MACRO: Multi-agent Reinforcement Learning-based Cross-layer Optimization of Operational Amplifier

Zihao Chen[1], Songlei Meng[1], Fan Yang[1*], Li Shang[2], Xuan Zeng[1*]
[1]State Key Lab of Integrated Chips and Systems, School of Microelectronics, Fudan University, Shanghai, China
[2]China and Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University, Shanghai, China

2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC) | 979-8-3503-9354-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/ASP-DAC58780.2024.10473977

***Abstract*—The optimization of operational amplifiers, including topology design and parameter tuning, is significantly challenging, given the high-dimensional and heterogeneous characteristics of the design space. This paper presents MACRO, a novel approach to operational amplifier design that employs multi-agent reinforcement learning for cross-layer optimization. We model the sequentially executed topology design and parameter tuning tasks as a Markov decision process, where the high-dimensional design space is effectively transformed into a series of manageable action spaces at each step. Two agents are meticulously tailored to specialize in these two distinct tasks respectively. The co-evolution of the agents is ensured by sharing design information and customizing the policy-gradient training method. Experimental results show that, compared with state-of-the-art methods, MACRO can produce superior-performing circuits while maintaining competitive design efficiency.**

***Index Terms*—operational amplifier, topology design, parameter tuning, multi-agent reinforcement learning**

I. Introduction

Operational amplifiers (opamps) are some of the most extensively utilized analog circuit modules. The pre-layout design of opamps involves topology design (TD) and parameter tuning (PT, or *sizing*). Even seasoned experts have traditionally performed these steps through a laborious manual tuning process to meet various specifications (specs). With escalating design complexity and relentless time-to-market pressure, automated opamp synthesis tools are increasingly required to expedite the design process and yield superior opamps.

There have been works in recent years focusing either on TD or PT. In the context of TD, graph-based traversal approaches [1], [2] and heuristics like genetic algorithms (GAs) [3], [4] have proven naturally well-suited for the discrete topological design space. Bayesian Optimization (BO) has also demonstrated feasibility [5], [6] when assisted by a graph embedding technique [7]. Recently, methods based on reinforcement learning (RL) [8], [9] are notable for their potential in extensive space exploration. In the context of PT, automated sizing tools employing either BO [10], [11] or RL [12], [13] have considerably succeeded on small-scale analog circuits.

However, all prior works share an essential flaw, i.e., treating TD and PT in isolation, despite their intrinsic interrelation for analog experts. Experts' PT experience for a specific topology can be transferred to other topologies. Conversely, TD knowledge can also be repurposed to reduce sizing difficulty. Regrettably, state-of-the-art works like [6] and [9] adopt a hierarchical optimization mode, where WEIBO [10] is employed to repeatedly tune parameters from scratch to convergence for each searched topology, inevitably leading to suboptimal results. Therefore, it is imperative to propose a topology-parameter co-optimization (so-called *cross-layer optimization*) method for opamps. However, undertaking this endeavor presents the following challenges.

High dimensionality. The pre-layout design space for opamps is high-dimensional. For instance, there are up to 25^{10} behavior-level three-stage opamps topologies [6], where each sample yields up to 29 tunable parameters. Searching within such an expansive space is daunting. It is thus critical to decompose the design space reasonably.

Heterogeneity. The *discreteness* of topological space and the *continuity* of parametric space reflect the essential heterogeneity between TD and PT. TD involves qualitative analysis of module combinations and their impacts on circuit performance. In contrast, PT focuses on quantitative formula analysis and empirical numerical fine-tuning. Thus, performing TD and PT with different modules is necessary.

Co-evolution. The cornerstone of cross-layer design is the co-evolution and reciprocal adaptation of TD and PT modules. For instance, the current PT policy is preferably compatible when updating the TD policy, and vice versa. This demands a rigorous overhaul of traditional methodologies [5], [6], [8], [9], which treat TD and PT hierarchically and isolatedly. It is thus essential to share some information within TD and PT modules and optimize them holistically.

This paper presents MACRO, a multi-agent reinforcement learning (MARL)-based cross-layer optimization framework handling opamp topology and parameters jointly. To the best of our knowledge, this is the first work on cross-layer opamp synthesis. The following techniques are introduced to tackle the above challenges.

- We innovatively frame the opamp design problem as a Markov decision process (MDP). The high-dimensional search space is thus decomposed into low-dimensional action spaces per step. Therefore, we can smoothly handle both topological and parametric spaces.
- We pioneeringly propose a MARL approach to tackle the design-space heterogeneity. Inspired by experts using different knowledge when performing TD and PT, two agents with graph neural network (GNN)-convolutional neural network (CNN) structures are trained to extract different features and adapt to TD and PT, respectively.
- We share circuit design information between the agents and customize the policy gradient (PG) training method [14] to facilitate their co-evolution. The agents consider circuit structures and specs when executing tasks, and receive performance feedback when updating themselves. We customize the PG method to associate the agents' policies with the design trajectory and the final evaluation.

Experimental results show that our framework can generate higher-performance opamps with competitive efficiency, thus preliminarily verifying the superiority of this cross-layer synthesis mechanism.

The remainder of this paper is organized as follows. Section II introduces prior knowledge of circuits and algorithms. Section III proposes our MARL-based opamp optimization method. Section IV presents the experimental results. Section V concludes this paper.

II. Preliminaries

A. Opamp Design Problem

We mathematicize the opamp design to an optimization problem as (1). The circuit is designed by tuning the topology $g \in \mathscr{G}$ and the derived parameters $\mathbf{x} \in \mathscr{X}$.

$$\begin{aligned} &\max_{g\in\mathscr{G},\ \mathbf{x}\in\mathscr{X}} f\left(g,\mathbf{x}\right) \\ &s.t.\ c_i\left(g,\mathbf{x}\right) > c_{th}^{i},\ \forall i \in \{1,...,N_c\}, \end{aligned} \tag{1}$$

where $c_i(\cdot)$ and c_{th}^{i}, normalized to positive values, are the i-th metric and its corresponding spec or so-called constraint threshold, respectively. N_c denotes the number of specs.

*Corresponding authors: {yangfan, xzeng}@fudan.edu.cn. This research is supported partly by National Natural Science Foundation of China (NSFC) research projects 61929102, 62141407 and 62090025, partly by Science and Technology Commission of Shanghai Municipality Project (22DZ1101100).

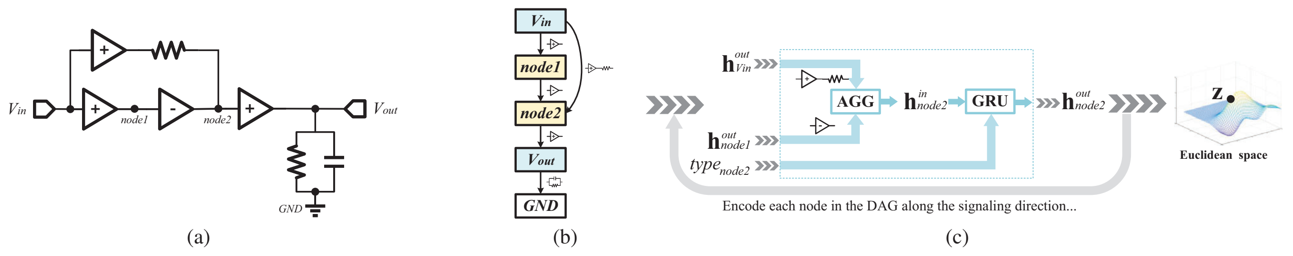
V_{in}
V_{out}
$node1$
$node2$
GND
Vin
node1
node2
Vout
GND
$\mathbf{h}^{out}_{Vin}$
$\mathbf{h}^{out}_{node1}$
$type_{node2}$
AGG
$\mathbf{h}^{in}_{node2}$
GRU
$\mathbf{h}^{out}_{node2}$
Z
Euclidean space
Encode each node in the DAG along the signaling direction...
(a)
(b)
(c)

Fig. 1. Topology modeling from circuit to embedding. Fig. 1(a) shows a three-stage opamp example, compared with the abstracted DAG model in Fig. 1(b). Disconnections are omitted here. In Fig. 1(c), an auto-encoder maps circuit topologies from DAGs into latent vectors (embeddings). Specifically, we only show the encoding process of $node2$ in the circuit example above.

The objective function $f(\cdot)$ evaluates the circuit performance. Experts usually use the small-signal Figure of Merit (FoM) in (2) to score circuits already meeting all specs in practice.

$$\mathrm{FoM} = \frac{\mathrm{GBW}\ [\mathrm{MHz}] \cdot \mathrm{C_L}\ [\mathrm{pF}]}{\mathrm{Power}\ [\mathrm{mW}]}, \tag{2}$$

where load $\mathrm{C_L}$, GBW (gain–bandwidth product), and Power are considered while Gain and PM (phase margin) are ignored.

B. Behavior-level Opamp Topology

Following [6], we model opamps at the behavior level to manage the complexity of transistor-level representation. A single-stage opamp is thus modeled by an ideal voltage-controlled current source (VCCS) $\mathrm{g_m}$ with a parallel pair of parasitic capacitor C and resistor R.

Fig. 1(a) shows a three-stage opamp example. A single topology has five nodes and $\mathrm{C}_5^2 = 10$ connections. The nodes V_{in}, V_{out}, $node1$, $node2$, and GND fall into 4 categories: *input node*, *output node*, *ground node*, and *intermediate node*. Table I shows as many as 25 options for each connection, including *disconnection*.

We abstract the behavior-level model as a directed acyclic graph (DAG) $g = (V; E)$. As shown in Fig. 1(b), each node $v \in V$ corresponds to a circuit node, and each edge $e \in E$ represents a connection. $|V|$ denotes the number of nodes.

TABLE I
CONNECTION TYPES BETWEEN CIRCUIT NODES

Connection types	Directions	Total number
R / C	+	2
$\mathrm{g_m^+}$ / $\mathrm{g_m^-}$	+/-	4
R, C in parallel/serial	+	2
$\mathrm{g_m}$, R/C in parallel/serial	+/-	16
disconnection	null	1

C. Circuit Topology Representation

The variational auto-encoder (VAE) in [6] can extract global and local topological features along the direction of signal transmission. Fig. 1(c) uses $node2$ in Fig. 1(b) as an example to illustrate the VAE's methodology. For each node v, the aggregation function (AGG) accumulates its input information from all predecessors, thus getting a hidden state $\mathbf{h}_v^{in}$ as the input signal. Afterward, the gated recurrent unit (GRU) processes the input signal according to different node types. We eventually get a hidden state $\mathbf{h}_v^{out}$ as the output signal at node v.

After traversing all nodes, we gather their *local* (output) hidden states as $\mathbf{H} = [\mathbf{h}_{V_{in}}, \mathbf{h}_{node1}, ..., \mathbf{h}_{GND}]$. A pair of multi-layer perceptrons (MLPs) transforms $\mathbf{h}_{GND}$ into the Gaussian *global* embedding $\mathbf{z}$. We summarize the encoder by (3).

$$(\mathbf{z}, \mathbf{H}) = \mathrm{VAE}(g). \tag{3}$$

An MLP-based decoder then reconstructs g as $\tilde{g}$. By maximizing the similarity between g as $\tilde{g}$, we obtain a reusable VAE.

III. Proposed Approaches

In this section, we employ MARL techniques to handle TD and PT tasks in Sections III-A and III-B, respectively. These tasks are then unified to construct an opamp design trajectory. The final opamp design is evaluated using the customized reward function in Section III-C. Section III-D presents our approach to cross-layer design policy optimization. Section III-E outlines the workflow of MACRO.

A. Topology Design Task

Following [9], we delegate the TD task to agent T. Inspired by human design flows, we decompose this task into multiple steps to simplify the design space. In each design step, agent T observes the current topology, recalls historical modifications, and based on its learned experience, suggests a policy for the subsequent modification to meet specs. The specific design methodology is detailed as follows.

1) State and Action: According to the human-like design concept described above, we define the state and action spaces, which constitute the input and output of agent T respectively.

- *State space*. As illustrated on the left of Fig. 2, the state consists of the topology state $\mathbf{g}$ and the spec state $\mathbf{c}_{th} = [c_{th}^1, ..., c_{th}^{N_c}]$. Agent T is expected to understand the lineage of the current topology, i.e., the design history $\mathbf{g} = [g_0, g_1, ..., g_n]$ from the initial to the current step n. Hence, the state is formed by merging $\mathbf{g}$ and $\mathbf{c}_{th}$.
- *Action space*. To avoid invalid circuit topologies, we set the initial topology as a basic cascode three-stage opamp, i.e., we fix five backbone edges (all edges except V_{in}-$node2$ in Fig. 1(a)). The remaining five tunable edges produce all possible actions from the $5{\times}25$-dimensional sampling space.

  In practice, we utilize domain knowledge to exclude invalid topologies further, thereby reducing the action space from 125 options to 57. Finally, we need to handle a 57-dimensional categorical probability distribution at each step (see the right of Fig. 2), rather than screening through 25^{10} topologies directly.

2) Agent Structure: Agent T analyzes the current state and generates actions to modify the topology. As shown in Fig. 2, we divide the agent into the representation and policy networks. The former is to understand the current state, while the latter is to make topology modification decisions.

- *Representation network*. The current state is fed into this module to yield the state embedding $\mathbf{z}_{state}$.

  Initially, the pre-trained VAE in Section II-C processes historical topologies in the trajectory (future topologies are filled with zeros) and produces their global topological embeddings respectively, denoted as $\mathbf{z}_{topo}$ collectively in (4).

$$\mathbf{z}_{topo} = \mathrm{VAE}(\mathbf{g}). \tag{4}$$

  Meanwhile, the specs' feature $\mathbf{z}_{spec}$ could be extracted by an MLP (or directly copied from $\mathbf{c}_{th}$). We then merge $\mathbf{z}_{spec}$ and $\mathbf{z}_{topo}$ to produce the state embedding in (5).

$$\mathbf{z}_{state} = \mathbf{z}_{spec}^{\mathrm{T}} \cdot \mathbf{z}_{topo}. \tag{5}$$

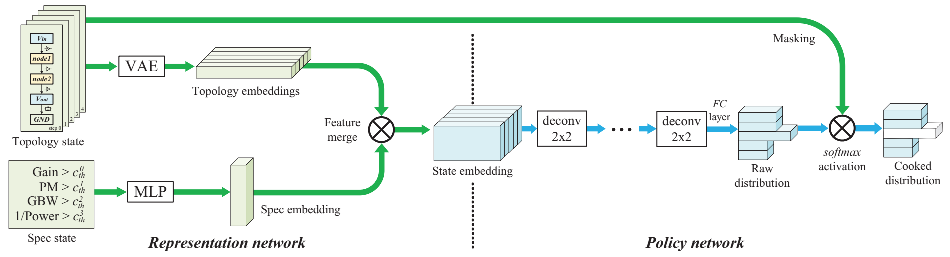

Fig. 2. The proposed agent T.

- *Policy network*. The state embedding $\mathbf{z}_{state}$ is fed into this module to produce the action probability distribution.

  Following [9], we employ 2×2 deconvolution layers to process $\mathbf{z}_{state}$. Each $\mathbf{z}_{state}$ channel corresponds to a historical design state. Deconvolution layers serve a dual purpose. They not only progressively reduce the number of channels, thereby comprehending the design history, but also expand the dimensions within each channel, extracting more detailed and intricate features within the design history. Finally, an MLP shapes the tensor into a 57-dimensional vector, referred to as *raw distribution*.

  To avoid designed edges from being modified repeatedly, we mask their selection probability to 0. Thus, the *softmax* function normalizes the masked *raw distribution* to *cooked distribution*.

  The policy network is summarized as (6).

$$a \sim \pi_{\mathsf{T}}(\mathbf{z}_{state}). \tag{6}$$

3) TD Flow: As shown on the left of Fig. 4, agent T generates a topology by designing total N_{T} edges.

In each step $n \in \{0, ..., N_{\mathsf{T}} - 1\}$, agent T reviews the design history and specs to understand the state s_n, and suggest an action probability distribution π_{T}^n based on its learned experience. According to the sampled action a_n, the state transitions from s_{n-1} to s_n.

We start from the initial state s_0 with a basic three-stage cascode topology and repeats the above step until all edges have been designed. Finally, agent T delivers the complete opamp topology to agent P.

B. Parameter Tuning Task

The downstream PT task is assigned to agent P. We decompose this task into tuning parameters for each device, aligning with the manual tuning paradigm. For each device, agent P considers the device type and topological location, recalls its learned experience, and proposes a parameter policy to meet the specs. Details are as follows.

1) State and Action: Following the above design concept, the state and action spaces of agent P are defined as follows.

- *State space*. We incorporate several key factors typically considered during manual tuning into the state, including the circuit topology $g = (V; E)$, design specs $\mathbf{c}_{th}$, the device type $d_{type} \in \{\mathrm{g_m}, \mathrm{R}, \mathrm{C}\}$, and location of device d. The state of agent P combines the above four. Specifically, we use a relaxed coding vector $\mathbf{L}$ in (7) to locate the device d.

$$\mathbf{L} = [l_0, \ldots, l_{|V|-1}], \tag{7}$$

  where N_{gap} is the device number between the i-th node $v_i \in V$ and device d. $l_i = 1/(N_{gap}+1)$ when agent T places the module (where d belongs) adjacent to v_i. Otherwise, $l_i = 0$.
- *Action space*. We empirically pre-define a parametric space (feasible domain) $\mathscr{X}_d = [x_{min}, x_{max}]$ according to d_{type} for each device d.

2) Agent Structure: Agent P is also comprised of a GNN-based representation network and a CNN-based policy network (see Fig. 3). However, given the intrinsic differences between TD and PT as well as the distinct characteristics of their state and action spaces (discussed in Section I), the structure of agent P, needs to be carefully tailored.

- *Representation network*. In PT, we consider not only the entire topology g but also the specific topological location of device d. To facilitate this, we first use the pre-trained VAE in Section II-C to obtain the tensor $\mathbf{H}$ gathered from the hidden states of each node in topology g, thereby capturing local features.

$$\mathbf{H} = [\mathbf{h}_{V_{in}}, \mathbf{h}_{node1}, ..., \mathbf{h}_{GND}] = \mathrm{VAE}(g). \tag{8}$$

  To emphasize the *key edge* where device d locates, we employ the vector $\mathbf{L}$ in (7) to multiply $\mathbf{H}$, effectively masking non-key edges. Thus we produce the weighted topology state in (9) with device type feature $\widetilde{d}_{type}$ also fused.

$$\widetilde{\mathbf{H}} = \widetilde{d}_{type} \cdot (\mathbf{L} + \delta) \cdot \mathbf{H}, \tag{9}$$

  where δ is a hyperparameter used to adjust the weighting of non-key edges (which are entirely masked when $\delta = 0$). Furthermore, $\widetilde{d}type \in \{1, 2, 4\}$ correspondingly aligns with $dtype \in \mathrm{g_m}, \mathrm{R}, \mathrm{C}$. Various types of devices are thereby effectively differentiated.

  Finally, agent P gathers all features of pending device d and design specs via (10).

$$\mathbf{z}_{state} = \mathbf{z}_{spec}^{\mathrm{T}} \cdot \widetilde{\mathbf{H}}. \tag{10}$$

- *Policy network*. Like agent T, we fuse features between channels continuously through deconvolution layers. This allows us to integrate the entire circuit topology with local information about the current adjustments. Finally, we employ a pair of MLP with the *tanh* activation function to produce normalized mean μ and variance σ, which are used to construct a raw Gaussian distribution.

  Then, we map the regularized μ and σ from $(-1, 1)$ to the actual parametric action space. For μ, we linearly map it into (x_m, x_M) by (11).

$$\widetilde{\mu} = \frac{x_M - x_m}{2}\mu + \frac{x_m + x_M}{2}. \tag{11}$$

  Similarly, we map σ from $(-1, 1)$ to (σ_m, σ_M) by (12).

$$\widetilde{\sigma} = \frac{\sigma_M - \sigma_m}{2}\sigma + \frac{\sigma_m + \sigma_M}{2}. \tag{12}$$

  Considering that $\widetilde{\sigma}$ with too large value brings inevitably low probability density value and gradient vanishment during training, while $\widetilde{\sigma}$ with too small value weakens the search space exploration, we set $\sigma_m = \frac{\Delta x}{30}$ and $\sigma_M = \frac{\Delta x}{10}$ emprically, where $\Delta x = x_M - x_m$.

  Finally, we get a cooked Gaussian distribution in (13) where we sample the parameter value x of device d.

$$x \sim \mathcal{N}(\widetilde{\mu}, \widetilde{\sigma}^2) = \pi_{\mathsf{P}}(\mathbf{z}_{state}). \tag{13}$$

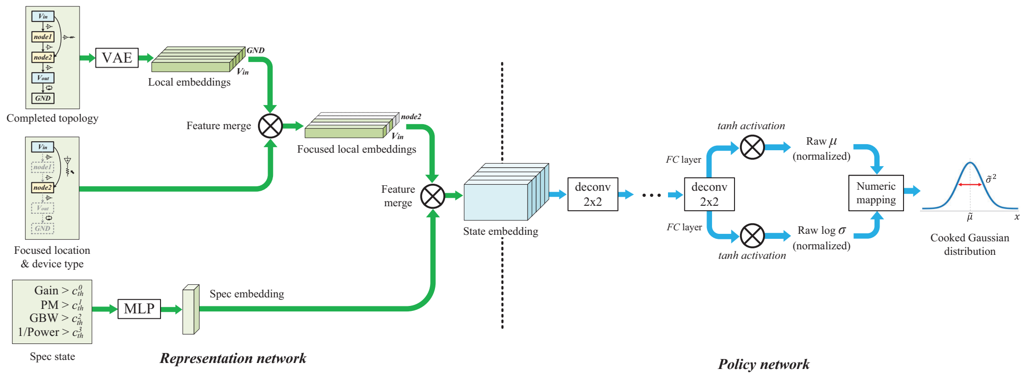

Fig. 3. The proposed agent P.

3) PT Flow: As shown on the right of Fig. 4, assuming that there are N_{P} devices to be tuned, we traverse them along the signaling direction. For the n-th device, agent P observes the device features emphatically, targets the design specs, and finally suggest an parametric Gaussian distribution $\mathcal{N}(\widetilde{\mu}, \widetilde{\sigma}^2)$.

Note that for a specific parameter $x \in [x_m, x_M]$, we should essentially tackle a truncated Gaussian distribution. However, the truncated Gaussian distribution's non-differentiability at boundaries means that if the sampled value is out of boundaries (which often happens), agent P cannot be updated via gradient methods, leading to paralysis of the entire framework. Therefore, we still sample from the original Gaussian distribution to ensure differentiability and limit the out-of-border value at the corresponding boundary.

C. Reward Function

Referring to [9], directly using (2) to score circuits could mislead agents to focus on unqualified circuits with high FoM. Thus a more effective scoring function is customized.

$$\text{score} = \begin{cases} \prod_{i=1}^{N_c} {\alpha_i}^{\gamma_i} + b, & \forall i \in \{1, ..., N_c\} : \alpha_i > 1 \\ \min\limits_{i \in \{1,...,N_c\}} (\alpha_i), & \exists i \in \{1, ..., N_c\} : \alpha_i \leq 1 \end{cases} \tag{14}$$

where γ_i (> 0) adjusts the attention to different metrics while b (> 0) controls the infimum gap between unqualified and qualified designs. Moreover, *redundancy rate* α_i is defined as

$$\alpha_i = \frac{c_i}{c_{th}^i}, \forall i \in \{1, ..., N_c\}, \tag{15}$$

$\alpha_i > 1$ means the i-th spec is already met, $0 < \alpha_i \leq 1$ means the i-th spec is not met yet.

We incorporate constraints into (14), thoroughly transforming the problem in (1) into a solvable unconstrained optimization problem. However, (14) still needs further adjustments to address the potential illegal circumstances.

Invalid topology. Despite our topology constraints in Section III-A1, it is still possible to produce invalid topologies leading to simulation failures. We assign a penalty value like 0 for these invalid topologies.

Negative redundancy. Unlike [9] with the classic sizing tool WEIBO [10] for iterations to ensure goodness of final opamp performance, in MACRO, sizing is actually a one-shot prediction process (see Section III-B3). Therefore, at the early stage of training, poor simulation results sometimes appear, often leading to a negative PM and thus a negative corresponding α_{PM}. Therefore, we have to assign odd values to γ_{PM} to ensure that a negative PM leads to a negative score. Otherwise, a ridiculous, high (positive) score could be generated and guide agents to evolve in the wrong direction.

In summary, the final scoring function $f(\cdot)$ is defined as (16).

$$f(g, \mathbf{x}) = \begin{cases} \text{penalty}, & \text{if simulation fails} \\ \text{score}(g, \mathbf{x}), & \text{otherwise} \end{cases} \tag{16}$$

D. Co-evolution of Agents

Fig. 4 shows a complete opamp design flow. For the i-th sampled flow τ_i under the same version of agents, agent T designs the topology and hands over it to agent P for sizing. The generated circuit is simulated at the behavior level, and the reward R_i is finally calculated.

The interaction between two agents is mathematically reflected in design features (including circuit structures and specs) shared in each state s_i, the simulation reward R_i of the final circuit, and the probability of the entire trajectory τ_i in (17).

$$p_\theta(\tau_i) = \prod_{n=1}^{N_{\mathsf{T}}} p_{\theta_{\mathsf{T}}}(s_{n-1}, s_n) \prod_{n=N_{\mathsf{T}}}^{N_{\mathsf{T}}+N_{\mathsf{P}}} p_{\theta_{\mathsf{P}}}(s_{n-1}, s_n), \tag{17}$$

where agent T and agent P are parameterized by θ_{T} and θ_{P} respectively, denoted as $\theta = \{\theta_{\mathsf{T}}, \theta_{\mathsf{P}}\}$.

We adopt the classic PG method [14] to train the agents. By maximizing the Monte Carlo estimation of the reward expectation given in (18) through the gradient algorithm, the agents tend to evolve synergistically.

$$\bar{R}_\theta = \sum_{i=1}^{N_{batch}} R_i \cdot p_\theta(\tau_i). \tag{18}$$

E. Overall Workflow

The workflow of MACRO is illustrated in Fig. 5.

The system requires only the user-defined specs as input, according to which, the same pair of agents independently replicates opamp design for N_{batch} times, and gets a total of N_{batch} trajectories. In each trajectory, agent T starts from the initial topology, selects all optional edges, and obtains the complete circuit topology, which is then handed over to agent P for PT. The final opamp is automatedly simulated and evaluated by the reward function in (16). The probability $p_\theta(\tau_i)$ and the accumulated reward R_i of each trajectory τ_i are recorded.

The pair of agents is then updated by maximizing (18). The new pair of agents re-samples design trajectories and updates θ until convergence.

We end up with a trained pair of agents and the optimal circuit topology recorded. The circuit is automatedly mapped to the transistor level and re-optimized by WEIBO [10] using scripts based on g_m/I_d method, which is not the focus of this paper. Finally, we get a transistor-level circuit meeting design specs.

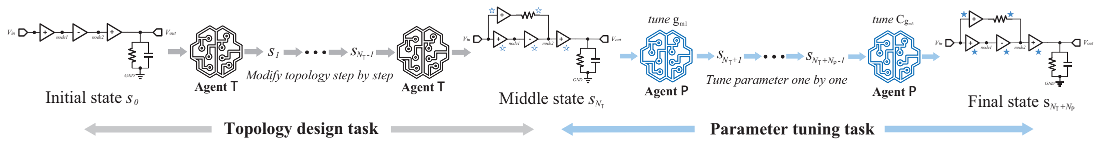

Fig. 4. A complete design trajectory example. Taking the design of Fig. 1(a) as an example, agent T generates the opamp topology in N_{T} steps and agent P designs the opamp parameters in N_{P} steps. Intermediate state transitions are omitted here. Hollow stars and solid stars mark circuit modules that have not been tuned and circuit modules that have been tuned, respectively.

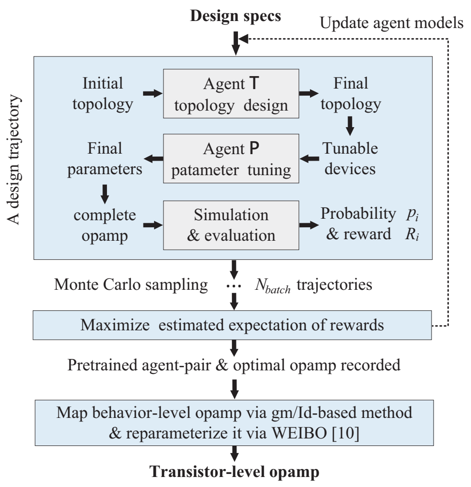

Fig. 5. Workflow of the proposed MACRO framework.

IV. Experiments

A. Experiment Platforms

All simulations utilize the *Cadence Spectre* simulator on a 64-core *Intel Xeon Gold 6226R* CPU. The pre-trained VAE and the agents are deployed on an *NVIDIA RTX 2080Ti* GPU.

B. Experiment Cases

1) Baselines: Comparison baselines include state-of-the-art hierarchical methods [6] and [9], which optimize topology via BO and RL, respectively, and delegate the bottom-layer sizing task to the sizing tool WEIBO [10] for each topology, using 15 initial points and a total of 40 iterations. Thus, we denote [6] as BOBO and [9] as RLBO.

2) MACRO case: We set up this case to evaluate the proposed MACRO framework. The customized scoring function (16) guides the search process while (2) selects optimal circuits from the data pool. The agents are trained from scratch to maximize (18). The behavior-level opamp with the highest FoM throughout the training process is selected. To ensure convergence, we cap the maximum number of training epochs at 200, with the learning rates for agents T and P set to 1×10^{-3}. In each iteration, 128 design trajectories as depicted in Fig.4 are independently and randomly sampled, i.e., $N_{batch} = 128$.

All experiments are repeated three times under randomly initialized conditions, marked by *exp1* to *exp3* in Tables II and III. The circuits obtained by exp1 for all cases above are shown in Figs. 6, 7 and 8.

C. Circuit Settings

Following [6] and [9], we consider four design metrics and their corresponding specs as follows: Gain (> 85 dB), PM (> 55 degree), GBW (> 0.7 MHz), and Power (< 250 μW), for which we empirically assign exponential weights in (14) to 0.4, 0.1, 0.7, and 1 (or 0.1, 0.3, 0.5, and 1), respectively. The capacitive load $\mathrm{C_L}$ and supply voltage V_{source} are set to 10 nF and 1.8 V, respectively.

To compare experimental results more precisely, we map the opamp from the behavior level to the transistor level under the TSMC 180-nm process. In the customized $\mathrm{g_m}/I_d$ mapping scripts, a current mirror differential input and single-ended output implement the input stage, followed by common-source amplifiers for intermediate and output stages. Subsequently, WEIBO [10] reparameterizes all mapped opamps using 20 initial points and a total of 50 iterations.

TABLE II
Behavior-Level Search Results

Methods		FoM	Gain (dB)	PM (degree)	GBW (MHz)	Power (μW)	Time* (h)
BOBO [6]	exp1	573497	85.1	73.5	0.8	13.1	3.04
	exp2	235292	86.2	55.9	2.3	97.2	0.65
	exp3	160779	86.4	60.5	2.5	154.0	4.15
	avg	**323189**	**85.9**	**63.3**	**1.9**	**88.1**	**2.61**
RLBO [9]	exp1	561382	86.0	57.3	2.6	46.2	3.43
	exp2	413630	90.1	56.2	3.1	75.5	2.27
	exp3	726207	86.8	57.9	2.4	33.3	2.77
	avg	**567073**	**87.6**	**57.1**	**2.7**	**51.7**	**2.82**
MACRO	exp1	676001	85.9	55.6	1.4	20.9	2.66
	exp2	785408	85.4	55.5	1.6	20.6	3.34
	exp3	691094	87.0	55.2	1.8	26.0	1.78
	avg	**717501**	**86.1**	**55.4**	**1.6**	**22.5**	**2.59**

* Time refers to the simulation time cost before the optimal solution in the data pool is no longer updated.

TABLE III
Transistor-Level Mapping* Results

Methods		FoM	Gain (dB)	PM (degree)	GBW (MHz)	Power (μW)	$\mathrm{C_L}$ (nF)	Process (nm)
BOBO [6]	exp1	244062	114.0	47.4	0.5	21.2	10	180
	exp2	112886	102.9	55.0	1.7	152.1	10	180
	exp3	184706	105.4	69.9	1.1	59.5	10	180
	avg	**180551**	**107.4**	**57.4**	**1.1**	**77.6**	**-**	**-**
RLBO [9]	exp1	309569	98.1	61.9	1.1	36.9	10	180
	exp2	260901	90.5	55.2	1.8	69.7	10	180
	exp3	237871	89.4	55.0	1.6	66.0	10	180
	avg	**269447**	**92.7**	**57.4**	**1.5**	**57.5**	**-**	**-**
MACRO	exp1	563391	98.1	56.7	2.1	37.9	10	180
	exp2	503255	101.0	61.2	2.3	46.1	10	180
	exp3	613469	98.8	55.2	3.1	50.2	10	180
	avg	**560038**	**99.3**	**57.7**	**2.5**	**44.7**	**-**	**-**

* We use the mapping and re-optimization tool in [15] open sourced at https://github.com/jialinlu/OPAMP-Generator.

D. Experimental Results and Analysis

1) Search results comparison: Table II shows that all methods generate opamps meeting specs with similar search efficiency under our parallel simulation condition. MACRO's average FoM is 2.22× and 1.27× that of BOBO [6] and RLBO [9], respectively.

Such results underscore RL frameworks' high-dimensional space exploration capability. RLBO [9] and MACRO use agent T to simplify TD into manageable steps, enabling smoother topological space exploration. Conversely, the top-layer BO in BOBO [6], converges prematurely to suboptimal topologies. The average FoMs of RLBO [9] and MACRO are thus 1.75× and 2.22× that of BOBO [6].

Such results also highlight MACRO's heterogeneous space exploration capability. MACRO assigns PT to agent P as a prediction task, ablating the BO-based sizing iterations in BOBO [6] and

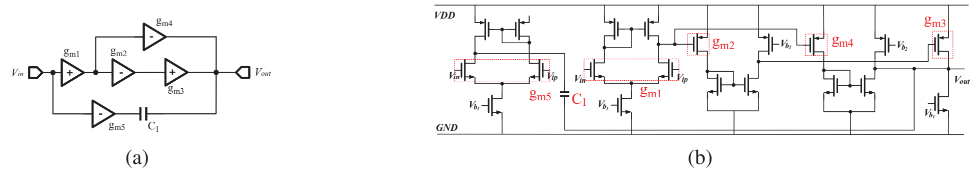

Fig. 6. Behavior model and its transistor-level schematic of the final selected circuit in BO-based baseline [6] (*exp1*).

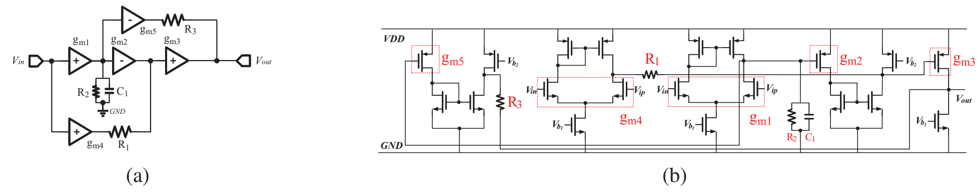

Fig. 7. Behavior model and its transistor-level schematic of the final selected circuit in RL-based baseline [9] (*exp1*).

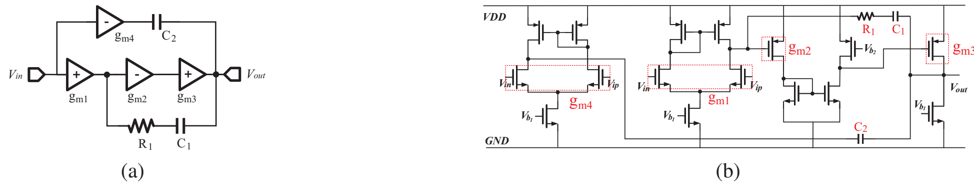

Fig. 8. Behavior model and its transistor-level schematic of the final selected circuit in MACRO (*exp1*).

RLBO [9]. Drawing upon agent structures in Sections III-A & III-B and our training mode in Section III-D, the agents mutually adapt and collaborate to yield superior circuits and marginally improve efficiency.

2) Mapping results comparison: We map opamps from the behavior level to the transistor level, re-optimize parameters via WEIBO [10], and yield Table III. After mapping, MACRO still outperforms, with an average FoM 3.1× and 2.08× that of BOBO [6] and RLBO [9] respectively, exceeding behavior-level outcomes (2.22× and 1.27×). During mapping, MACRO's average FoM degradation is 0.78×, while BOBO [6]'s and RLBO [9]'s are 0.56× and 0.48× respectively.

Such results are credited to our cross-layer optimization mechanism. In MACRO, agents T and P co-evolve and interact, leading agent T to prefer circuits easier for agent P to adjust (given agent P's limited capabilities due to the parameter prediction method). This contributes to a preference for a simpler behavior-level parametric space. When mapped to the transistor level, some parametric space simplicity is maintained, thus easing the WEIBO [10]-based resizing.

On the other hand, BOBO [6] and RLBO [9] employ WEIBO [10] to tune behavior-level parameters, which is more potent than agent P, thus beneficial to behavior-level results. After mapping, however, transistor-derived new parameters escalate the space's complexity (including dimensionality), which could stretch WEIBO [10] and cause more serious FoM degradation. Especially for RLBO [9], whose exploratory agent T prefers more complex topologies, the FoM degradation (0.48×) is worse than BOBO [6] (0.56×).

V. Conclusions

We propose MACRO, a MARL-based method for cross-layer opamp optimization. We model TD and PT tasks as MDPs, decomposing the high-dimensional design space into manageable steps. Two customized agents tackle these tasks, respectively, enabling the exploration of the heterogeneous design space. The agents' shared design features and customized training method facilitate their co-evolution. Simulation results show that, compared with state-of-the-art methods, MACRO can generate superior opamps with competitive design efficiency.

References

[1] M. Meissner and L. Hedrich, "Feats: Framework for explorative analog topology synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 2, pp. 213–226, 2014.

[2] Z. Zhao and L. Zhang, "Graph-grammar-based analog circuit topology synthesis," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2019, pp. 1–5.

[3] C. Mattiussi and D. Floreano, "Analog genetic encoding for the evolution of circuits and networks," *IEEE Transactions on evolutionary computation*, vol. 11, no. 5, pp. 596–607, 2007.

[4] Ž. Rojec, Á. Bűrmen, and I. Fajfar, "Analog circuit topology synthesis by means of evolutionary computation," *Engineering Applications of Artificial Intelligence*, vol. 80, pp. 48–65, 2019.

[5] J. Lu, L. Lei, F. Yang, C. Yan, and X. Zeng, "Automated compensation scheme design for operational amplifier via bayesian optimization," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2021, pp. 517–522.

[6] J. Lu, L. Lei, F. Yang, L. Shang, and X. Zeng, "Topology optimization of operational amplifier in continuous space via graph embedding," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2022, pp. 142–147.

[7] M. Zhang, S. Jiang, Z. Cui, R. Garnett, and Y. Chen, "D-vae: A variational autoencoder for directed acyclic graphs," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[8] Z. Zhao and L. Zhang, "Analog integrated circuit topology aynthesis with deep reinforcement learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2022.

[9] Z. Chen, S. Meng, F. Yang, L. Shang, and X. Zeng, "Total: Topology optimization of operational amplifier via reinforcement learning," in *2023 24th Int'l Symposium on Quality Electronic Design (ISQED)*. IEEE, 2023.

[10] W. Lyu, P. Xue, F. Yang, C. Yan, Z. Hong, X. Zeng, and D. Zhou, "An efficient bayesian optimization approach for automated optimization of analog circuits," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 6, pp. 1954–1967, 2017.

[11] W. Lyu, F. Yang, C. Yan, D. Zhou, and X. Zeng, "Batch bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design," in *International conference on machine learning*. PMLR, 2018, pp. 3306–3314.

[12] H. Wang, K. Wang, J. Yang, L. Shen, N. Sun, H.-S. Lee, and S. Han, "Gcn-rl circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

[13] K. Settaluri, A. Haj-Ali, Q. Huang, K. Hakhamaneshi, and B. Nikolic, "Autockt: Deep reinforcement learning of analog circuit designs," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 490–495.

[14] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[15] J. Lu, L. Lei, J. Huang, F. Yang, L. Shang, and X. Zeng, "Automatic op-amp generation from specification to layout," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2023.