MA-Opt: Reinforcement Learning-based Analog Circuit Optimization using Multi-Actors

Youngchang Choi, Minjeong Choi, Kyongsu Lee, and Seokhyeong Kang*
Department of EE, POSTECH, Pohang, South Korea
*shkang@postech.ac.kr

***Abstract*—Analog circuit design requires significant human efforts and expertise; therefore, electronic design automation (EDA) tools for analog design are needed. This study presents MA-Opt that is an analog circuit optimizer using reinforcement learning (RL)-inspired framework. MA-Opt using multiple actors is proposed to provide various predictions of optimized circuit designs in parallel. Sharing a specific memory that affects the loss function of network training is proposed to exploit multiple actors effectively, accelerating circuit optimization. Moreover, we devise a novel method to tune the most optimized design in previous simulations into a more optimized design. To demonstrate the efficiency of the proposed framework, MA-Opt was simulated for three analog circuits and the results were compared with those of other methods. The experimental results indicated the strength of using multiple actors with a shared elite solution set and the near-sampling method. Within the same number of simulations, while satisfying all given constraints, MA-Opt obtained minimum target metrics up to 24% better than DNN-Opt. Furthermore, MA-Opt obtained better Figure of Merits (FoMs) than DNN-Opt at the same runtime.**

***Index Terms*—Analog circuit optimization, RL-inspired, multiple actors, shared elite solution set, near-sampling method**

I. Introduction

Digital circuit designs exploit electronic design automation (EDA) tools, but analog and mixed-signal (AMS) designs still require human efforts and expertise. For AMS designs, significant knowledge of circuits and many simulations for parameter fine-tuning to satisfy performance specifications are required. Furthermore, device characteristics become more complicated owing to CMOS technology scaling, which complicates AMS designs in an existing manner and incurs a long time to market. Consequently, EDA tools for sizing analog circuit components to provide optimized circuits are required.

Previous studies for automation of analog circuit sizing can fall into two classes: knowledge-based and optimization-based methods. In the knowledge-based methods, knowledge of circuits is translated into methods and equations [1], [2]. However, these methods are not scalable because they are dependent on knowledge that circuit designers have. Optimization-based methods can be divided into two categories: equation-based and simulation-based methods. Equation-based methods [3]–[6] use simulation data to describe circuit performance as polynomial equations or regression models, and to search for an optimal solution, these methods are applied to convex or non-convex formulated problems. Equation-based methods are fast in general, but they can not express circuit performances accurately due to their deviations from the real values obtained by circuit simulations.

On the contrary, simulation-based methods directly exploit the results of circuit simulations. Simulation-based methods employ black-box function optimization methods or learning-based optimization methods. Particle swarm optimization (PSO) [7] and advanced differential evolution (DE) [8] have been proposed to explore the solution space efficiently, but they have low convergence rates. Bayesian optimization (BO) has been applied to analog circuits and it has optimized circuit designs by replacing circuit sizing with black-box function optimization problems using Gaussian process regression (GPR) [9]–[11]. The main drawback of BO is the computational complexity of training, $O(N^3)$, where N is the number of training data points. For BO, the simulation time increases rapidly as the number of simulations increases. Reinforcement learning (RL) methods such as the deep deterministic policy gradient (DDPG) [12] have been employed for analog circuit sizing [13]–[15] as learning-based optimization methods. However, these frameworks require thousands of SPICE simulations for circuit optimization.

To overcome the shortcomings of the RL-based circuit optimizations, DNN-Opt framework [16] has been proposed. DNN-Opt is an RL-inspired method that modifies the DDPG method that uses an RL actor-critic method [17]. In addition, DNN-Opt adopts the advantages of RL, BO, and population-based techniques [20]. Therefore, it can realize optimized circuit designs that satisfy their performance constraints within a few simulations. DNN-Opt deploys a single agent to explore design spaces, so it may be insufficient for DNN-Opt to explore the design spaces because the diversity of circuit parameters provides a number of dimensions for state spaces, and sizing circuit designs requires continuous state spaces.

In this study, MA-Opt, a novel RL-inspired framework that adopts multiple actors and other techniques, is proposed. For RL methods, a parallel RL paradigm, such as the general RL architecture (GORILA) [18] and asynchronous advantage actor-critic (A3C) [19], have been proposed to achieve better results than RL using a single agent. These RL methods deploy multi-agents running in parallel to generate more data and explore the state space more effectively. Similarly, to overcome the drawback of exploration of the existing RL-inspired framework, we propose the RL-inspired framework that uses multi-agents and parallelism. In addition, to exploit multi-agents effectively in an RL-inspired framework, we propose sharing an elite solution set that comprises the best circuit designs simulated and affects the loss function of network training. Furthermore, we propose a tuning technique that finds a more optimized design than the previous best design. The key contributions of this study are as follows:

- The proposed framework uses multiple actors to make diverse predictions of optimized circuit designs. Because multiprocessing is applied, training actors and performing SPICE simulations for circuits predicted by actors are implemented in parallel.
- We propose a shared elite solution set that affects network training. By sharing an elite solution for each actor, the update of an elite solution set is boosted and that assists training actors; thus, sharing an elite solution set for multiple actors enhances circuit optimization.
- We employ a near-sampling method to optimize the circuits. The near-sampling method generates samples adjacent to the most optimized design from previous simulations, and a critic network that mimics the SPICE simulator is applied to predict results of the sampled designs. Because circuit designs are sampled densely, the near-sampling method can successfully search for the optimized design.

The rest of the paper is organized as follows. In Section II, we present the details of the proposed framework. In Section III, we provide the experimental results and compare the performances of our proposed framework with other optimization methods. We conclude the paper in Section IV.

978-3-9819263-7-8/DATE23/© 2023 EDAA

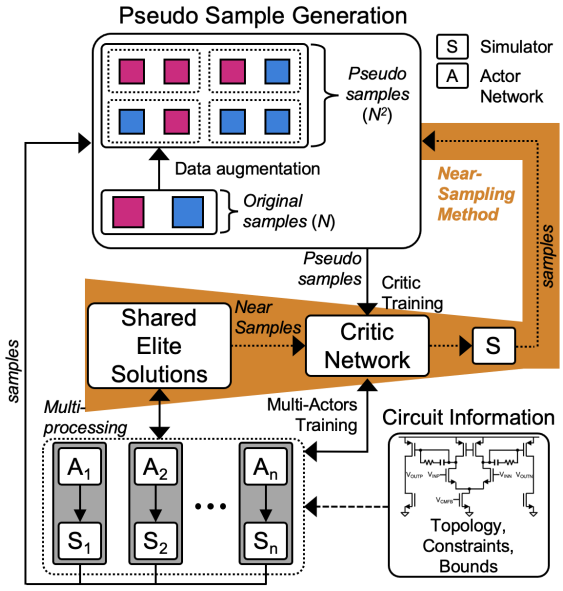

Fig. 1. The proposed framework.

II. Proposed Framework

This section introduces the proposed framework. We formulate the circuit sizing optimization problem, and explain the architecture of MA-Opt. A near-sampling method is devised to find more optimized designs and introduced in this section.

A. Problem Formulation

A constrained optimization problem for analog circuit sizing was formulated as follows:

$$\begin{aligned} &\text{minimize } f_0(\mathbf{x}) \\ &\text{subject to } f_i(\mathbf{x}) \leq 0 \quad \text{for } i = 1, ..., m \end{aligned} \tag{1}$$

where, $\mathbf{x} \in \mathbb{D}^d$ is the parameter vector, $\mathbb{D}^d$ is the design space, and d is the number of design variables to be sized. $f_0(\mathbf{x})$ is the target metric that should be minimized; here, $f_i(\mathbf{x})$ is a metric that denotes the i^{th} constraint.

B. MA-Opt Architecture

MA-Opt consists of an actor-critic deep neural network architecture with multiple actors and a single critic, a circuit simulator, original samples, pseudo-samples, and a shared elite solution set (Fig. 1). By applying population-based techniques [20] to the original samples in the design space, the pseudo-samples are generated and used to train a critic network. Subsequently, the critic network predicts the performance of given parameter vectors. The actor networks propose new parameter vectors for simulations by using the critic network's predictions. For the MA-Opt framework, most notations obey an RL method.

Design: A group of circuit parameters is represented as a vector $\mathbf{x}$ of size d. Finding the value of $\mathbf{x}$ satisfying Eq. 1 is the optimization goal.

Population: A group of simulated designs.

Design Population Matrix: This is denoted by $\mathbf{X} \in \mathbb{R}^{N \times d}$, where N is the population size. The k^{th} row in $\mathbf{X}$ is the k^{th} design $\mathbf{x}_k$.

Total Design Set: A total design set $\mathbf{X}^{tot}$ is a design population matrix that consists of all designs where circuit simulations were executed.

Elite Solution Set: An elite solution set $\mathbf{X}^{ES} \in \mathbb{R}^{N_{es} \times d}$ is a design population matrix that comprises the best N_{es} solutions of the total design set $\mathbf{X}^{tot}$ in terms of a figure of merit (FoM) ranking. To quantify circuit sizing optimization, an FoM function, $g(\cdot)$, is defined as follows:

$$g[f(\mathbf{x})] = w_0 \times f_0(\mathbf{x}) + \sum_{i=1}^{m} \min(1, \max(0, w_i \times |\frac{f_i(\mathbf{x}) - c_i}{c_i}|)) \tag{2}$$

where w_i is the weighting factor, $f_i(\mathbf{x})$ is the i^{th} SPICE simulation value of design $\mathbf{x}$, and c_i is a constraint of $f_i(\mathbf{x})$. Moreover, an elite solution set of size N_{es} is used to limit the search space of an actor network.

State Space: Design $\mathbf{x}$ represents a state as an RL notation. The k^{th} design $\mathbf{x}_k$ represents the k^{th} state $\mathbf{s}_k$.

Action Space: The notation $\mathbf{a}_k$ is an action of the k^{th} state $\mathbf{s}_k$, and is the change required to optimize the k^{th} design $\mathbf{x}_k$, which is denoted as $\Delta\mathbf{x}_k$. When an action of the k^{th} design $\mathbf{x}_k$ is applied, the next design of $\mathbf{x}_k$ is $\mathbf{x}_k + \Delta\mathbf{x}_k$.

Critic Network: In general, a critic-network parameterized by θ^Q approximates the Markov Decision Processes (MDP) return value $Q(\mathbf{s}_t, \mathbf{a}_t|\theta^Q)$. The original role of this network is modified to a regression of the SPICE simulator. Its input is $(\mathbf{x}, \Delta\mathbf{x}) \in \mathbb{D}^{2d}$, and its output is simulation predictions $Q(\mathbf{x}, \Delta\mathbf{x}|\theta^Q) \in \mathbb{R}^{m+1}$, where one-dimension is for target metric and m for constraint metrics.

Actor Network: The role of an actor is prediction for change of optimized designs. To predict various optimized designs, multiple actors were adopted for the RL-inspired framework. All actor networks were parameterized by θ^{μ_1}, θ^{μ_2},... $\theta^{\mu_{N_{act}}}$, where N_{act} denotes the number of actors. Their input is a state $\mathbf{s}_k$, and each actor θ^{μ_i} returns an action $\mathbf{a}_{k,i} = \mu(\mathbf{s}_k|\theta^{\mu_i})$. In MA-Opt, each actor network θ^{μ_i} presents the change to optimize $\mathbf{x}_k$ as: $\Delta\mathbf{x}_{k,i} = \mathbf{a}_{k,i} = \mu(\mathbf{x}_k|\theta^{\mu_i})$.

Critic-Network Training: Population-based techniques that generate N^2 pseudo-samples using N original samples effectively train the critic network. By using two samples, $\mathbf{x}_i$ and $\mathbf{x}_j$, and vectors returned by the circuit simulator, $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$, pseudo-samples are generated as follows:

$$\begin{aligned} \mathbf{x}_{ij}^{\text{ps}} &= (\mathbf{x}_i, \Delta\mathbf{x}_{ij}) = (\mathbf{x}_i, \mathbf{x}_j - \mathbf{x}_i) \\ f^{\text{ps}}(\mathbf{x}_{ij}^{\text{ps}}) &= f(\mathbf{x}_j) \end{aligned} \tag{3}$$

In the proposed framework, pseudo-samples are generated by using designs in the total design set $\mathbf{X}^{tot}$. The critic network is trained by the mean square error (MSE) loss function with a batch-size of N_b pseudo-samples as follows:

$$L(\theta^Q) = \frac{1}{N_b(m+1)} \sum_{k=1}^{N_b} \sum_{l=1}^{m+1} |Q(\mathbf{x}_k, \Delta\mathbf{x}_k)^l - f(\mathbf{x}_k + \Delta\mathbf{x}_k)^l|^2 \tag{4}$$

where $Q(\mathbf{x}_k, \Delta\mathbf{x}_k)^l$ is the l^{th} approximated metric of the k^{th} pseudo-sample and $f(\mathbf{x}_k + \Delta\mathbf{x}_k)^l$ is the simulation result for the k^{th} pseudo-sample. Unlike multiple actors, multiple critics are not used because using multiple regression models for circuit simulation does improve optimization, but consumes more memory resources than using one critic network.

Actor-Network Training: After training the critic network, the actor networks are trained by using the $g(\cdot)$ function and replacing the SPICE simulation values $f(\mathbf{x})$ with the critic-network predictions $Q(\mathbf{x}, \Delta\mathbf{x})$. To search optimized designs by exploiting predictions of actors, a proper loss function for training actors is required. The actor networks are trained by the following loss function with a batch-size of N_b pseudo-samples as follows:

$$L(\theta^{\mu_i}) = \frac{1}{N_b} \sum_{k=1}^{N_b} (g[Q(\mathbf{x}_k, \mu(\mathbf{x}_k \mid \theta^{\mu_i}))] + ||\lambda * \text{viol}_k||_2) \tag{5}$$

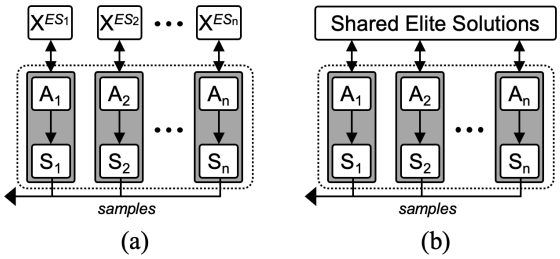

Fig. 2. Types of elite solution sets for multiple actors: (a) individual elite solution sets; (b) a shared elite solution set.

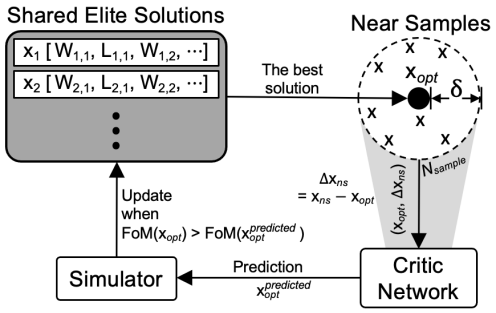

Fig. 3. Framework of the near-sampling method.

where $\mu(\mathbf{x}_k \mid \theta^{\mu_i})$ is the action vector $\Delta\mathbf{x}_{k,i}$ obtained from the i^{th} actor network θ^{μ_i}. When training a single actor, to prevent any boundary violation and restrict the search space, element-wise vector multiplication $(\lambda * \text{viol}_k)$ is applied (Eq. 5) by using an elite solution set, where λ is a significantly large weighting coefficient. For action $\Delta\mathbf{x}_k$, the total boundary violation viol_k is defined as follows:

$$\text{viol}_k = \max(0, lb_{rest} - (\mathbf{x}_k + \Delta\mathbf{x}_k)) + \max(0, (\mathbf{x}_k + \Delta\mathbf{x}_k) - ub_{rest}) \quad (6)$$

where lb_{rest} and ub_{rest} are the boundary vectors determined by the elite solutions as follows:

$$lb^i_{rest} = \min(\mathbf{x}^i) \quad \forall_i = 1, \ldots, d$$
$$ub^i_{rest} = \max(\mathbf{x}^i) \quad \forall_i = 1, \ldots, d$$

where, $\mathbf{x}^i$ is the column vector comprising the i^{th} design parameter of all elite solutions.

In DNN-Opt [16], only one actor is used, so one elite solution set is applied for the actor-network training. However, in MA-Opt, to train multiple actors, each actor can possess its own elite solution set (Fig. 2a), or share one elite solution (Fig. 2b). Of those, using a shared elite solution set $\mathbf{X}^{SES} \in \mathbb{R}^{N_{es} \times d}$ is adopted from two perspectives, the update speed of elite solutions and memory consumption. If actors are trained with their own elite solution set, N_{act} elite solution sets are needed, and each elite solution set can be updated up to only one elite solution for one simulation. In contrast, if each actor shares one elite solution set, only one elite solution set is required and the elite solution set can be renewed up to N_{act} elite solutions for one simulation. Instead of using their own elite solutions set for each actor, using a shared elite solution set not only saves memory resources, but also enhances the renewing boundary violation of the loss function (Eq. 5) while training actors by enhancing the speed updating of a shared elite solution set. The speed improvement of updating the boundary violation may assist in training actors and enhance their design predictions of actors. Thus, for DNN-Opt deploying multiple actors, using a shared elite solution set is expected to improve circuit optimization.

After training actors, each actor θ^{μ_i} propose the change $\Delta\mathbf{x}_{k,i}$ to optimize its input design $\mathbf{x}_k$, and a circuit simulation for the proposed design $\mathbf{x}_k + \Delta\mathbf{x}_{k,i}$ is executed. Thus, using N_{act} actors requires N_{act} actor-network trainings and N_{act} circuit simulations. The runtime for optimization may increase if training actors and circuit simulations are executed sequentially; therefore, parallelism is applied. Training actors and SPICE simulations are implemented over N_{act} CPU cores through multiprocessing.

Algorithm 1 Optimization with Multi-Actors and Critic Networks

1: Initialize actor and critic network parameters θ^{μ_1}, $\theta^{\mu_2} \ldots$, $\theta^{\mu_{N_{act}}} \ldots$ and θ^Q
2: Generate pseudo-samples by using $\mathbf{X}^{tot}$ (Eq. 3)
3: Train the critic network (Eq. 4)
4: Train the actor networks (Eq. 5)
5: Calculate FoM for each design in $\mathbf{X}^{tot}$ (Eq. 2)
6: Choose N_{es} designs in $\mathbf{X}^{tot}$ with smallest FoM to form $\mathbf{X}^{SES}$
7: **for** $i = 1, 2, \ldots N_{act}$ **do**
8: $\quad \mathbf{x}_i^{sample} \leftarrow \underset{\mathbf{x} \in \mathbf{X}^{SES}}{\text{argmin}}\ g[Q(\mathbf{x}, \mu_i(\mathbf{x}))]$ (Eq. 2)
9: $\quad$ Execute SPICE simulation of $\mathbf{x}_i^{sample}$ and obtain the FoM (Eq. 2)
10: $\quad \mathbf{X}^{tot} \leftarrow \mathbf{X}^{tot} \cup \mathbf{x}_i^{sample}$
11: **end for**

C. Near-Sampling Method

We devise a near-sampling method (Fig. 3) to search for a design that is better than the most optimized design $\mathbf{x}_{opt}$ found in previous simulations by exploiting designs adjacent to $\mathbf{x}_{opt}$. This method begins by sampling $N_{samples}$ designs near $\mathbf{x}_{opt}$. The i^{th} design parameter of $\mathbf{x}_{opt}$, $x_{i,opt}$, has a sampling radius δ_i and samples are generated within the range $[x_{i,opt} - \delta_i,\ x_{i,opt} + \delta_i]$ to form a near-sampling set $\mathbf{X}^{NS}$. Subsequently, using a critic network as a regression model of SPICE simulation and the FoM function (Eq. 2), FoMs of $N_{samples}$ designs sampled near $\mathbf{x}_{opt}$ can be predicted. For each sample $\mathbf{x}_{ns}$ in $\mathbf{X}^{NS}$, $(\mathbf{x}_{opt}, \mathbf{x}_{ns} - \mathbf{x}_{opt}) \in \mathbb{D}^{2d}$ is generated and used as an input vector of the critic network. The critic-network prediction $Q(\mathbf{x}_{opt}, \mathbf{x}_{ns} - \mathbf{x}_{opt})$ is applied to the FoM function (Eq. 2), so the FoM of $\mathbf{x}_{ns}$ can be predicted. After predicting the FoM values of the sampled designs, the design predicted as the best solution of $N_{samples}$ samples, $\mathbf{x}_{opt}^{predicted}$, was selected. The SPICE simulation was executed for $\mathbf{x}_{opt}^{predicted}$, and if the FoM of $\mathbf{x}_{opt}^{predicted}$ is better than the FoM of $\mathbf{x}_{opt}$, $\mathbf{x}_{opt}$ is replaced by $\mathbf{x}_{opt}^{predicted}$.

For the successive near-sampling method, compact sampling near $\mathbf{x}_{opt}$ is required to use a critic network as an accurate regression model. For an intensive sampling density, $N_{samples}$ should be large, and δ_i for each design parameter $\mathbf{x}_i$ should be small enough, such that this method does not realize a dramatically optimized design, unlike the optimization using actor-critic training. Thus, the near-sampling method was applied after finding the optimized design satisfying all given performance constraints. After finding the design that meets all constraints, both optimizations with actor-critic training and the near-sampling method were implemented alternatively. Optimization with actor-critic training incurs a significant change in $\mathbf{x}_{opt}$ and can achieve a numerically optimized design with a lower optimization success rate than the near-sampling method. Thus, optimization with actor-critic training can be regarded as exploration, and the near-sampling method serves as exploitation.

D. Overall Proposed Framework

Using the optimization with multi-actors and critic networks (Algorithm 1) and the near-sampling method (Algorithm 2), the overall proposed framework (Algorithm 3) is provided. N_{act} actors were

Algorithm 2 Near-Sampling Method

$\mathbf{x}_{opt} \leftarrow \underset{\mathbf{x}\in\mathbf{X}^{SES}}{\text{argmin}}\ g[f(\mathbf{x})]$ (Eq. 2)
$\mathbf{X}^{NS} \leftarrow \emptyset$
for $N = 1, 2, \ldots N_{samples}$ **do**
  $\mathbf{x}_{ns} \leftarrow \text{UniformDistribution}(\mathbf{x}_{opt} - \delta,\ \mathbf{x}_{opt} + \delta)$
  $\mathbf{X}^{NS} \leftarrow \mathbf{X}^{NS} \cup \mathbf{x}_{ns}$
end for
$\mathbf{x}_{opt}^{predicted} \leftarrow \underset{\mathbf{x}_{ns}\in\mathbf{X}^{NS}}{\text{argmin}}\ g[Q(\mathbf{x}_{opt},\ \mathbf{x}_{ns} - \mathbf{x}_{opt})]$ (Eq. 4)
Execute SPICE simulation of $\mathbf{x}_{opt}^{predicted}$ and obtain the FoM (Eq. 2)
if FoM($\mathbf{x}_{opt}$) > FoM($\mathbf{x}_{opt}^{predicted}$) **then**
  $\mathbf{x}_{opt} \leftarrow \mathbf{x}_{opt}^{predicted}$
end if

Algorithm 3 Overall MA-Opt Framework

Require: Initial set $\mathbf{X}^{init}$ and their evaluations $f(\mathbf{X}^{init})$
$\mathbf{X}^{tot} \leftarrow \mathbf{X}^{init}$
$\mathbf{X}^{SES} \leftarrow \mathbf{X}^{init}$
for $t = 1, 2, \ldots t_{max}$ **do**
  if specs are not met **then**
    Execute the optimization (Algorithm 1)
  else if specs are met **then**
    if (t mod T_{NS}) == k **then**
      Apply the near-sampling method (Algorithm 2)
    else
      Execute the optimization (Algorithm 1)
    end if
  end if
end for
return The design with the lowest FoM

deployed, and training N_{act} actors and N_{act} circuit simulations are executed in parallel (Lines 7–11 for Algorithm 1). In our study, the N_{init} designs were initially sampled, and an initial sample set $\mathbf{X}^{init}$ was defined. The near-sampling method was applied with the period T_{NS} (Line 7 for Algorithm 3).

III. Experiment Results

A. Experiment Settings

A two-stage operational transconductance amplifier (OTA) (Fig. 4a), a three-stage transimpedance amplifier (TIA) (Fig. 4b), and a low-dropout (LDO) regulator (Fig. 4c) were applied to our experiments. The proposed framework was compared to other optimization methods, BO [21], DNN-Opt [16], a framework that adopts multiple actors without using a shared elite solution set and does not apply the near-sampling method (MA-Opt[1]), and a framework that adopts multiple actors while using a shared elite solution set and apply the near-sampling method (MA-Opt[2]). Synopsis HSpice was used for circuit simulation, and the experiments were executed in Intel(R) Xeon(R) Gold 6132 CPUs at a clock frequency of 2.60 GHz. Each circuit was implemented in a commercial 180 nm CMOS technology. To demonstrate each method statistically, we ran each method 10 times, and the number of simulations was limited to 200. For each circuit, to define an initial sample set $\mathbf{X}^{init}$, one hundred designs were randomly sampled and Spice simulations were implemented for the sampled designs, and the same $\mathbf{X}^{init}$ was applied to each optimization method. For the actor and critic networks of DNN-Opt, MA-Opt[1], MA-Opt[2] and MA-Opt, the number of hidden layers was set to two and the number of nodes in each hidden layer was fixed at one hundred. To use the three actors, N_{act} was set to three. While executing the near-sampling method of MA-Opt, T_{NS} was fixed at five and $N_{samples}$ was set to 2000.

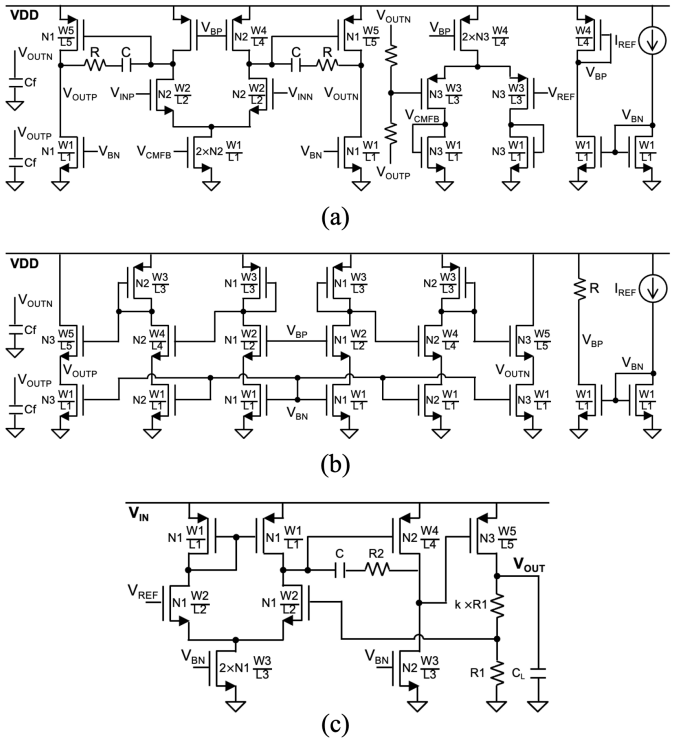

Fig. 4. Schematics of three circuits: (a) two-stage OTA; (b) three-stage TIA; and (c) LDO regulator.

The success rate, minimum target metric, average target metric, average FoM, and total runtime were used to evaluate the optimization methods (TABLE II, TABLE IV, and TABLE VI). The success rate indicates the number of successfully optimized designs that meet all the given circuit-performance constraints within 200 simulations over the all 10 runs, and that evaluates the feasibility of the optimization methods. The minimum target metric was found for the optimized designs that satisfied all constraints within 200 simulations. The average FoM is the mean of the FoM values of designs acquired from all 10 runs, and is indicated on a log scale to compare methods distinctly (Fig. 5). For each optimization method, the total runtime measured in the simulations was averaged. For a fair comparison, by considering the difference in the simulation speed of each optimization method, the average FoM of each method was compared based on the total runtime of DNN-Opt [16].

B. Algorithm Comparison

1) Two-Stage OTA: The two-stage OTA has total 16 design parameters with the ranges (TABLE I). The optimization methods were simulated using the constraints (Eq. 7). The power consumption of the two-stage OTA was used as the target metric. RL-inspired frameworks, DNN-Opt, MA-Opt[1], MA-Opt[2] and MA-Opt achieved better success rates, average FoMs, and lower total runtimes than BO (TABLE II). Additionally, MA-Opt[2] and MA-Opt achieved the highest success rates. The total runtime of DNN-Opt was 55%, 42%, 40% and 24% lower than those of BO, MA-Opt1, MA-Opt2 and MA-Opt, respectively, and that was considered to the comparison for average FoMs of the methods. The average FoMs of MA-Opt[2] and MA-Opt was better than those of DNN-Opt and MA-Opt[1]. Within 200 simulations, MA-Opt realized the optimized design that consumes

TABLE I
TYPES AND RANGES OF DESIGN PARAMETERS FOR A TWO-STAGE OTA

Types of Parameters	Unit	Ranges
L1, L2, L3, L4, and L5	μm	[0.18, 2]
W1, W2, W3, W4, and W5	μm	[0.22, 150]
R	kΩ	[0.1, 100]
C	fF	[100, 2000]
Cf	fF	[100, 10000]
N1, N2, and N3	integer	[1, 20]

TABLE II
ALGORITHM COMPARISON FOR A TWO-STAGE OTA

Algorithm	BO	DNN-Opt	MA-Opt^1	MA-Opt^2	**MA-Opt**
Success rate	0/10	8/10	7/10	**10/10**	**10/10**
Min power (mW)	-	0.852	0.994	1.097	**0.737**
log_{10}(average FoM)	-0.04	-2.05	-1.25	-2.75	$\mathbf{-2.92}$
Total runtime (h)	1.54	**0.69**	1.19	1.15	0.91

lowest power and satisfies all constraints, and the minimum power obtained in MA-Opt was 13% lower than that of DNN-Opt.

$$\begin{aligned}&\text{Minimize Power}\\&\text{s.t.}\quad \text{DC Gain} > 60\text{ dB} \qquad \text{Settling Time} < 100\text{ ns}\\&\qquad \text{CMRR} > 80\text{ dB} \qquad \text{Unity Gain Freq.} > 30\text{ MHz}\\&\qquad \text{PSRR} > 80\text{ dB} \qquad \text{Out. Swing} > 1.5\text{ V}\\&\qquad \text{Phase Margin} > 60\text{ deg} \qquad \text{Out. Noise} < 30\text{ mV}_{rms}.\end{aligned} \tag{7}$$

2) Three-Stage Transimpedance Amplifier: The three-stage TIA has total 15 design parameters with the ranges (TABLE III). The optimization methods were simulated using the constraints (Eq. 8). The power consumption of the three-stage TIA was defined as the target metric. RL-inspired frameworks achieved higher success rates, average FoMs, and lower total runtimes than BO (TABLE IV). Additionally, MA-Opt^2 and MA-Opt achieved the highest success rates. The total runtime of DNN-Opt was 67%, 54%, 48% and 41% lower than those of BO, MA-Opt1, MA-Opt2 and MA-Opt, respectively, and that was reflected to the comparison for average FoMs of the methods. The average FoMs of MA-Opt^2 and MA-Opt were better than those of DNN-Opt and MA-Opt^1. Within 200 simulations, MA-Opt realized the design with the minimum power consumption while satisfying all constraints, and the minimum power achieved in MA-Opt was 24% lower than that of DNN-Opt.

$$\begin{aligned}&\text{Minimize Power}\\&\text{s.t.}\quad \text{DC Gain} > 80\text{ dB}\\&\qquad \text{Unity Gain Freq.} > 1\text{ GHz}\\&\qquad \text{Input Referred Noise} < 10\sqrt{\text{pA}}/\text{Hz}.\end{aligned} \tag{8}$$

3) LDO Regulator: A 3.3 V to 1.8 V LDO regulator was used to our experiment. The LDO regulator has total 16 design parameters with the ranges (TABLE V). The optimization methods were simulated using the constraints (Eq. 9). The quiescent current of the LDO regulator with a load current of 50 mA was used as the target metric. $V_{OUT,\,V_{IN}=3.3\text{ V}}$ is the output voltage V_{OUT} when the input voltage V_{IN} is 3.3 V. $T_{L,\,\text{I1}\rightarrow\text{I2}}$ is the settling time of V_{OUT} when V_{IN} is 3.3 V and the load current I_{LOAD} changes from I1 to I2. $T_{V,\,\text{V1}\rightarrow\text{V2}}$ is the settling time of V_{OUT} when I_{LOAD} is 50 mA and V_{IN} changes from V1 to V2. RL-inspired methods achieved higher success rates, average FoMs, and lower total runtimes than BO (TABLE VI). Also, MA-Opt^2 and MA-Opt showed the highest success rates. The total runtime of DNN-Opt was 52%, 40%, 44% and 34% lower than those of BO, MA-Opt1, MA-Opt2 and MA-Opt, respectively, and that was reflected to the comparison for average FoMs of the methods. The average FoMs of MA-Opt^2 and MA-Opt are better than that of DNN-Opt and MA-Opt^1. Within 200 simulations, MA-Opt realized the design

TABLE III
TYPES AND RANGES OF DESIGN PARAMETERS FOR A THREE-STAGE TIA

Types of Parameters	Unit	Ranges
L1, L2, L3, L4, and L5	μm	[0.18, 2]
W1, W2, W3, W4, and W5	μm	[0.22, 150]
R	kΩ	[0.1, 100]
Cf	fF	[100, 2000]
N1, N2, and N3	integer	[1, 20]

TABLE IV
ALGORITHM COMPARISON FOR A THREE-STAGE TIA

Algorithm	BO	DNN-Opt	MA-Opt^1	MA-Opt^2	**MA-Opt**
Success rate	0/10	4/10	2/10	**10/10**	**10/10**
Min power (mW)	-	0.196	-	0.190	**0.148**
log_{10}(average FoM)	-0.01	-1.04	-0.76	-3.43	$\mathbf{-3.50}$
Total runtime (h)	1.38	**0.46**	1.02	0.90	0.78

that has minimum quiescent current and meets all constraints, and the minimum quiescent current with a load current of 50 mA in MA-Opt was 17% lower than that of DNN-Opt.

$$\begin{aligned}&\text{Minimize Quiescent Current } (I_{LOAD} = 50\text{ mA})\\&\text{s.t.}\quad V_{OUT,\,V_{IN}=3.3\text{ V}} > 1.75\text{ V} \qquad V_{OUT,\,V_{IN}=3.3\text{ V}} < 1.85\text{ V}\\&\qquad \text{Load Regulation} < 0.1\text{ mV/mA} \qquad \text{Line Regulation} < 0.1\text{ \%/V}\\&\qquad T_{L,\,0.1\,\mu\text{A}\rightarrow150\text{ mA}} < 35\ \mu\text{s} \qquad T_{L,\,150\text{ mA}\rightarrow0.1\,\mu\text{A}} < 35\ \mu\text{s}\\&\qquad T_{V,2.0\text{ V}\rightarrow3.3\text{ V}} < 35\ \mu\text{s} \qquad T_{V,3.3\text{ V}\rightarrow2.0\text{ V}} < 35\ \mu\text{s}\\&\qquad \text{PSRR} > 60\text{ dB}.\end{aligned} \tag{9}$$

C. Experiment Analysis

For the three circuits, the RL-inspired methods achieved higher success rates and lower total runtime than BO (TABLE II, TABLE IV, and TABLE VI). Moreover, the RL-inspired methods achieve better results in terms of target metrics and FoMs. Thus, RL-inspired methods can achieve better optimization capabilities than BO.

MA-Opt^2 and MA-Opt achieved higher success rates than DNN-Opt and MA-Opt^1 within 200 simulations. In addition, because MA-Opt^1, MA-Opt^2 and MA-Opt require more runtimes than DNN-Opt owing to the context switching of multiprocessing, the average FoM of each methods was compared considering differences of simulation speed for a fair comparison. The average FoMs of MA-Opt^1 and MA-Opt^2 were better than that of DNN-Opt, so MA-Opt^2 and MA-Opt tended to find more optimized designs than DNN-Opt and MA-Opt^1 at the same runtime, and that proves the strength of adopting multiple actors with a shared elite solution set.

MA-Opt produced the most optimized designs within 200 simulations and the lowest average FoMs, demonstrating the advantage of the near-sampling method. The effects of the near-sampling method can be explained from two perspectives. First, the near-sampling method can find a better design adjacent to $\mathbf{x}_{opt}$ with a high success rate. Second, the shared elite solution set updated by the near-sampling method can change boundary violations in the loss function of actor-networks (Eq. 5), which may assist the optimization with actor-critic networks. Furthermore, the total runtime of MA-Opt was less than that of MA-Opt^2. When sampling many samples for predictions, the near-sampling method requires less runtime than the training actor or critic networks; therefore, the runtime of the near-sampling method is less than that of optimization with multiple actors and critic networks. MA-Opt alternatively uses the near-sampling method and optimization with multiple actors and critic networks; thus, within the same number of simulations, MA-Opt has a less runtime than MA-Opt^2. Therefore, MA-Opt can achieve a better optimized design and requires a shorter total runtime than MA-Opt^2.

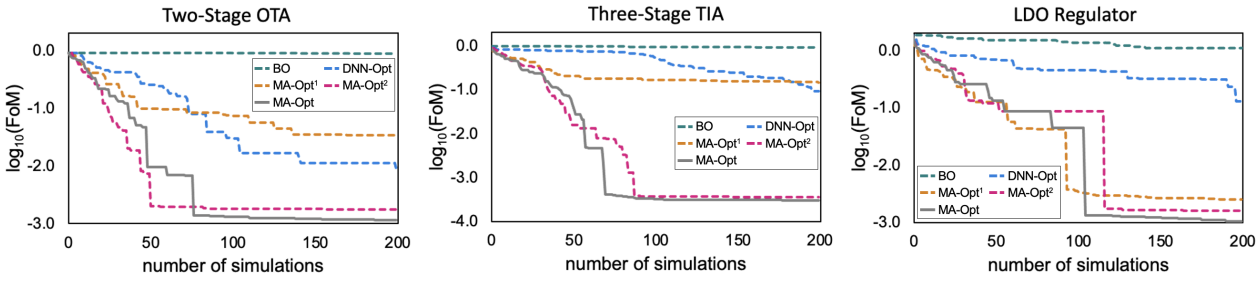

Fig. 5. Average FoMs of three optimization cases: (a) two stage OTA; (b) three stage TIA; and (c) LDO regulator.

TABLE V
TYPES AND RANGES OF DESIGN PARAMETERS FOR AN LDO REGULATOR

Types of Parameters	Unit	Ranges
L1, L2, L3, L4, and L5	μm	[0.32, 3]
W1, W2, W3, W4, and W5	μm	[0.22, 200]
R1, R2	kΩ	[1, 100]
C	fF	[100, 2000]
N1, N2, and N3	integer	[1, 20]

TABLE VI
ALGORITHM COMPARISON FOR AN LDO REGULATOR

Algorithm	BO	DNN-Opt	MA-Opt^1	MA-Opt^2	**MA-Opt**
Success rate	0/10	7/10	9/10	**10/10**	**10/10**
Min Q.C. (mA)	-	0.320	0.335	0.382	**0.265**
log_{10}(average FoM)	0.04	-0.88	-2.59	-2.79	$\mathbf{-2.98}$
Total runtime (h)	1.57	**0.75**	1.26	1.35	1.14

MA-Opt obtained better circuit designs than DNN-Opt for the same number of simulations. In our experiments, MA-Opt obtained minimum target metrics of optimized designs up to 24% better than DNN-Opt. Moreover, the average FoM of MA-Opt is better than that of DNN-Opt; therefore, MA-Opt finds more optimized designs than DNN-Opt at the same runtime. Thus, MA-Opt has a better optimization capability than DNN-Opt.

IV. CONCLUSION

To improve the existing RL-inspired framework, we proposed MA-Opt, a novel RL-inspired framework using multiple actors sharing an elite solution set and a near-sampling method. To evaluate the proposed framework, it was applied to three analog circuits, and then its performance was compared with that of other methods. Optimization with actor-critic training acts as exploration, and the near-sampling method serves as exploitation. The experimental results revealed the efficiency of using multiple actors with a shared elite solution set and near-sampling method. Within the same number of simulations, while satisfying all given constraints, MA-Opt produced minimum target metrics up to 24% better than those of DNN-Opt. Furthermore, MA-Opt realized better Figure of Merits than those of DNN-Opt within the same runtime. Therefore, MA-Opt achieved a better capability of circuit optimization than DNN-Opt.

ACKNOWLEDGMENT

This work was supported by DRAM PIM Design Base Technology Development (No.2022-0-01172) and Software Systems for AI Semiconductor Design (No.2021-0-00754) through Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT). The EDA Tool was supported by the IC Design Education Center (IDEC).

REFERENCES

[1] N. Horta, "Analogue and mixed-signal systems topologies exploration using symbolic methods," Analog Integrated Circuits and Signal Processing, 2002.
[2] N. Jangkrajarng *et al.*, "Iprail—intellectual property reuse-based analog ic layout automation," Integration, 2003, analog and Mixed-signal IC Design and Design Methodologies.
[3] W. Daems *et al.*, "Simulation-based generation of polynomial performance models for the sizing of analog integrated circuits," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), 2003.
[4] M. d. Hershenson *et al.*, "Optimal design of a cmos op-amp via geometric programming," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), 2001.
[5] Y. Wang *et al.*, "Enabling efficient analog synthesis by coupling sparse regression and polynomial optimization," 2014 51th ACM/IEEE Design Automation Conference (DAC), 2014.
[6] S. P. Boyd *et al.*, "Geometric programming for circuit optimization," roceedings of International Symposium on Physical Design (ISPD), 2005.
[7] R. Acar Vural *et al.*, "Analog circuit sizing via swarm intelligence," AEU - International Journal of Electronics and Communications, 2012.
[8] B. Liu *et al.*, Automated Design of Analog and High-frequency Circuits: A Computational Intelligence Approach. Springer, 2013.
[9] W. Lyu *et al.*, "Batch Bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design," 2018 35th International Conference on Machine Learning (ICML), 2018.
[10] W. Lyu *et al.*, "Multi-objective bayesian optimization for analog/rf circuit synthesis," 2018 55th ACM/IEEE Design Automation Conference (DAC), 2018.
[11] S. Zhang *et al.*, "An efficient multi-fidelity Bayesian optimization approach for analog circuit synthesis," 2019 56th ACM/IEEE Design Automation Conference (DAC), 2019.
[12] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2016 33th International Conference on Machine Learning (ICML), 2016.
[13] K. Settaluri *et al.*, "Autockt: Deep reinforcement learning of analog circuit designs," 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2020.
[14] H. Wang *et al.*, "Gcn-rl circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning," 2020 57th ACM/IEEE Design Automation Conference (DAC), 2020.
[15] K. Somayaji *et al.*, "Prioritized reinforcement learning for analog circuit optimization with design knwoledge," 2021 58th ACM/IEEE Design Automation Conference (DAC), 2021.
[16] A. Budak *et al.*, "Dnn-opt: An RL inspired optimization for analog circuit sizing using deep neural networks," in 2021 58th ACM/IEEE Design Automation Conference (DAC), 2021.
[17] V. Konda *et al.*, "Actor-critic algorithms," in SIAM Journal on Control and Optimization. MIT Press, 2000.
[18] A. Nair *et al.*, "Massively parallel methods for deep reinforcement learning," in CoRR, 2015.
[19] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," 2016 33th International Conference on Machine Learning (ICML), 2016.
[20] R. Turner *et al.*, "Bayesmark," https://github.com/uber/ bayesmark, 2020.
[21] J. Snoek *et al.*, "Practical bayesian optimization of machine learning algorithms," Neural Information Processing Systems (NIPS), 2012.