Topology Optimization of Operational Amplifier in Continuous Space via Graph Embedding

Jialin Lu[1], Liangbo Lei[1], Fan Yang[1*], Li Shang[2], Xuan Zeng[1*]
[1]State Key Lab of ASIC & System, School of Microelectronics, Fudan University, Shanghai, China
[2]China and Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University, Shanghai, China

***Abstract*—Operational amplifier is a key building block in analog circuits. However, the design process of the operational amplifier is complex and time-consuming, as there are no practical automation tools available in the industry. This paper presents a new topology optimization method for operational amplifiers. The behavioral description of the operational amplifier is described using a directed acyclic graph (DAG), which is then transformed into a low-dimensional embedding in continuous space using a variational graph autoencoder. Topology search is performed in the continuous embedding space using stochastic optimization methods, such as Bayesian Optimization. The yield search results are then transformed back to operational amplifier topologies using a graph decoder. The proposed method is also equipped with a surrogate model for performance prediction. Experimental results show that the proposed approach can achieve significant speedup over the genetic searching algorithms. The produced three-stage operational amplifiers offer competitive performance compared to manual designs.**

I. Introduction

The operational amplifier is an indispensable part of the analog circuit. It is necessary for designers to efficiently synthesize operational amplifiers that meet the design specifications. The automatic synthesis of operational amplifiers can shorten the design cycle of the circuit and reduce costs. Generally, the automated design of analog circuits can be divided into two parts, the selection/optimization of topology, and the further improvement of the circuit specifications through transistor sizing. Both of them can be formulated as optimization problems. However, there are more than 100 tested operational amplifier topologies in real world [1]. Many customized structures are also proposed by designers to boost the performances. The challenge is how to deal with the huge search space.

For the circuit topology selection/optimization, the early works were basically based on the library and random selection method [2], [3]. In [4], knowledge-based filters such as boundary check (BC) and interval analysis (IA) were used to screen the circuits in the library for rapid synthesis. However, these works require high costs to build and maintain the circuit library, and the choice of topology is also limited. To overcome these shortcomings, the subsequent works turned to the generation/optimization of the circuit topology, and often combine the sizing step at the same time. The topology optimization and sizing were formulated as a mixed-integer nonlinear programming (MINLP) problem in [5]. The topology of the circuit and the design parameters of transistors are represented as integer variables and continuous variables, respectively. In [6], a precedent is proposed for circuit topology optimization based on genetic programming algorithms. The topology of the circuit was represented by a tree structure and was unified with the sizes of the transistors to utilize genetic programming for automatic optimization. This approach allows us to easily explore arbitrary connections between different devices, which greatly increases the diversity of the topological search space. With the improvement of genetic algorithm (GA), many similar works were proposed [7]–[11]. In [9], [10], predefined analog building blocks are used to construct the search space of circuit topology. In [11], the upper triangular matrix and continuous variable vector are used to represent the connections between different devices and their sizes.

*Corresponding authors: {yangfan, xzeng}@fudan.edu.cn.

The definition of the search space and the optimization algorithm jointly determine the efficiency and effect of the entire topology optimization. However, search space based on transistor-level descriptions will still produce a large number of invalid circuit structures even after complex rule settings, which will greatly reduce the efficiency of optimization. The diversity of the search space constructed by basic circuit building blocks or cells will be greatly reduced. Inspired by the process of manually designing the circuit, we found that it is more efficient to optimize the circuit and sizing at the behavioral-level. This approach can not only ensure the diversity of the overall search space but also reduce the appearance of physically invalid circuits. Moreover, it is very easy to convert the behavioral-level description of the circuit to a transistor-level circuit.

Naturally, the behavioral-level structure of the circuit topology can be transformed into a directed acyclic graph (DAG) representation [12]. However, due to the discrete nature of DAGs, the most advanced black-box optimization techniques such as Bayesian Optimization based on continuous space are difficult to apply to DAG optimization. [12] embeds the DAG representing the circuit topology into the continuous space to construct an online surrogate model, which can accelerate the efficiency of the entire search. But the search in the topological space is still forced to use a less efficient genetic algorithm due to the limitation of discrete-input characteristics.

In this paper, we propose an efficient and practical three-stage operational amplifier topology automatic optimization method. The behavioral topologies of operational amplifiers are represented as DAGs. To solve the discrete and expensive DAG optimization problem, we customized a variational graph auto-encoder (VGAE) to embed the topology in a continuous and smooth hidden space for optimization. A surrogate model which is used to predict the performances of an operational amplifier is also built based on the embedding in continuous space. The optimization can thus be performed in the continuous embedding space through methods like Bayesian Optimization. After optimization in continuous space, the corresponding optimal embedding can be transformed back to a topology of the operational amplifier through the graph decoder. It is worth mentioning that since the topology has been embedded in the continuous space, we can use the gradient-based method to solve the optimization problem. Experimental results show that our proposed approach can achieve significant speedup over the evolutionary-like searching algorithms in discrete graph space. Also, our proposed approach can obtain competitive three-stage operational amplifiers compared to manual designs.

The remainder of this paper is organized as follows. Section II presents the problem formulation, the reviews of the VGAE model, and BO algorithm. Our proposed topology optimization approach is presented in section III. In section IV, the experimental results and the comparison with manual design are presented to demonstrate the efficiency of our proposed approach. The paper is concluded in section V.

II. Background

A. Problem Formulation

We regard the topology optimization of the operational amplifier (opamp) as a bi-level single objective optimization problem with inequality constraints. The goal of the upper-level optimization task is to search for the optimal topology that meets the constraints in behavioral-level, and the lower-level optimization task is to determine the size of the devices corresponding to the topology, i.e., the sizing step. The entire optimization flow can be formulated as follows.

$$\begin{aligned} &\underset{g\in\mathcal{G},y\in Y}{\text{maximize}} && f(g,\boldsymbol{y}) \\ &\text{s.t.} && \boldsymbol{y} \in arg\ \underset{\boldsymbol{z}\in Y}{max}\{f(\boldsymbol{z}:g^*):c_i(\boldsymbol{z}:g^*)<0\} \\ &&& \forall i \in \{1\dots N_c\}, \end{aligned} \quad (1)$$

where $g \subseteq \mathcal{G}$ represents the DAG which corresponds to a behavioral topology of an opamp, while the nodes of graph g represent the circuit nodes in the opamp, and the direction and type of the edges represent different circuit devices. $Y \subseteq R^{n_y}$ represents the n_y^{th} device design variables of the opamp. f and c_i denote the Figure of Merit (FOM) for the topology optimization and the i^{th} specification constraint, respectively. N_c is the number of specifications. $\boldsymbol{y}$ is the optimal point where g is fixed at g^* while all constraints are satisfied. Design variable $\boldsymbol{y}$ of each device should be limited in a range $[p_i^-, p_i^+]$ according to design conditions. The goal of the optimization is to find the corresponding topology and transistor sizes that minimize the objective function f. This formulation can also be extended to represent general topology optimization problems.

B. Variational Graph Auto-Encoder

The variational graph auto-encoder [13] is introduced for unsupervised learning on graph-structured data based on the variational auto-encoder (VAE) [14]. Generally, a graph convolutional network (GCN) serves as the encoder while an inner product layer serves as the decoder.

Given an undirected and unweighted graph $g = (V, E)$, where $N = |V|$ is the number of nodes. $\boldsymbol{A}$ is introduced as the adjacency matrix of g with degree matrix $\boldsymbol{D}$. Node features are represented by matrix $\boldsymbol{X}$. The stochastic latent variables $\boldsymbol{z}_i$ are represented by matrix $\boldsymbol{Z}$. The encoder characterizes the latent variables as a conditional Gaussian distribution $q(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{A})$. An embedding can then be drawn from this conditional Gaussian distribution. An encoder with a two-layer GCN can be formulated as follows [15]

$$q(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{A}) = \prod_{i=1}^{N} q(z_i|\boldsymbol{X},\boldsymbol{A}), \quad (2)$$

where

$$q(z_i|\boldsymbol{X},\boldsymbol{A}) = \mathcal{N}(\boldsymbol{z_i}|\boldsymbol{\mu}_i, diag(\boldsymbol{\sigma}_i^2)).$$

Here, $\boldsymbol{\mu} = GCN_{\boldsymbol{\mu}}(\boldsymbol{X},\boldsymbol{A})$ is the matrix of mean vector $\boldsymbol{\mu}_i$, and $log\boldsymbol{\sigma} = GCN_{\boldsymbol{\sigma}}(\boldsymbol{X},\boldsymbol{A})$, which are obtained from two separate two-layer GCN. A more detailed description of GCN can be found in [15].

The inner product layer for the decoder can be formulated as follows

$$p(\boldsymbol{A}|\boldsymbol{Z}) = \prod_{i=1}^{N}\prod_{j=1}^{N} p(A_{ij}|\boldsymbol{z}_i), \quad (3)$$

where

$$p(A_{ij}=1|\boldsymbol{z}_i,\boldsymbol{z}_j) = \sigma(\boldsymbol{z}_i^\top \boldsymbol{z}_j).$$

Here $\sigma(\cdot)$ is the logistic sigmoid function and A_{ij} are the elements of $\boldsymbol{A}$.

Then we calculate the embeddings with trained GCN, i.e., $\boldsymbol{Z} = GCN(\boldsymbol{X},\boldsymbol{A})$, and get the reconstructed adjacency matrix $\widehat{A}$:

$$\widehat{A} = \sigma(ZZ^T). \quad (4)$$

The VGAE is trained through maximizing the variational low bound $\mathcal{L}$ with respect to the variational parameters $\boldsymbol{W}_i$

$$\mathcal{L} = \mathbb{E}_{q(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{A})}[logp(\boldsymbol{A}|\boldsymbol{Z})] - KL[q(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{A})||p(\boldsymbol{Z})], \quad (5)$$

where $KL[q(\cdot)||p(\cdot)]$ represents the Kullback-Leibler divergence between $q(\cdot)$ and $p(\cdot)$.

C. Bayesian Optimization

Bayesian Optimization (BO) is an efficient global optimization algorithm for black-box functions. Generally, BO consists of two main parts, the online surrogate model and the acquisition function [16]. The overall process of BO is summarized in Algorithm 1. Gaussian Process (GP) is used as the surrogate model and weighted Expected Improvement (wEI) is used as the acquisition function. The optimization flow will stop after T iterations and regard the best $f(\boldsymbol{x})$ recorded as the optimal solution.

Algorithm 1 Bayesian Optimization

```
1: Generate sampling points and construct GP;
2: while $t \le T$ do
3:     Find the $\boldsymbol{x_t}$ by maximizing the $wEI$;
4:     Sample $y_t = f(\boldsymbol{x_t})$;
5:     Update GP;
6: end while
7: return The best $f(\boldsymbol{x})$ recorded during iterations;
```

III. Proposed Approach

In this section, we will present our proposed topology optimization method based on the customized variational graph auto-encoder model. Bayesian Optimization is utilized to solve the optimization problem in continuous embedding space.

A. Behavioral Modeling

We summarize our framework in Figure 1. We aim to optimize the topology and sizes of a three-stage operational amplifier. The operational amplifier can be described in behavioral-level as shown in Figure 1. We use an ideal voltage-controlled current source (VCCS) with a pair of parasitic capacitor C and resistor R connected to the ground to simulate a single-stage amplifier stage g_m. For a three-stage operational amplifier, there will be 5 vertices in the graph: $\{V_{in}$, V_{out}, gnd, $node1$, $node2\}$. $node1$ and $node2$ are the intermediate nodes of the three-stage operational amplifier. There are 10 edges between

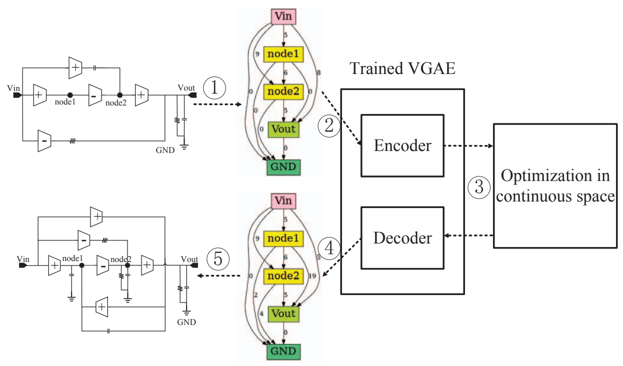
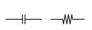
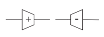
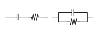
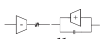


Fig. 1. The overview of the proposed framework, including behavioral level topology mapping, VAE-based encoding, topology search and sizing, topology decoding, and circuit mapping.

TABLE I
ILLUSTRATION OF CONNECTION TYPES BETWEEN DIFFERENT NODES.

Device types	Device directions	Schematic examples	Total number
R, C	+		2
$g_m^{forward}$, g_m^{back}	+/-		4
$\{R, C\}$ in parallel/serial	+		2
$\{g_m, R/C\}$ in parallel/serial	+/-		16
null	null	null	1

the 5 vertices. The edges can be one of the types of the 25 possible connections as summarized in Table I. Therefore, there are 25^{10} possible topologies for exploration. Meanwhile, for each topology, the corresponding parameters such as g_m, resistances, and capacitances can be further tuned to adapt to different technology processes. The behavioral model of the three-stage operational amplifier can be represented as a DAG as shown in Figure 1. The nodes of the circuit are also the nodes of the DAG, and the edges have weights corresponding to the types of the connection as summarized in Table I. Different g_m, capacitors, and resistances are connected to each other to form a behavioral-level topology of the opamp. The DAGs corresponding to all possible opamp topologies form the search space $\mathcal{G}$.

B. Overview of the Proposed Framework

An overview of our proposed approach is shown in Figure 1. In a single iteration, (1) The behavioral-level topologies of the operational amplifier are mapped to DAGs; (2) The trained VGAE encoder embeds DAGs representing the circuit topologies into the matrix $\boldsymbol{Z}$ in a continuous space $\mathcal{C}$; (3) Bayesian Optimization algorithm is utilized to optimize the operational amplifier in the continuous space $\mathcal{C}$ to meet the design specifications. The optimized embedding vector z^* will be returned; (4) The trained VGAE decoder decodes z^* from the embedding space $\mathcal{C}$ to a DAG g^*; (5) The optimized DAG g^* is mapped back to the behavioral-level topology of the opamp, and the optimized circuit and netlist are generated. The construction and training of the VGAE model will be discussed in Section III-C. The Bayesian Optimization algorithm that optimizes the topology and device sizes of the opamp in a continuous space will be discussed in Section III-D.

C. Customized Variational Graph Auto-Encoder

In our method, DAG is used to represent the behavioral-level topology of operational amplifiers. However, we cannot quantitatively measure the distance between different graphs in discrete space, which makes it impossible to directly optimize the DAG. We noticed that variational auto-encoders (VAE) are often used to reduce dimensionality and feature extraction of graphs. More importantly, the original data is embedded in another continuous space that follows the Gaussian distribution. Therefore, we customized a variational graph auto-encoder for the DAG with edge weights mapped from the three-stage operational amplifier. Different from the VGAE for the undirected graph as presented in Section II-B, in our proposed auto-encoder, the GNN is updated along with the weights and directions of the directed edges.

Algorithm 2 Customized VGAE Encode and Decode Algorithm

```
1:  INPUT: DAG represent the opamp topology $G = (V, E)$;
2:  OUTPUT: Embed feature $\boldsymbol{z}$ of $G$ in continuous space,
    reconstructed DAG $G_{re}$;
3:  Encode:
4:  for $v$ in $V$ do
5:      get $v$'s predecessors $V_p$;
6:      $\boldsymbol{h}_v^{in} = \mathcal{A}(h_{V_p} : V_p \to v)$;
7:      $\boldsymbol{h}_v = \mathcal{U}(x_v, \boldsymbol{h}_v^{in})$;
8:  end for
9:  $q(\boldsymbol{Z}|G) = MLP_{encode}(\boldsymbol{h}_v)$;
10: sample $\boldsymbol{z}$ from $q(\boldsymbol{Z}|G)$;
11: return $\boldsymbol{z}$;
12: Decode:
13: $\boldsymbol{h}_d = GRU_{decode}(\boldsymbol{z})$;
14: $p_{edge} = MLP_{decode}(\boldsymbol{h}_d)$;
15: get samlpe edges $E_{re}$ with probability $p_{edge}$;
16: return $G_{re} = (V, E_{re})$;
```

Encoder. Instead of the standard graph neural network (GNN), we leverage the idea in [17] to construct the encoder of our customized VGAE, which is based on the asynchronous propagation of information and conforms to the flow of current in the circuit. As can be seen from Algorithm 2, similar to the standard GNN, the update function $\mathcal{U}$ and the aggregation function $\mathcal{A}$ are both used to obtain the hidden state $\boldsymbol{h}_v$ of the nodes in the graph. For every node v in the DAG, $\boldsymbol{h}_v^{in}$ is the aggregation of the states of all v's predecessors. The difference between [17] and our proposed auto-encoder is that we consider the weights of DAG edges at the same time. Specifically, $\mathcal{A}$ and $\mathcal{U}$ are described by the following formula:

$$\mathcal{A}(h_{V_p} : V_p \to v) = \sum_{V_p} g(e_{V_p \to v} \times \boldsymbol{h}_{V_p}) \odot m(e_{V_p \to v} \times \boldsymbol{h}_{V_p}), \tag{6}$$

$$\mathcal{U}(x_v, \boldsymbol{h}_v^{in}) = GRU_{encode}(x_v, \boldsymbol{h}_v^{in}), \tag{7}$$

where g is a gating network and m is a mapping network. e represents the edge weight between node v and its predecessors V_p, x_v represents the node type. A gated recurrent unit (GRU) [18] is used to model the update function $\mathcal{U}$.

Figure 2 gives an example of how the information at node V_{out} is propagated. The propagation of node information starts from the input node V_{in} of the circuit and ends with a virtual node V_n, which is the successor of the output node V_{out} of the circuit and the GND node. The hidden state at the last node $\boldsymbol{h}_{v_n}$ in the graph will be fed to two multilayer perceptrons (MLP) to get the posterior approximation $q(\boldsymbol{Z}|G)$ in Equation 2.

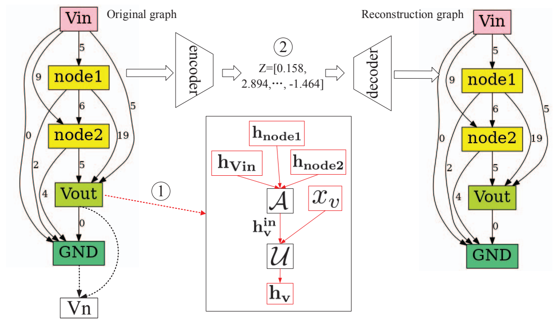


Fig. 2. Illustration of (1) information propagation of a certain node. (2) graph embedding and reconstruction with VGAE model.

Decoder. For the VGAE decoder, we use another GRU_{decode} to update the embedded vector and then feed it to an MLP_{decode} to compute the edge probability p_{edge} in Equation 3. After that, based on the probability p_{edge}, we can sample the reconstructed edge E_{re}, and get the reconstructed DAG $G_{re} = (V, E_{re})$.

Training. In the training phase, we can randomly generate a large number of DAGs. The training process ensures that the decoded DAGs are the same as the DAGs fed into the auto-encoder. The teacher forcing [19] method is used to measure the reconstruction loss following [17]. We sum the negative loglikelihood of each edge by forcing them to generate the ground truth edge type. Mini-batch gradient descent is used to optimize the VAE loss (the negative of (5)). Note that the teacher forcing is only used in the training phase and all the decoded edges are sampled from the probability distribution described in Algorithm 2.

Fortunately, as typical unsupervised learning, VGAE's training does not need to ensure the authenticity of the circuit, and there is no need to execute any time-consuming circuit simulations. Those DAGs used for training can be quickly generated randomly. We think this is an effective way to deploy neural networks in the EDA algorithm, which can avoid barriers in data collection. Based on the graph space $\mathcal{G}$ described in section III-A, we constructed a dataset with 10000 three-stage opamps and DAGs. We used one GeForce RTX 2080 Ti GPU to train the model for 300 epochs, and the entire training was completed in less than 5 hours. The trained VGAE model can achieve 100% reconstruction accuracy for edges of DAGs that have not been seen in the predefined graph space. Figure 2 shows how we embed and reconstruct the graph with the trained VGAE model. Note that the training is only needed to do one time. The trained model can be reused for the optimization of this kind of circuit, even the specifications are changed.

At the same time, we designed a small experiment to verify the ability to embed continuous vectors to represent DAGs. Suppose there are three different DAGs g_1, g_2, and g_3, which respectively represent three different three-stage opamps. g_1 and g_3 are DAGs randomly selected in space $\mathcal{G}$, and g_2 is a DAG with minor changes to g_1. After embedding the three DAGs through the encoder, continuous vectors $\boldsymbol{z}_1$, $\boldsymbol{z}_2$, and $\boldsymbol{z}_3$ can be obtained. We calculated the l_2 norm between the three vectors and took the average after repeating 100 times. The result of $||\boldsymbol{z}_1 - \boldsymbol{z}_2||$ is 1.943, and $||\boldsymbol{z}_1 - \boldsymbol{z}_3||$ is 9.355, which shows that the embedding vector $\boldsymbol{z}$ of DAG can effectively measure the differences of the graphs.

D. Optimization of DAG in the Continuous Space

As can be seen from Figure 1, thanks to the use of VGAE, we can build a bi-level Bayesian Optimization flow to optimize the circuit DAG and execute device sizing. The algorithm is specifically summarized in Algorithm 3. G is the initial training set, which is used to build the initial surrogate model for Bayesian optimization in upper-level, i.e., topology level. The entire optimization flow will be executed T iterations. After encoding, the DAG is embedded in the continuous vector Z, which will be used to construct GP models and acquisition functions in the upper-level optimization. After the DAG representing the opamp topology is determined, the lower-level Bayesian optimization based on Algorithm 1 will be started to determine the sizes of the devices in this topology. Finally, the algorithm will return the optimal opamp topology g^* and corresponding size y^* that satisfies all constraints.

Algorithm 3 Bi-level Bayesian Optimization for DAG

```
1: Initialize trainset with opamp $G$;
2: for $t = 1, 2, \cdots, T$ do
3:     Encode $g$ in $G$ to embedding $\boldsymbol{Z}$;
4:     Construct/Update GP models of objective and constraint
       functions in upper-level optimization with $\boldsymbol{Z}$;
5:     Find the $\boldsymbol{x_t}$ by maximizing the acquisition function with
       gradient descent;
6:     Find the best $f_t = f(\boldsymbol{x_t} : \boldsymbol{y_t})$ and $\boldsymbol{y_t}$ with Algorithm 1;
7:     Decode $\boldsymbol{x_t}$ to $g_t$ with trained decoder;
8:     Update trainset with $(g_t, f_t)$;
9: end for
10: return The best $f(g^*, \boldsymbol{y^*})$ recorded during $T$ iterations;
```

IV. Experimental Results

A. Experiment Setup

For comparison purpose, we used the GA-based method in [12] as the baseline for topology search. We conducted three independent repeated experiments for both methods and set the number of optimization iterations T to 50. We randomly sample 20 points as the initial training set of GP. The training set and the number of simulations performed in each experiment remain the same. All experiments were conducted on a Linux workstation with two Intel Xeon CPUs and 512 GB memory. Both behavioral and transistor-level simulations of the circuit are based on Cadence Spectre simulator.

B. Behavioral-level Simulation Results

Firstly, we set the circuit specifications of the three-stage operational amplifier to drive the entire optimization process. Due to the limitation that an ideal VCCS can only perform AC simulation, we use the FOM_S and constraints described in Equation 8. $GAIN$ refers to the DC gain which is expected to be larger than 85dB, PM refers to the phase margin which is excepted to be larger than 55°, GBW means the gain bandwidth product and is excepted to be larger than 0.7MHz. We use the empirical formula to estimate the power consumption as $Power$ of the opamp and hope it to be less than 250μW. The supply voltage V_{source} and load capacitance C_L are set to 1.2V and 10nF respectively. We use our proposed method and GA algorithm to find the optimal solution of FOM_S while all constraints were satisfied.

TABLE II
OPTIMIZATION RESULTS OF THE BEHAVIORAL-LEVEL CIRCUIT.

EXPs	Methods	FOM_S	$GAIN$	PM	GBW	$Power$	#CNVG
1	our method	**573497**	86.1	56.8	0.7	14	**28**
	GA in discrete space [12]	412936	86.5	57.1	2.3	103	45
2	our method	193203	86.4	60.4	2.4	154	**36**
	GA in discrete space [12]	**247473**	86.7	60.1	2.7	112	46
3	our method	**235292**	86.2	55.9	2.2	97	**21**
	GA in discrete space [12]	224133	86.1	60.1	2.6	121	31
Average	our method	**333998**	86.2	57.7	1.8	88	**28**
	GA in discrete space [12]	294847	86.4	59.1	2.5	112	41

$$\begin{array}{llll} \text{maximize} & FOM_S & & \\ \text{s.t.} & GAIN & > & 85\text{dB} \\ & PM & > & 55^\circ \\ & GBW & > & 0.7\text{MHz} \\ & Power & < & 250\mu\text{W}, \end{array} \tag{8}$$

where

$$\begin{cases} Power & = \frac{\Sigma_{i=1}^{n}|g_{m1}|+|g_{m2}|+\cdots+|g_{mn}|}{20} \times V_{source} \\ FOM_S & = \frac{GBW[MHz]\times C_L[pF]}{Power[mW]} \\ C_L & = 10nF \\ V_{source} & = 1.2V. \end{cases} \tag{9}$$

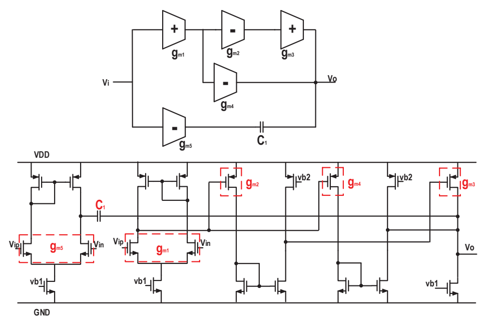
Table II shows the results of optimizing the three-stage operational amplifier on the behavioral-level by two methods. The three experiments are based on different initial training sets. #CNVG refers to the number of iterations required for the optimization result to reach convergence in each experiment. As can be seen from Table II, both two methods can find the topology of the three-stage operational amplifier that satisfies the constraints in all three experiments. Compared with using the GA algorithm to search in the discrete graph space, our proposed method of optimizing in the embedded continuous space can increase the value of FOM_S by an average of 13.3%. More importantly, while being able to find a better solution, our method reduces the number of required optimization iteration convergences by an average of 31.8% in three experiments, and at most 37.8%. The experimental results show that the idea of embedding graphs in continuous space can significantly speed up the optimization of DAG.

The simulation results of the three-stage operational amplifier listed in Table II are all on the behavioral-level, which is based on the modeling of ideal VCCS devices and empirical formulas. In section IV-C, we will map these behavioral-level opamps to transistor-level circuits for further verification.

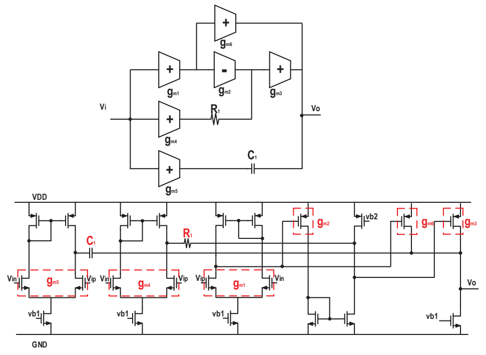
C. Transistor-level Simulation Results

The simulation results of the circuit at the behavioral-level can verify the correctness of its function and give a rough estimate of the specification. Next, simple rules will be used to map the behavioral-level description of the opamp to the transistor-level circuit. For the input stage of the three-stage opamp, we implemented it with differential input, single-ended output amplifiers in current mirror mode. For the intermediate stage and the output stage, we use a simple common-source amplifier for implementation. To prove the practicality of the whole design idea, when mapping the transistor-level circuit, we gave up the usage of techniques such as current-reuse for further optimization, which can reduce the power consumption of the opamp and improve the FOM_S. Figure 3 shows the behavioral-level topologies searched by our method and their mapped transistor-level schematics. All our circuits are implemented with a 40nm process.

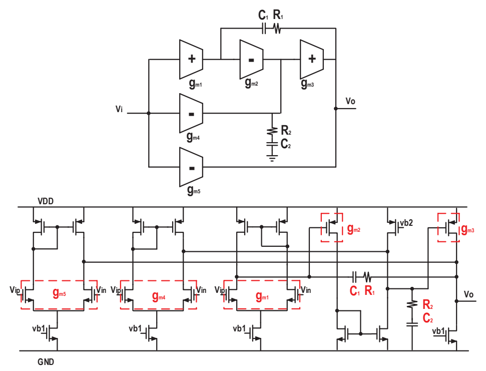
When the transistor-level schematic of an operational amplifier is obtained, we will perform DC simulation to get



(a) case1.



(b) case2.



(c) case3.

Fig. 3. The behavioral-level topologies of all three runs and their mapped transistor-level schematics.

real power consumption results. We compare the three-stage operational amplifiers published in top journals in recent years with the circuits obtained with our method in Table III. As can be seen from Table III, the FOM_S, which is the optimization target, can reach a level similar to or even better than the advanced operational amplifier carefully designed by hand. At the same time, the other specifications of the opamp have reached a relatively excellent level. According to the octagonal rule of analog circuit design, there is a strong constraint relationship between each circuit specification. For our method, by modifying the settings in (8), circuits with another set of design specifications can be easily obtained.

Note that there is a certain loss between the simulation results of Table II and Table III. This is because the process of mapping to transistor-level circuits is strongly related to the CMOS technology. Specifically, under different manufacturing processes, the single-stage gain of the transistor cannot reach the desired value under a certain frequency characteristic. In addition, the estimation of circuit power consumption based on the empirical formula will also have a loss with the real power consumption.

TABLE III
MEASURED PERFORMANCE COMPARISON OF RECENT PRIOR-ART THREE-STAGE AMPLIFIERS.

Benchmark	Yan [20]	Tan [21]	Qu [22]		This Work		
	JSSC'13	JSSC'14	JSSC'16	case1	case2	case3	case3(post-layout)
FOM_S	73611	180441	74569	282051	52083	101064	69861
$C_L(PF)$	10000	680	1500	10000	10000	10000	10000
$GAIN(dB)$	>100	>100	>100	80.8	80.2	85	84
$PM(deg)$	57.2	45	75.5	47	50	55	53
$GBW(MHz)$	1.06	3.37	1.46	1.1	1.0	1.9	1.31
$Power(\mu W)$	144	12.7	69.6	39	192	188	188
$Technology(\mu m)$	0.35	0.13	0.18	0.04	0.04	0.04	0.04

We performed a post-layout simulation for case3, where all specifications are more balanced. Fig. 4 shows the layout. From Table III we can see that the post-layout simulation results are still reliable, except for some loss in GBW. This is caused by more significant parasitic effects in the layout. Therefore, we can conclude that our method can efficiently optimize the usable three-stage operational amplifier with competitive performance.

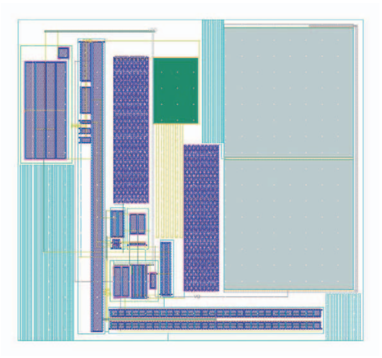


Fig. 4. The layout of case3.

V. CONCLUSION

This paper presents a topology optimization method for operational amplifiers based on a variational graph autoencoder. The behavioral description of a multi-stage opamp is first mapped to a directed acyclic graph and then embedded in the continuous space using VGAE. Opamp topology search and device sizing are performed in the continuous embedding space using Bayesian optimization. Experimental studies demonstrate the capability of VGAE-based representation learning for opamp topological structure, and the efficacy of the proposed topology optimization framework. As a case study, the produced three-stage opamp offers competitive performance advantages over manual designs.

ACKNOWLEDGEMENT

This research is supported partly by National Key R&D Program of China 2020YFA0711900, 2020YFA0711901, partly by the National Natural Science Foundation of China (NSFC) Research Projects under Grant 61822402, Grant 61774045, Grant 62090025, Grant 61929102, and Grant 62011530132.

REFERENCES

[1] Xiaohan Gao and et al. Layout symmetry annotation for analog circuits with graph neural networks. In *Proceedings of the 26th Asia and South Pacific Design Automation Conference*, pages 152–157, 2021.

[2] Marc GR Degrauwe and et al. Idac: An interactive design tool for analog cmos circuits. *IEEE Journal of solid-state circuits*, 22(6):1106–1116, 1987.

[3] Ramesh Harjani and et al. Oasys: A framework for analog circuit synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 8(12):1247–1266, 1989.

[4] Petar Veselinovic and et al. A flexible topology selection program as part of an analog synthesis system. In *Proceedings the European Design and Test Conference. ED&TC 1995*, pages 119–123. IEEE, 1995.

[5] Prabir C Maulik and et al. Integer programming based topology selection of cell-level analog circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14(4):401–412, 1995.

[6] John R. Koza and et al. Automated synthesis of analog electrical circuits by means of genetic programming. *IEEE Transactions on evolutionary computation*, 1(2):109–128, 1997.

[7] C Goh and et al. Ga automated design and synthesis of analog circuits with practical constraints. In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546)*, volume 1, pages 170–177. IEEE, 2001.

[8] Angan Das and et al. Topology synthesis of analog circuits based on adaptively generated building blocks. In *2008 45th ACM/IEEE Design Automation Conference*, pages 44–49. IEEE, 2008.

[9] Trent McConaghy and et al. Variation-aware structural synthesis of analog circuits via hierarchical building blocks and structural homotopy. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(9):1281–1294, 2009.

[10] Trent McConaghy and et al. Trustworthy genetic programming-based synthesis of analog circuit topologies using hierarchical domain-specific building blocks. *IEEE Transactions on Evolutionary Computation*, 15(4):557–570, 2011.

[11] Žiga Rojec and et al. Analog circuit topology synthesis by means of evolutionary computation. *Engineering Applications of Artificial Intelligence*, 80:48–65, 2019.

[12] Jialin Lu and et al. Automated compensation scheme design for operational amplifier via bayesian optimization. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 517–522. IEEE, 2021.

[13] Thomas N Kipf and et al. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[14] Diederik P Kingma and et al. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[15] Thomas N Kipf and et al. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[16] Wenlong Lyu and et al. An efficient bayesian optimization approach for automated optimization of analog circuits. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65(6):1954–1967, 2017.

[17] Muhan Zhang and et al. D-vae: A variational autoencoder for directed acyclic graphs. *arXiv preprint arXiv:1904.11088*, 2019.

[18] Kyunghyun Cho and et al. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[19] Wengong Jin and et al. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pages 2323–2332. PMLR, 2018.

[20] Zushu Yan and et al. A 0.016-mm^2 144-μw three-stage amplifier capable of driving 1-to-15 nf capacitive load with >0.95-mhz gbw. *IEEE journal of solid-state circuits*, 48(2):527–540, 2013.

[21] Min Tan and et al. A cascode miller-compensated three-stage amplifier with local impedance attenuation for optimized complex-pole control. *IEEE Journal of Solid-State Circuits*, 50(2):440–449, 2014.

[22] Wanyuan Qu and et al. Design-oriented analysis for miller compensation and its application to multistage amplifier design. *IEEE Journal of Solid-State Circuits*, 52(2):517–527, 2016.