for layout uncertainty after sizing. The new figure of merit additionally considers robustness against layout uncertainty and is defined as

$$FOM_R = \sum_{j=1}^{M} w_j \left(\mu(q_j) - \beta \cdot \sigma(q_j)\right) \tag{2}$$

where μ and σ indicate mean and standard deviation, respectively, and β is a weighting factor. The transistor sizing problem we will solve is formulated as

$$\begin{aligned} &\max_{\boldsymbol{x}} \quad FOM_R(\boldsymbol{x}) \\ s.t. \quad & x_L \le x_i \le x_U, i = 1, 2, \ldots, N \\ & x_i \in \mathbb{N}, i = 1, 2, \ldots, N \end{aligned} \tag{3}$$

where x_L and x_U are lower and upper bounds for transistor sizes, respectively.

The first term $\mu(q_j)$ in the objective function plays the same role as the conventional FOM. By maximizing the second term $-\beta \cdot \sigma(q_j)$, the sizing solution with small performance variance induced by layout is preferred. As such, solution B in Fig. 1 would be selected over A according to this formulation. It is evident that the worst case post-layout performance from B would be superior to that from A. Please note that FOM_R is for guiding solutions search in our approach while FOM can still be used to assess a specific circuit either schematic or layout.

III. Robust Transistor Sizing Method

A. Overview

The robust transistor sizing problem (3) is solved through DDPG (Deep Deterministic Policy Gradient) [14] framework as this is one of the most advanced reinforcement learning methods. The **states** are defined by transistor sizes $\boldsymbol{x}$ in their legal range $[x_L, x_U]$. The **actions** $\boldsymbol{a} = [a_1, a_2, \ldots, a_N] \in \{-1, 0, +1\}^N$ are transistor size changes by -1, 0 or +1. However, multiple transistors can change their sizes simultaneously in one time step. The raw output values from DDPG actor are continuous and we round them to obtain integer actions. The **reward** is FOM_R defined by (2). An overview of our approach is depicted in Figure 2. After an action is taken according to the actor, the circuit is modified with corresponding transistor size changes. Then, RC insertion and Monte Carlo circuit simulation are performed to compute reward FOM_R, and these two steps form the kernel of our stochastic technique addressing layout effect. According to the reward, value function $Q(\boldsymbol{x}, \boldsymbol{a})$ is updated for the critic network. Then, the policy is revised for the actor network. The actor/critic network is a dedicate design, called Circuit Attention Network (CAN), which is different from deep neural network in DDPG. The actor/critic network update process, including sampling replay buffer and updating target networks, follows the DDPG framework [14].

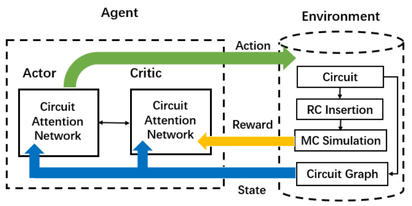

Fig. 2: Robust transistor sizing flow.

B. Reward Computation

Reward computation is to estimate FOM_R, which considers robustness against layout uncertainty. It is the kernel of our stochastic technique for considering layout effect during sizing. Its key advantage is that the expensive process of layout, extraction and post-layout simulation can be avoided. Please note the Monte Carlo circuit simulation is before extraction and thus has much less elements than post-layout simulation, although it considers layout effect.

Layout effect is mainly due to RC (Resistance and Capacitance) parasitic from metal wires generated during layout design. Given a sizing solution, we insert RC elements for each net to emulate layout effect and its variations. The insertion is applied through star models. That is, a conceptual center node is generated for each net and an edge is inserted between the center and each pin of this net. For example, D is the center node for the net connecting pins A, B and C in Fig. 3. For each edge in a star model, an RC element is inserted as shown in the right part of Fig. 3. The R and C for the same edge are correlated following the same random wire-length of this edge. The random RC values are assumed to follow Gaussian distribution, and their mean/variance are obtained from historical design data.

For a sizing solution, K sets of RC elements are separately inserted and K-run Monte Carlo (MC) circuit simulations are performed, where each run corresponds to one set of random RC elements. A circuit simulator, such as SPICE, usually has embedded MC support, which executes much faster than separated MC runs. Even

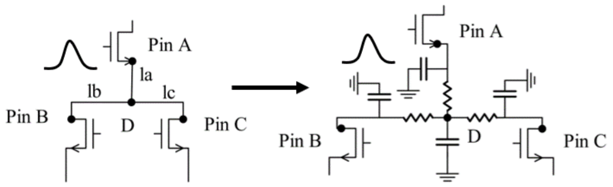

Fig. 3: RC insertion via star model in schematic.

after RC elements are inserted, circuit simulations at schematics are still much faster than post-layout simulations. After MC simulations, $\mu(q_j), \sigma(q_j), j = 1, 2, \ldots, M$ are obtained and the reward FOM_R can be calculated according to Equation (2).

C. *Circuit Attention Network for Actor and Critic*

We develop Circuit Attention Network (CAN), a customized Graph Neural Network (GNN), to serve as actor and critic networks in reinforcement learning. This is similar to [7], which is also a customized GNN for circuit design. However, CAN architecture is different from [7], which is for performance prediction in analog placement.

1) Circuit Graph and Features for CAN: A circuit is abstracted to a graph and associated features $\mathcal{G}(\mathcal{V}, \mathcal{E}, \boldsymbol{X}, \boldsymbol{E})$ to serve as input to CAN. Devices (including transistors), IO pins and net centers constitute the set of nodes $\mathcal{V}$. Edges $\mathcal{E}$ indicate connections between nodes $v \in \mathcal{V}$ and are described by an adjacency matrix $\boldsymbol{A}$. Node feature matrix and edge feature matrix are denoted by $\boldsymbol{X}$ and $\boldsymbol{E}$, respectively. Figure 4 shows an example of representing a current mirror as a graph.

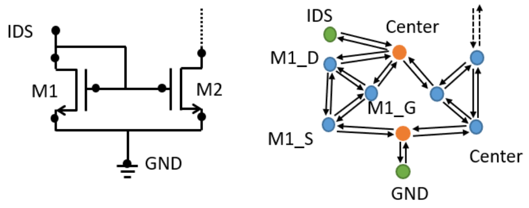

Fig. 4: A current mirror and its graph.

Feature vector for the i-th ($i = 1, 2, ..., n$) node, $\boldsymbol{X}_i \in \mathbb{R}^d$ consists of the following components:

- Node type: PMOS, NMOS, capacitor, current source, GND, net center point, etc., represented by a one-hot encoded vector.
- A scalar feature for functional module that a node belongs to, such as current mirror, differential pair and active load.
- Sizes for nodes representing transistors.

The feature $\boldsymbol{E}_{ij} \in \mathbb{R}^p$ for an edge from node j to node i is a one-hot encoded vector. The encoding is based on the types of node i and node j, such as transistor source, drain, gate, net center, etc. Except transistor sizes, the other features are constant for a specific circuit. Nevertheless, they are important for knowledge transfer among different topologies of the same type of circuits.

2) CAN Architecture: Like most GNNs, CAN is composed by a series of feature embedding layers followed by MLP (Multi-Layer Perceptron). A feature embedding layer can be described by

$$\boldsymbol{X}^{(l+1)} = \Phi_{embed}(\boldsymbol{X}^{(l)}) \tag{4}$$

where l is the layer index and $\boldsymbol{X}^{(0)}$ is the original feature tensor $\boldsymbol{X}$. Function Φ_{embed} indicates the embedding operation, which is to be elaborated later.

The work of [8] uses an off-the-shelf approach of GCN [15], where a feature embedding layer computes a uniform convolution (aggregation) of each node feature with its neighbors. However, analog circuit behavior is so complex that such uniform convolution is inadequate. Like [7], CAN adopts attention-based convolution [16] in feature embedding and simultaneously performs graph pooling to capture a global view. The application scenario of [7] is analog placement, where distances among transistor are known. By contrast, our work is prior to layout and has significantly less information than [7]. Therefore, we devise enhancements to overcome the difficulty.

In the l-th layer of feature embedding, node features become $\boldsymbol{X}_i^{(l)}, i = 1, 2, \ldots, n$ and edge features are transformed to $\boldsymbol{E}_{ij}^{(l)}, i, j = 1, 2, \ldots, n$ [7]. Each feature embedding layer in CAN consists of three steps, which are elaborated as follows.

Step 1: Attention construction and compression. Attention [16] can be treated as weights with trainable parameters for graph convolution. In the l-th layer of feature embedding, the k-th channel of the raw **attention coefficient** between nodes i and j is defined as

$$\hat{\alpha}_{ijk}^{(l)} = \tau_{ij} E_{ijk}^{(l)} \tag{5}$$

where τ_{ij} is a function of $\boldsymbol{X}_i^{(l)}$ and $\boldsymbol{X}_j^{(l)}$, and $E_{ijk}^{(l)}$ is the k-th channel of edge feature $\boldsymbol{E}_{ij}^{(l)}$. Then, bidirection normalization is performed to obtain 3D attention, which is defined as

$$\boldsymbol{\alpha}^{(l)} = \begin{cases} \tilde{\alpha}_{ijk}^{(l)} = \text{softmax}_{\text{row}}(\hat{\alpha}_{ijk}^{(l)}) \\ \alpha_{ijk}^{(l)} = \displaystyle\sum_{m=1}^{n_l} \frac{\tilde{\alpha}_{imk}^{(l)} \tilde{\alpha}_{jmk}^{(l)}}{\sum_{u=1}^{n_l} \tilde{\alpha}_{umk}^{(l)}} \end{cases} \tag{6}$$

where $\tilde{\alpha}_{ijk}^{(l)}$ is the k-th channel of the row-normalized raw attention coefficient of layer l, and n_l is the number of nodes in layer l. Since graph pooling is performed at each layer, the number of nodes may decrease from one layer to the next layer. Next, the attention is compressed from 3D to 2D as

$$e_{ij}^{(l)} = g(\boldsymbol{\alpha}_{ij}^{(l)}; \boldsymbol{b}^{(l)}) = \sum_{k=1}^{p_l} \alpha_{ijk}^{(l)} b_k^{(l)} \tag{7}$$

where $\boldsymbol{b}^{(l)} \in \mathbb{R}^{p_l}$ is a trainable vector and p_l is the number of edge feature channels in layer l.

Our main contribution is a new treatment of τ_{ij} for analog transistor sizing. Its key idea is to separate the common and differential parts of node features during attention construction as they affect circuit performance differently in general. For example, consider a workload pair and a differential pair. Their size difference affects the gain of an amplifier in a different way from their common part or average size. We compute the mean value of node features $\bar{\boldsymbol{X}}^{(l)}$ and intermediate node features $\tilde{\boldsymbol{X}}_i^{(l)}$ are calculated as:

$$\begin{aligned} \bar{\boldsymbol{X}}^{(l)} &= \frac{1}{n_l} \sum_{k=1}^{n_l} \boldsymbol{X}_k^{(l)} \\ \boldsymbol{X}_{d,i}^{(l)} &= \boldsymbol{X}_i^{(l)} - \bar{\boldsymbol{X}}^{(l)} \\ \tilde{\boldsymbol{X}}_i^{(l)} &= \boldsymbol{X}_{d,i}^{(l)} + \boldsymbol{X}_{d,i}^{(l)2} \end{aligned} \tag{8}$$

where d indicates differential term. Symbol '+/-' in (8) means element-wise vector addition/subtraction. It should be noted that $\boldsymbol{X}_i^{(l)}$, $\tilde{\boldsymbol{X}}_j^{(l)}$, $\boldsymbol{X}_{d,i}^{(l)}$ and $\boldsymbol{X}_{d,i}^{(l)2}$ are vectors with the same size. This step divides node features $\boldsymbol{X}^{(l)}$ into the mean value $\bar{\boldsymbol{X}}^{(l)}$, which serves as the common mode term, and $\boldsymbol{X}_{d,i}^{(l)}$, which acts as the differential term. Furthermore, the differential term is appended with its square to account for higher order effect, which often exists in analog circuits. Finally, τ_{ij} is the sum of differential mode term $\tau_{d,ij}$ and common mode term $\tau_{c,i}$ as

$$\begin{aligned} \tau_{d,ij} &= \text{LeakyReLU}\left(\boldsymbol{a}_d^{(l)} \cdot \left[\left(\boldsymbol{W}^{(l)\mathrm{T}} \boldsymbol{X}_i^{(l)}\right) || (\boldsymbol{W}^{(l)\mathrm{T}} \tilde{\boldsymbol{X}}_j^{(l)}\right)\right]\right) \\ \tau_{c,i} &= \text{LeakyReLU}\left(\boldsymbol{a}_c^{(l)} \cdot \left[\left(\boldsymbol{W}^{(l)\mathrm{T}} \boldsymbol{X}_i^{(l)}\right) || (\boldsymbol{W}^{(l)\mathrm{T}} \bar{\boldsymbol{X}}^{(l)}\right)\right]\right) \\ \tau_{ij} &= \tau_{d,ij} + \tau_{c,i} \end{aligned} \tag{9}$$

where $\boldsymbol{W}^{(l)}$ is a trainable weight matrix, $\boldsymbol{a}_d^{(l)}$ and $\boldsymbol{a}_c^{(l)}$ are trainable vectors, and $||$ means vector concatenation. Compared with PEA [7], this new attention scheme benefits CAN with up to 2.6X convergence speed up, which is demonstrated in Table VI.

Step 2: Graph convolution. The graph convolution is performed as

$$\boldsymbol{Z}^{(l)} = \Gamma\left(g(\boldsymbol{\alpha}^{(l)}; \boldsymbol{b}^{(l)})\boldsymbol{X}^{(l)}\boldsymbol{W}^{(l)}\right) \tag{10}$$

where $\Gamma(\cdot)$ is an activation function. An intermediate node embedding $\boldsymbol{Z}^{(l)}$ is aggregated from node feature $\boldsymbol{X}^{(l)}$ with a 2D attention matrix $g(\boldsymbol{\alpha}^{(l)}; \boldsymbol{b}^{(l)})$ defined by (7).

Step 3: Node and edge pooling. A trainable assignment matrix $\boldsymbol{S}^{(l)} \in \mathbb{R}^{n_l \times n_{l+1}}$ for layer l is computed with details in [7], [17]. According to $\boldsymbol{S}^{(l)}$, $\boldsymbol{Z}^{(l)}$ is mapped to node feature matrix $\boldsymbol{X}^{(l+1)}$ with reduced dimension. Similarly, adjacency matrix $\boldsymbol{A}^{(l+1)}$ and edge feature tensor $\boldsymbol{E}^{(l+1)}$ for layer $l+1$ are obtained [7], [17].

Node/edge feature matrices in the last feature embedding layer are flattened into a 1D vector and fed to MLP (Multi-Layer Perceptrons), whose output forms actions for the actor or value function for the critic.

3) CAN in DDPG: The robust transistor sizing follows the same procedure of DDPG [14] except that the deep neural networks in DDPG are replaced by CANs. Accordingly, updating neural network weights becomes training $\boldsymbol{W}, \boldsymbol{a}, \boldsymbol{b}$, etc., in CANs. The sizing iterations are not organized in epochs. Hence, each iteration corresponds to one time step, where one action is made.

D. Knowledge Transfer among Different Topologies

In analog IC designs, a function can often be realized by different circuit topology designs. For instance, OTA can be realized as either cascode OTA or current mirror OTA with the same functionality. To facilitate knowledge reuse and thereby improve design efficiency, we wish that a reinforcement learning agent trained for one topology (source topology) can be applied to a different topology (target topology) with incremental training by a few data samples on target topology.

Knowledge transfer among different topologies requires to solve two problems: (1) How to let a single GNN model accommodate different input dimension sizes resulted from different graph structures? (2) How to make a single GNN model provide output of different sizes due to different action spaces? Please note that the critic network has scalar output and does not face the second problem.

In CAN, the first problem is solved in the same way as [7]. That is, the graph pooling in CAN performs node/edge feature dimension reduction and naturally handles input of different dimension sizes. Since the GNN in [7] generates a scalar output for different topologies, it does not need to handle the second problem. We solve the second problem in way similar to [8]. That is, the actor network of target topology inherits the feature embedding layers from source topology, and replaces the MLP of source topology with an untrained MLP. As the bulk of knowledge is stored in feature embedding layers, the incremental training for the new MLP needs very few data samples.

The work of [8] requires an extra step of node feature transformation for the knowledge transfer, while CAN naturally supports knowledge transfer among different topologies without such requirement. Besides, two MLPs are used in [8], one at the input and the other at the output. By contrast, CAN only needs to incrementally train one MLP. Hence, knowledge transfer in CAN is more efficient than [8]. This advantage is confirmed by the experiment results.

IV. Experimental Results

A. Experiment Setup

Analog circuit testcases include three topologies of OTA (Operational Transconductance Amplifier), two topologies of comparator circuits, two topologies of VCO (Voltage-controlled Oscillator) circuits and two topologies of SCF (Switched Capacitor Filter) circuits. The three topologies of OTAs are 5T (5 Transistor)-OTA, CC (Cascode)-OTA and CM (Current Mirror)-OTA. The machine learning models are implemented with TensorFlow in Python. Layouts are generated automatically with an open source analog physical synthesis tool, ALIGN[1]. Post-layout circuit simulations are performed through Cadence Spectre with GlobalFoundries 12nm FinFET technology process. The experiments are conducted on a Linux machine with Xeon (R) E5-2680 V2 processor of 2.8GHz frequency and 256G memory.

The configurations of CAN networks in the experiment are summarized in Table I. For both the actor and critic, CAN has four feature embedding layers. According to our experience, it is better to skip graph pooling in certain layers and this is why the first (last) two layers have the same configuration. The MLPs in the actor and critic have 3 and 5 layers, respectively. The last MLP layer in the critic network has a single output for the value function. The number of neurons in the last layer of the actor MLP is the same as the number of transistors in a circuit.

Feature embedding layers			#neurons in MLPs	
#nodes	#node features	#edge features	actor	critic
				32
12	11	31	32	16
12	11	31	16	8
6	5	12		4
6	5	12	#transistors	1

TABLE I: CAN configuration.

For FinFET technology, transistor size refers to the number of fins per finger. The transistor sizes are in the range of [2, 20] with initial size being 10. For some transistors, their sizes need to be matched, e.g., differential pairs. The size matching is maintained for such transistors during the sizing. In the actor-critic learning, the maximum training step is set as 300. The learning rates for the actor and critic are set as 0.001 and 0.002, respectively. The replay buffer size is 50 for both the actor and critic. The training batch size 32. The reward discount is 0.9. The number of Monte Carlo runs for each reward computation is 100.

B. Circuit Performance and Robustness

In this part, we compare the solution quality of our method **CAN-RL** (with FOM_R reward) and **GCN-RL** [8] (with FOM reward). Bayesian optimization (**BO**)-based (with FOM objective) transistor sizing [4] has also been evaluated. Its solution quality is very similar to GCN-RL and thus its results are omitted here for brevity. The main results are shown in Fig. 5, where each column corresponds to the sizing solution of one method on one testcase. For each sizing solution, we generate hundreds of different layout designs by varying the tool parameters. The color bars and their associated horizontal segments indicate the median quartile, mean, the maximum and the minimum FOM among these layout designs. In addition, FOM values of schematic designs, which are sizing solution prior to layout, and those generated by default layout tool setting are provided as red crosses and green triangles, respectively. Although FOM_R (2) is used as reward within CAN-RL, it means to evaluate a statistical effect for guiding solution search. For a specific design, either schematic or layout, the conventional deterministic FOM is still employed to assess its overall circuit performance. We have the following observations for Fig. 5.

- Layout usually degrades circuit performance and this is especially obvious for the two comparator designs. Although a

[1]https://github.com/ALIGN-analoglayout/ALIGN-public

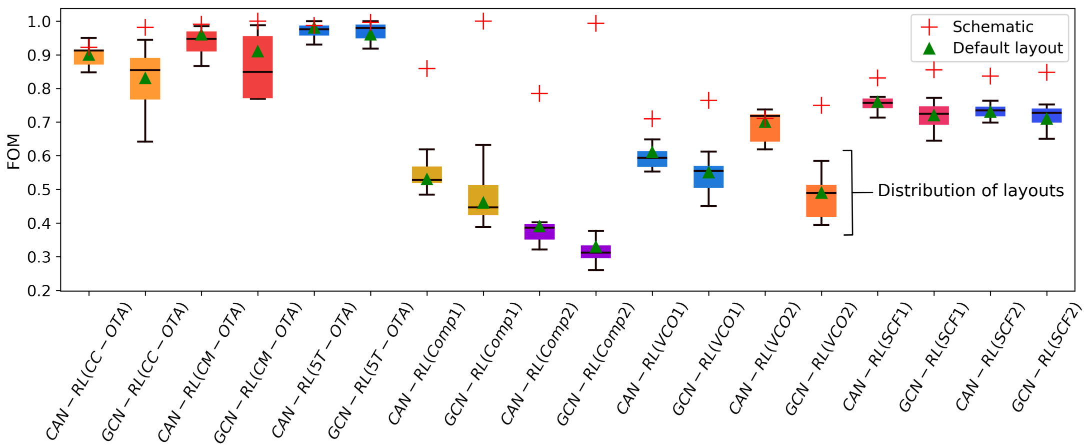

Fig. 5: FOM distributions for different methods. Color bars show ranges of median quartiles, the horizontal segments within color bars indicate mean value and the horizontal segments above and below color bars mean the maximum and the minimum values, respectively.

layout-oblivious approach, such as GCN-RL, can achieve a very good performance for schematic design, its corresponding post-layout performance can be significantly worse than that from a layout-aware sizing.

- For multiple layout designs, our CAN-RL reaches significantly better mean FOM than GCN-RL [8], i.e., our method achieves better circuit performance on average.
- The FOM values from multiple layout designs of our CAN-RL solutions have remarkably smaller variance than those of GCN-RL. Thus, our sizing solutions are much less sensitive to layout uncertainty than GCN-RL.
- When only the default layout setting is used, performance of our sizing solutions is still superior to that of GCN-RL.

Circuit	CC-OTA		CM-OTA		5T-OTA	
Method	CAN-RL	GCN-RL	CAN-RL	GCN-RL	CAN-RL	GCN-RL
Gain (dB)	23.26	24.07	25.14	20.75	21.28	20.62
UGF (MHz)	968.3	821.1	2132	1280	731.3	998.1
BW (MHz)	67.55	48.06	139.2	127	63.24	93.6
PM (degree)	85.54	87.63	50.37	66.2	91.2	89.0
FOM	0.90	0.83	0.96	0.91	0.98	0.96

TABLE II: Post-layout performance of OTAs using default layout setting.

Circuit	Comp1		Comp2	
Method	CAN-RL	GCN-RL	CAN-RL	GCN-RL
Delay 1 (ps)	67.15	267.8	85.28	158.1
Delay 2 (ps)	19.67	2.1	48.02	41.87
Power (mW)	194.9	179.0	303.3	442.3
Offset (mV)	24.91	39.99	24.66	35.23
FOM	0.53	0.46	0.39	0.33

TABLE III: Post-layout performance of Comparators using default layout setting.

Circuit	VCO1		VCO2	
Method	CAN-RL	GCN-RL	CAN-RL	GCN-RL
Min Frequency (GHz)	0.17	0.15	0.14	0.27
Max Frequency (GHz)	1.22	1.53	0.88	0.43
Power (uW)	447.0	636.9	338.8	443.9
FOM	0.61	0.55	0.70	0.49

TABLE IV: Post-layout performance of VCOs using default layout setting.

Besides FOM values, we summarize individual circuit performance characteristics for OTAs, Comparators, VCOs and SCFs in Tables II, III, IV, and V, respectively. The average FOM of default layouts is 0.73 for CAN-RL and 0.66 for GCN-RL. So CAN-RL has 10.6% FOM improvement over GCN-RL. The results show that indeed our CAN-RL leads to performance superior to that of GCN-RL. Since the solution quality of BO is about the same as GCN-RL, the observations for GCN-RL are also applicable to BO.

Circuit	SCF1		SCF2	
Method	CAN-RL	GCN-RL	CAN-RL	GCN-RL
Gain (dB)	16.05	15.02	16.17	15.52
UGF (MHz)	3.29	3.11	3.08	3.01
BW (KHz)	520.9	557.0	479.5	508.4
FOM	0.76	0.72	0.73	0.71

TABLE V: Post-layout performance of SCFs using default layout setting.

C. Convergence and Knowledge Transfer

In this section, we compare the convergence rates of our CAN-RL, GCN-RL [8] and BO (Bayesian Optimization) [4]-based analog transistor sizing. Each method has two variants: one uses layout-oblivious FOM as its objective/reward and the other employs layout-aware FOM_R as its objective/reward. Using FOM_R for BO and GCN-RL can be regarded as an enhancement with our stochastic technique. We also assess the impact of knowledge transfer for CAN-RL and GCN-RL. Table VI summarizes the time to reach 97.5% of the maximum FOM/FOM_R for different methods. We have the following observations.

- Although CAN-RL, GCN-RL and BO can eventually achieve similar solution quality when the objective/reward is the same (one example is shown in Figure 6), CAN-RL has the fastest convergence. Compared to BO, CAN-RL converges 5.6× and 6.8× faster for rewards of FOM_R and FOM, respectively.
- Knowledge transfer facilitates further speedup. For example, when reward function is FOM_R, the transfer makes CAN-RL about 2.8× faster. CAN-RL enjoys higher speedup from knowledge transfer than GCN-RL.
- Compared with using reward FOM, using FOM_R causes about 2.5× slowdown on convergence. However, considering that 100-run Monte Carlo circuit simulations are performed for each FOM_R computation, the 2.5× slowdown implies quite high parallel efficiency in the Monte Carlo simulations.

To have a further intuition on the convergence of different methods, curves of FOM/FOM_R versus iterations are plotted in Figs 6 for Comp1. The upper row is for reward of layout-oblivious FOM and the lower row is for reward of layout-aware FOM_R. For clarity, we separate methods without knowledge transfer and those with comparing transfer to the left and right columns. The solid red curves show that CAN-RL mostly converges faster than GCN-RL and BO. The dashed red curves exhibit further speedup for CAN-RL

Circuit	Reward=FOM							Reward=FOM_R						
	CAN Transfer	CAN No Trnsf	PEA Transfer	PEA No Trnsf	GCN Transfer	GCN No Trnsf	BO	CAN Transfer	CAN No Trnsf	PEA Transfer	PEA No Trnsf	GCN Transfer	GCN No Trnsf	BO
CC-OTA	0.02	0.13	0.04	0.26	0.24	0.26	0.39	0.67	2.09	0.67	3.41	2.94	4.50	6.51
CM-OTA	0.06	0.86	0.14	0.93	0.37	0.77	2.70	2.33	6.73	3.83	6.62	5.32	6.23	10.56
5T-OTA	0.12	0.17	0.14	0.39	0.13	0.37	0.23	0.92	4.51	1.47	4.32	2.86	3.84	4.21
Comp1	1.63	4.07	5.56	6.77	8.48	8.22	13.68	23.74	38.24	46.15	63.29	68.41	89.46	97.45
Comp2	0.33	1.98	1.82	3.30	6.15	6.96	3.27	12.90	59.66	35.47	66.10	62.75	86.88	91.78
VCO1	11.90	44.20	35.70	44.20	37.22	106.58	94.14	46.03	167.06	63.08	173.88	61.23	187.07	190.70
VCO2	13.31	74.82	43.24	81.47	54.34	88.91	75.19	17.33	230.22	103.98	240.12	177.65	310.88	303.77
SCF1	66.62	145.72	174.86	216.49	274.51	270.35	424.47	183.73	197.34	251.78	340.24	353.63	462.45	653.06
SCF2	16.84	33.68	30.87	56.13	106.33	134.31	142.86	45.01	208.14	151.90	230.65	325.81	376.36	511.39
Avg.	12.31	33.96	32.49	45.55	54.20	68.53	84.10	36.96	101.55	73.15	125.40	117.84	169.74	207.71
Speedup	6.8X	2.5X	2.6X	1.8X	1.6X	1.2X	1.0X	5.6X	2.0X	2.8X	1.7X	1.8X	1.2X	1.0X

TABLE VI: Time (minutes) to reach 97.5% of the maximum FOM/FOM_R. Transfer learning includes CC-OTA $\leftrightarrow$ CM-OTA, CC-OTA $\rightarrow$ 5T-OTA, Comp1 $\leftrightarrow$ Comp2, VCO1 $\leftrightarrow$ VCO2 and SCF1 $\leftrightarrow$ SCF2.

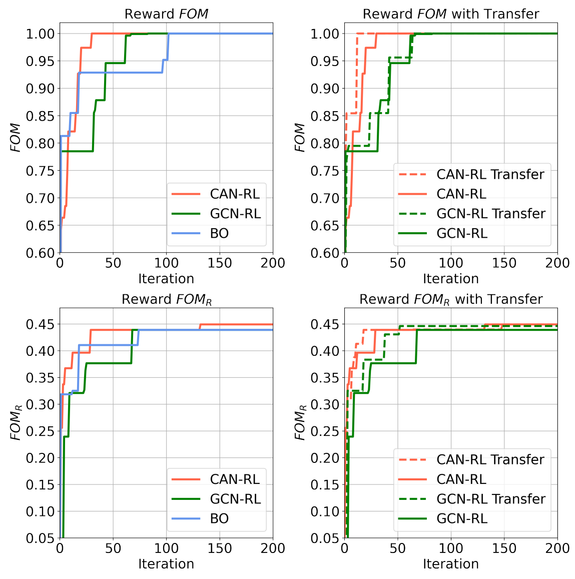

Fig. 6: Learning curves of Comp1.

by knowledge transfer. The convergence rate in terms of iterations is similar between FOM reward and FOM_R reward. Hence, the extra computation cost of FOM_R is within each iteration.

D. Additional Analysis

One may wonder if a single run of layout, extraction and post-layout simulation can be cheaper than the Monte Carlo (MC) pre-layout simulation. Fig. 7 shows such comparison for CC-OTA and Comparator1. It can be observed that the time of parasitic extraction is very long and overall the MC simulation is still significantly faster.

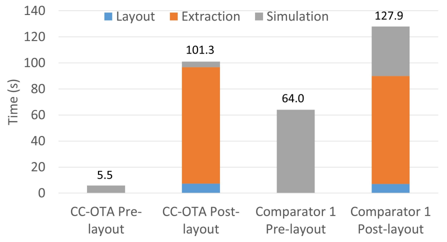

Fig. 7: Runtime comparison between pre-layout MC circuit simulation and a single run of layout, extraction and post-layout simulation.

V. Conclusions and Future Research

In this work, a specialized graph neural network is developed and applied in an actor-critic learning framework for analog transistor sizing. A stochastic technique is developed to address layout effect without actually performing expensive layout, extraction and post-layout simulation. Compared to GCN-based reinforcement learning and Bayesian optimization, our method can significantly improve circuit performance and robustness against layout variations. When GCN-RL and BO adopt our stochastic technique, they can reach solution quality similar to ours but take several times longer computation time. Moreover, our knowledge transfer achieves significantly more speedup than the knowledge transfer of GCN-RL.

In future research, we will integrate machine learning-based parasitic prediction techniques with our stochastic reward function so that the optimization can be more accurate and more efficient.

Acknowledgement

This work is supported by the DARPA ERI IDEA program.

References

[1] H. Graeb *et al.*, “The sizing rules method for analog integrated circuit design,” in *Proc. ICCAD*, 2001, pp. 343–349.
[2] W. Daems *et al.*, “Simulation-based generation of posynomial performance models for the sizing of analog integrated circuits,” *IEEE TCAD*, vol. 22, no. 5, pp. 517–534, 2003.
[3] B. Liu *et al.*, “Analog circuit optimization system based on hybrid evolutionary algorithms,” *Integration*, vol. 42, no. 2, pp. 137–148, 2009.
[4] W. Lyu *et al.*, “An efficient bayesian optimization approach for automated optimization of analog circuits,” *IEEE TCAD*, vol. 65, no. 6, pp. 1954–1967, 2017.
[5] Z. Zhao and L. Zhang, “Deep reinforcement learning for analog circuit sizing,” in *Proc. ISCAS*, 2020, pp. 1–5.
[6] K. Settaluri *et al.*, “AutoCkt: Deep reinforcement learning of analog circuit designs,” in *Proc. DATE*, 2020, pp. 1–6.
[7] Y. Li *et al.*, “A customized graph neural network model for guiding analog IC placement,” in *Proc. ICCAD*, 2020, pp. 1–8.
[8] H. Wang *et al.*, “GCN-RL circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning,” in *Proc. DAC*, 2020, pp. 1–6.
[9] P. Vancorenland *et al.*, “A layout-aware synthesis methodology for RF circuits,” in *Proc. ICCAD*, 2001, pp. 358–362.
[10] N. Lourenço *et al.*, “Layout-aware sizing of analog ICs using floorplan & routing estimates for parasitic extraction,” in *Proc. DATE*, 2015, pp. 1156–1161.
[11] M. Ranjan *et al.*, “Fast, layout-inclusive analog circuit synthesis using pre-compiled parasitic-aware symbolic performance models,” in *Proc. DATE*, 2004, pp. 604–609.
[12] B. Shook *et al.*, “MLParest: Machine learning based parasitic estimation for custom circuit design,” in *Proc. DAC*, 2020, pp. 1–6.
[13] H. Ren *et al.*, “ParaGraph: Layout parasitics and device parameter prediction using graph neural networks,” in *Proc. DAC*, 2020, pp. 1–6.
[14] T. P. Lillicrap *et al.*, “Continuous control with deep reinforcement learning,” *arXiv preprint arXiv:1509.02971*, 2019.
[15] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” *arXiv preprint arXiv:1609.02907*, 2016.
[16] P. Veličković *et al.*, “Graph attention networks,” in *Proc. ICLR*, 2018, pp. 1–12.
[17] R. Ying *et al.*, “Hierarchical graph representation learning with differentiable pooling,” in *Proc. NIPS*, 2018, pp. 4805–4815.