Using Probabilistic Model Rollouts to Boost the Sample Efficiency of Reinforcement Learning for Automated Analog Circuit Sizing

Mohsen Ahmadzadeh, Georges G.E. Gielen
ESAT-MICAS, KU Leuven, 3001 Leuven, Belgium
{mohsen.ahmadzadeh, georges.gielen}@kuleuven.be

ABSTRACT

Despite recent advances in algorithms, such as the use of reinforcement learning, analog circuit sizing optimization remains a challenging task that demands numerous circuit simulations, hence extensive CPU times. This paper introduces the application of Model-Based Policy Optimization (MBPO) to highly boost the sample efficiency of reinforcement learning for analog circuit sizing. This method leverages an ensemble of probabilistic dynamic models to generate short rollouts branched from real data for a fast but extensive exploration of the design space, thereby speeding up the learning process of the reinforcement learning agent and improving its convergence. Integrated in the Twin Delayed DDPG (TD3) algorithm, our new model-based TD3 (MBTD3) approach is validated on analog circuits of different complexity, outperforming the existing model-free TD3 method by achieving power/area-optimal design solutions within up to ∼3x fewer simulations and half the run time. In addition, for larger analog circuits, we present a multi-agent version of MBTD3, in which multiple simultaneous agents use global probabilistic models for sizing the different sub-blocks within the circuit. Demonstrated for a complex data receiver circuit, it surpasses the model-free multi-agent TD3 method with ∼2x less simulations and half the run time. The proposed novel algorithms clearly boost the efficiency of automated analog circuit sizing.

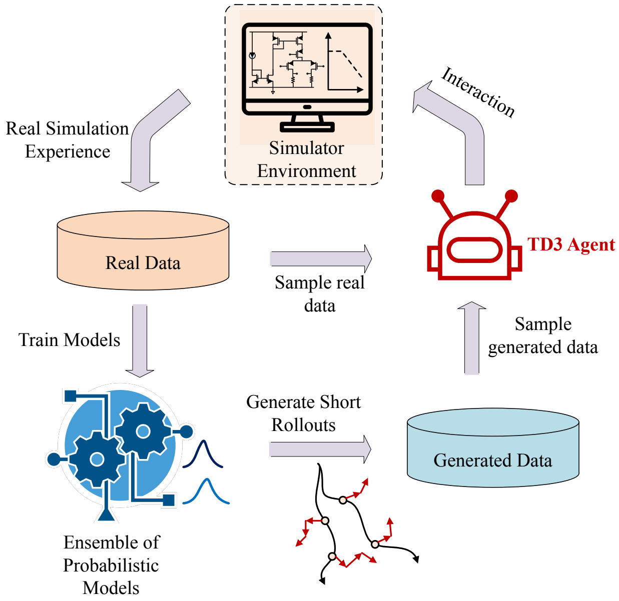

Figure 1: TD3-based MBPO for circuit sizing

KEYWORDS

Circuit design automation, Model based policy optimization, Twin delayed deep deterministic policy gradient, Analog circuit sizing

ACM Reference Format:
Mohsen Ahmadzadeh, Georges G.E. Gielen. 2024. Using Probabilistic Model Rollouts to Boost the Sample Efficiency of Reinforcement Learning for Automated Analog Circuit Sizing. In *61st ACM/IEEE Design Automation Conference (DAC '24), June 23–27, 2024, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3649329.3657335

1 INTRODUCTION

Analog/mixed-signal circuits are an essential part of any electronic system that requires interaction with the outside physical world. Applications include internet of things (IoT), biomedical, imaging, telecom systems and many more. While the core of most electronic systems consists of digital circuits, the analog part, despite occupying a minor fraction of the chip area, is still a main bottleneck in design efforts due to the absence of efficient and widely adopted automation tools [1]. Within the analog design cycle, circuit sizing is notably time-consuming and labor-intensive because of its high dimensionality of the design space. Many simulation-based methods have been proposed in recent decades [2] to address this, using black-box optimization tools such as Genetic Algorithms (GA) [3, 4] and Bayesian Optimization (BO) [5, 6]. However, for complex problems, GAs demand a vast number of simulations and offer no guaranteed convergence due to their inherent stochasticity [7]. Similarly, the Gaussian Processes (GPs) used in BO for design space modeling suffer from cubic computational scaling, making them computation and runtime expensive with increasing circuit dimensionality. [8, 9]. Therefore, these algorithms are mostly useful for circuit problems of less than around 20 dimensions [9].

Recent approaches in analog circuit sizing automation employ Reinforcement Learning (RL) or RL-inspired algorithms, and have been shown to be more sample-efficient and less time consuming [7, 10–18]. Particularly, actor-critic RL methods like Deep Deterministic Policy Gradients (DDPG) have been shown to be efficient in sizing analog circuits [10–12, 14–16]. Especially the Twin Delayed DDPG (TD3) of [16] is a robust actor-critic agent for circuit sizing. That work has also adopted a multi-agent RL (MARL) approach for sizing complex circuits comprised of several sub-blocks.

DAC '24, June 23–27, 2024, San Francisco, CA, USA
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0601-1/24/06
https://doi.org/10.1145/3649329.3657335

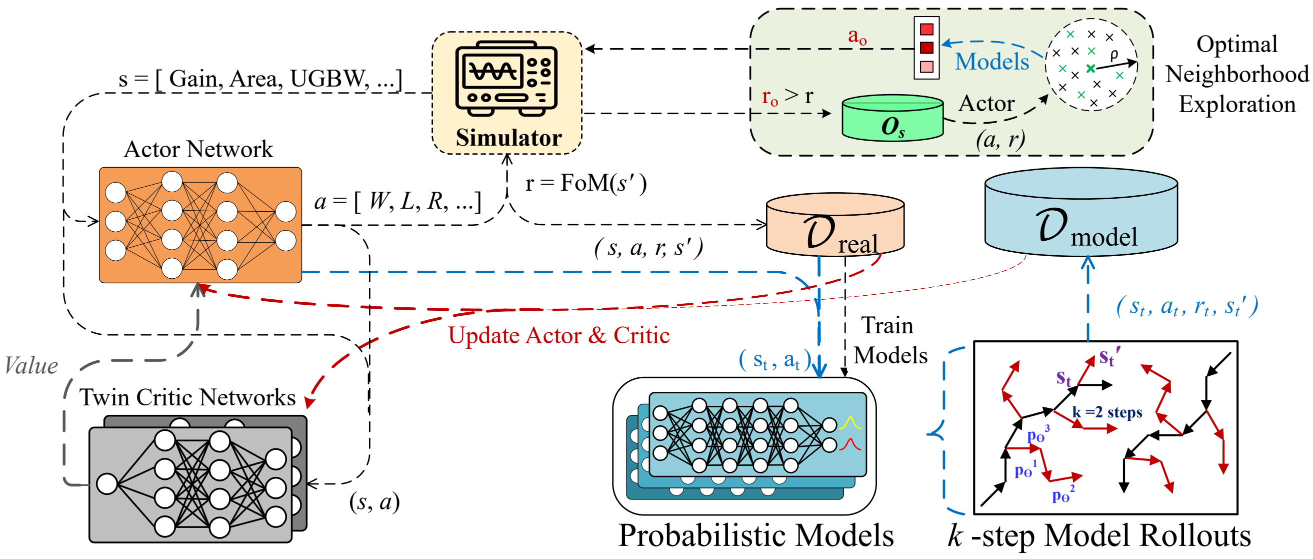

Figure 2: Detailed diagram of the MBTD3 algorithm for analog circuit sizing.

Despite their advances, these methods still require a large number of simulations, which hinders scaling them to truly complex circuits. The difficulty of data collection in complex physical systems has also led RL researchers to devise advanced sample-efficient model-based methods [19, 20]. Such methods can explore the design space faster by generating short rollouts using environment models that are locally trained and updated during the learning process. The main challenge is that model accuracy becomes a bottleneck to the quality of solutions by gradually injecting destructive bias into the policy. To solve this problem, MBPO [20] has shown significant sample efficiency through two key strategies: 1) using an ensemble of probabilistic dynamic models for addressing both aleatoric and epistemic uncertainties, and 2) generating numerous short trajectories branched from randomly selected past experiences and using them in a combination with real data for updating the actor and critic networks at each step. These techniques have helped MBPO in mitigating the usual pitfalls of model bias injection, leading to similar monotonic improvements as in model-free methods. In this paper, we will demonstrate that a TD3-based MBPO (Figure 1) is a powerful RL agent in sizing analog circuits. We will also show the applicability of model-based TD3 in multi-agent schemes.

The main contributions and features of this paper are therefore:

(1) We develop a TD3-based MBPO agent. This model-based TD3 (MBTD3) extracts several short trajectory rollouts using an ensemble of performance models at each step. A combination of such model-generated transitions and real simulated data is used for updating the agent's neural networks at each environment step, which leads to a considerable speedup of the learning process.

(2) We also exploit the probabilistic performance models for improved exploration of the solution space and find better design solutions.

(3) We develop a multi-agent variant of the MBTD3 (MA-MBTD3) and leverage it for sizing a complex data receiver circuit.

(4) We experimentally demonstrate that MBTD3 outperforms model-free TD3 in sizing two operational amplifiers with different levels of complexity by reaching more optimal solutions in up to approximately three times fewer simulations and two times lower runtime. We also show that MA-MBTD3 reaches optimal solutions for a complex data receiver circuit with ~2x lower simulation count and runtime.

The paper is organized as follows. In Section II, we will explain the detailed structure of the MBTD3 framework and its multi-agent variant. In section III, we will demonstrate the experimental results of both MBTD3 and MA-MBTD3. Finally, we will conclude the paper in section IV.

2 METHODOLOGY

A. Reinforcement Learning Formulation of Circuit Sizing

To formulate our Reinforcement Learning framework, we need three key elements: 1) the objective Figure-of-Merit (FoM) for the reward signal, 2) the features defining the state space, and 3) the design parameters to be tuned as the actions:

Reward. We use the following FoM as our reward signal (r), which is inspired by [7] with some small modifications:

$$z_y = \frac{y - y^*}{y + y^*} \rightarrow r_y = \begin{cases} \min(z_y, 0), & y \in Y_L \\ -\max(z_y, 0), & y \in Y_U \end{cases} \rightarrow r_H = \sum_{y \in Y} r_y$$
$$o_t = \frac{t - t^*}{t + t^*} \rightarrow r_t = o_t \rightarrow r_T = \sum_{t \in T} r_t \qquad (1)$$
$$FoM = \begin{cases} r_H - \alpha \times r_T, & \text{if } r_H < 0 \\ 0.3 - \beta \times r_T, & \text{if } r_H = 0 \end{cases}$$

where Y is the set of all hard constrained specifications, whether with lower boundaries Y_L or upper boundaries Y_U. T is the set of optimization targets, r_H is the total reward from all hard specifications, r_T is the total reward from all optimization targets, α is a small value (0.05 in our experiments) and β scales the influence of the optimization targets on the success rewards (1.0 in our experiments). The y^* and t^* terms set objective thresholds for hard constraints and normalization values for the optimization targets, respectively.

State Space. The state of the circuit at each step consists of the list of the normalized terms of all specifications (both hard constraints and optimization targets):

$$s = [z_1, ..., z_Y, o_1, ..., o_T] \tag{2}$$

Action Space. The action at each step consists of assigning a value to all the design parameters that need to be tuned (sized). These involve the width (W), length (L), and multiplier (m) for each transistor; as well as the resistance (R), the capacitance (C), and any biasing voltages (V_i) in the circuit:

$$a = [W_1, L_1, m_1, ..., W_n, L_n, m_n, R_1, R_2, ..., C_1, C_2, ..., V_1, V2, ...] \tag{3}$$

The predicted actions by the RL agent are in the range of [-1, 1] and must be refined to the real range of the design parameters for all simulations.

B. MBTD3 Framework

In MBPO, probabilistic neural networks are used as performance models. These neural networks are designed to learn and provide the mean and the variance of a Gaussian distribution for each output during training. As a result, they produce a distribution of possible outputs, rather than a single deterministic output, reflecting the uncertainty. Using an ensemble of these models (n_m models) for data generation can further mitigate epistemic uncertainty. In our MBTD3 framework, as shown in Figure 2, we utilize these models for both accelerating the policy optimization and also exploring around the current optimal solutions to find potential better candidates. Each model is expected to receive the current state and the selected action (s, a) as input, and outputs the mean and variance of a predicted next state s' [19, 20]:

$$p_\theta^i(s'|s, a) = \mathcal{N}(\mu_\theta^i(s, a), \Sigma_\theta^i(s, a)) \tag{4}$$

in which the mean μ_θ and the diagonal matrix of covariances Σ_θ are parameterized by θ as they are outputs of the neural network. Assuming a batch of N_b real transitions, if (s_n, a_n) is the state-action pair of each transition and $s_{n'}$ is its true next state, then the loss function used for training such networks is:

$$\begin{aligned} L_p(\theta) = \sum_{n=1}^{N_b} \Bigg[& \left[\mu_\theta(s_n, a_n) - s'_n\right]^T \Sigma_\theta^{-1}(s_n, a_n) \left[\mu_\theta(s_n, a_n) - s'_n\right] \\ & + \log\det \Sigma_\theta(s_n, a_n) \Bigg] \end{aligned} \tag{5}$$

In the first term of this loss function, the mean errors are weighted inversely by the variances to penalize confident errors more. In the second term, the determinant of the variances is considered to discourage general large variances, promoting predictions that are both accurate and confident.

Algorithm 1: MBTD3 for circuit sizing

```
Initialize actor network $\pi(s|\theta^\pi)$, two critics networks $Q_i(s, a|\theta_i^Q)$, $n_m$
  predictive models $p_\theta$, empty real buffer $D_{real}$ and model buffer $D_{model}$,
  Optimal Solutions Storage $O_s \leftarrow \emptyset, *$
for $t = 1$ to $T$ do
    Select action $a$ using actor $\pi(s|\theta^\pi)$ and exploration noise
    Simulate this action, obtain next state $s'$ and reward $r$
    Store this transition $(s, a, s', r)$ in $D_{real}$
    if size$(D_{real}) > w$ then
        if $t \mod U = 0$ then
            Train/update models $p_\theta$ on $D_{real}$ with the loss $L_p(\theta)$ in Eq. 5
        if $t \mod R = 0$ then
            for $M$ model rollouts do
                Sample $s_t$ uniformly at random from $D_{real}$
                Choose $e$ least uncertain models from the $n_m$ models
                for $k$-steps do
                    Sample action $a_t$ using the actor
                    Input $(s_t, a_t)$ into models; obtain means and variances
                    Randomly select one of the $e$ least uncertain models
                    Obtain next state $s'_t$ using that model, get reward $r$
                    Add this transition $(s_t, a_t, r, s'_t)$ to $D_{model}$
                    Consider $s'_t$ as the new $s_t$
                Pick $\epsilon B$ samples from $D_{model}$ and $(1 - \epsilon)B$ from $D_{real}$
                Obtain target action; $a'_k \leftarrow \mu(s'|\theta^\mu)$+ exploration noise
                Compute target values; $V_i = Q_i(s', a'|\theta^{Q_i}), i = 1, 2$
                Compute the TD3 target value; $V = r + \gamma. \min(V_1, V_2)$
                Update the critic networks by minimizing loss functions:
                  $L(\theta_i^Q) \leftarrow \frac{1}{R} \sum (V_i - V)^2, i = 1, 2$
        if $t \mod d = 0$ then
            Update actor network by minimizing loss function:
              $L(\theta^\pi) \leftarrow -\frac{1}{B} \sum V_1.\nabla_{\theta^\pi} \mu(s|\theta^\pi)$
        Update and reorganize the Optimal Solutions Storage $O_s$
        if $t \mod \psi = 0$ and $O_s \neq \emptyset$ then
            Run the Optimal Neighborhood Exploration of Algorithm 2
// * $\epsilon$: portion of model-generated data in the updating batch, $e$: number of
  least uncertain models, $d$: frequency of updating the actor network, $\psi$:
  frequency of using Optimal Neighborhood Exploration
```

Algorithm 2: Optimal Neighborhood Exploration

```
for each optimal solution $(a, r)$ in $O_s$ do
    $C \leftarrow \emptyset$
    for $S$ samples do
        $a_c \leftarrow$ uniformly select random action within $(a - \rho, a + \rho)$
        Input $a_c$ to all models and obtain collective average results; average $\mu_c$
          and average uncertainty $\sigma_c \longrightarrow$ reward $r_c$
        if $r_c > r$ then
            Add $(a_c, \sigma_c, r_c)$ to $C$
    $X \leftarrow$ Select top $n$ candidates with highest rewards from $C$
    Find the candidate $a_o$ with least uncertainty $\sigma_o$ from $X$
    Simulate $a_o$ and obtain real reward $r_o$
    if $r_o > r$ then
        Add $a_o$ to $O_s$
```

In TD3, after w warmup steps (~100-200), the Actor and Critic networks are updated using a random batch from the previous data stored in the replay buffer $\mathcal{D}_{real}$. In MBTD3, models are trained/updated after each U steps. Then, after each R steps, these models are used to roll out M short k-step trajectories starting from randomly selected previous transitions. During each step of these k-steps, one of the models with less uncertainty is selected at random, allowing for different transitions along a single rollout to be sampled from different dynamics models. These model-generated transitions are stored in another replay buffer $\mathcal{D}_{model}$. Then, at each policy optimization step, a combination of these real and generated transitions is selected as the batch (B) of data for updating the Actor and Critic networks. In other words, for better handling of the bias of the models, at each policy optimization step, MBTD3 selects a portion of the batch from generated data ($\mathcal{D}_{model}$), and the remaining portion is chosen from real data ($\mathcal{D}_{real}$). This pessimistic approach in using generated trajectories, however, significantly assists the agent in faster observing unexplored spaces and therefore speeds up the policy optimization

Algorithm 3: MA-MBTD3 for complex circuits

```
Initialize actor and critic networks of $N$ agents, global probabilistic models $P_\theta$,
  buffers $D_{real}$ and $D_{model}$, optimal storage $O_s \leftarrow \emptyset$
Perform $W$ warmup steps to initialize $D_{real}$
for $t = 1$ to $T$ do
    if $t \mod U = 0$ then
        Train/Update global models $P_\theta$ on $D_{real}$
        Take actions $(a_1, a_2, \ldots, a_N)$ using $N$ agent actors
        Simulate all actions and obtain next state
        Obtain reward $r = (r_1, r_2, \ldots, r_N)$; add transition to $D_{real}$
    if $t \mod R = 0$ then
        for $M$ model rollouts do
            sample $s_t$ uniformly at random from $D_{real}$
            for $k$-steps do
                Take action $a_t = (a_{1,t}, a_{2,t}, \ldots, a_{N,t})$ using $N$ agents
                Input $(s_t, a_t)$ to $P_\theta$; obtain $s'_t$ and $r_t = (r_{1,t}, r_{2,t}, \ldots, r_{N,t})$
                Add this transition to $D_{model}$ and $s_t = s'_t$
    Update actor and critic networks of all $N$ agents using $\epsilon B$ samples from
      $D_{model}$ and $(1 - \epsilon)B$ samples from $D_{real}$
    Update and reorganize $O_s$; Run Algorithm 2 every $\psi$ steps
```

[20]. Algorithm 1 explains the different steps of MBTD3; *U*, *R*, *M*, and k are hyper-parameters to be tuned in this algorithm. In particular, we gradually increase the length of the rollouts (k) from 1 step to 5 steps during the learning process. We train and update an ensemble of seven ($n_m = 7$) neural networks (each having 4 hidden layers of 100 neurons) and pick the five least uncertain models ($e = 5$) for rollout generation.

Optimal Neighborhood Exploration. To optimize the solution space exploration, as shown in Algorithm 2 and Figure 2, we leverage performance models to identify promising candidates near the current optimal solutions, incorporating the uncertainty representation to prioritize high-reward, low-uncertainty options. This algorithm is part of the MBTD3 framework and selects N points within a radius ρ of each optimal solution, evaluates them through the probabilistic models, and chooses the top n candidates with predicted rewards higher than that of the original solution. From these, MBTD3 picks the candidate with minimal average uncertainty, simulates it, and integrates any successful outcomes into the optimal solution storage (O_s). This process is executed every ψ steps. In our experiments, *N*, *n*, and ψ are set to 100, 5, and 100, respectively.

C. Multi-Agent MBTD3

For sizing complex circuits consisting of several sub-blocks, we have implemented a multi-agent version of model-based TD3 (MA-MBTD3). A complex analog circuit is divided into smaller sub-blocks using topology and design knowledge, with key nodes creating boundaries for relatively independent current and voltage behaviors [16]. We assign each sub-block to a separate TD3 agent like in MATD3 [16]. The reward engineering is similar to Eq. (1) for each agent. We use global probabilistic models for all sub-blocks that try to mimic the simulator and predict an estimate of the outputs of all sub-blocks. All different agents have access to these global models when generating rollouts. Algorithm 3 and Figure 3 demonstrate our MA-MBTD3 method for sizing complex circuits. For the reward of each block, we consider its own reward (r^i) added to a second reward term that comes from the general specifications of the entire circuit weighed by a coefficient γ (0.3 in our experiments): $R^i = \gamma R_G + r^i$. The total FoM is then the sum of all R^i terms ($R_T = \sum R^i$).

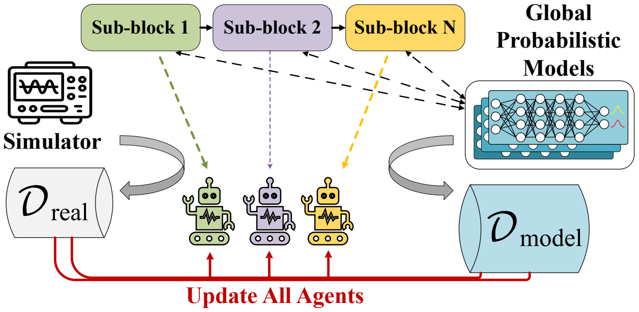

Figure 3: Multi-agent MBTD3 for sizing a complex circuit.

3 EXPERIMENTAL RESULTS

A. Sizing Operational Amplifiers

Our experiments are done in the 45*nm* BSIM predictive technology [21] with $V_{DD} = 1.2V$. For the area calculation, we consider an estimate of the active area for each set of actions (no layout considerations). We run the algorithms on a server with Intel® Xeon® Silver 4110 CPU and Nvidia GeForce RTX 2080ti GPU. We first show the efficacy and efficiency of our MBTD3 algorithm on two amplifier circuits: a basic two-stage opamp (Figure 4-a), and a folded-cascode opamp with common-mode feedback (CMFB) (Figure 5-a). We compare MBTD3 with the baseline model-free TD3 agent as well as with the NSGA-II genetic algorithm used in [4]. We limit these experiments to 7 hours.

For the two operational amplifier circuits, we have considered six hard specification constraints; Gain, Phase Margin, Unity Gain Bandwidth, Slew Rate, Settling Time, and Output Noise; and we aim for the minimization of the total current (or power consumption) and the area. We search for the optimum transistor sizing parameters (*W*, *L*, *m*), resistor values (*R*) and capacitor values (*C*) in both circuits and, for the folded-cascode opamp, also for some biasing voltages. The ranges of the different design variables are; *W*=[0.25, 5] μm, *L*=[45, 225] *nm*, *m*=[1, 25], *C*=[0.1, 10] *pF*, Miller resistor $R_c = [100, 10k]\Omega$, CMFB resistor $R = [1k, 1M]\ \Omega$, and biasing voltages $V_i = [0, 1.2]\ V$. The reference current *ibias* is 30 μA, and the capacitive loads C_L are 10 *pF*. The basic two-stage opamp is a simpler circuit with 20 optimization parameters. The folded-cascode opamp has a more complicated structure with 48 optimization parameters.

Table 1 shows a summary of the results obtained by all the algorithms: the values reached for the constrained performances, the

Table 1: Performance comparison for the opamps

Performance	two-stage opamp			folded-cascode opamp		
	target	TD3	MBTD3	Target	TD3	MBTD3
Gain	200	221	237	350	417	455
Phase Margin (°)	60	78	75	60	64	67
Unity GBW (*MHz*)	1	19	24	10	178	186
Slew Rate ($V/\mu sec$)	15	28	26	15	18.2	24.2
Settling Time (μsec)	3	2.45	1.86	2	1.79	1.27
Out. Noise (mV_{rms})	10	7.4	5.6	30	26.7	23.6
Total Current (*mA*)	-	0.107	0.103	-	1.52	1.48
Area ($\times 10^{-3} mm^2$)	-	1.081	1.01	-	22.15	21.56
FoM	-	0.78	**0.9**	-	0.79	**0.81**
# Simulations	-	2926	**1452**	-	4283	**1417**
Runtime (*hours*)	-	2.24	**1.65**	-	3.57	**1.74**

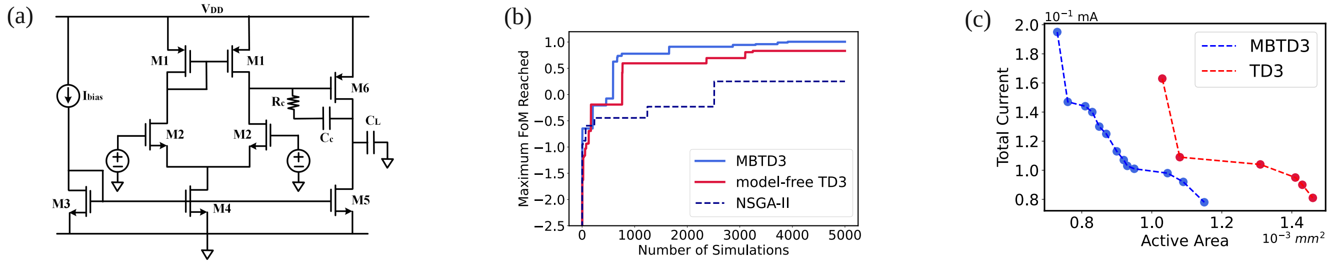

Figure 4: a) Schematic of the two-stage opamp, b) learning curves of algorithms, c) comparison of the optimal solution fronts

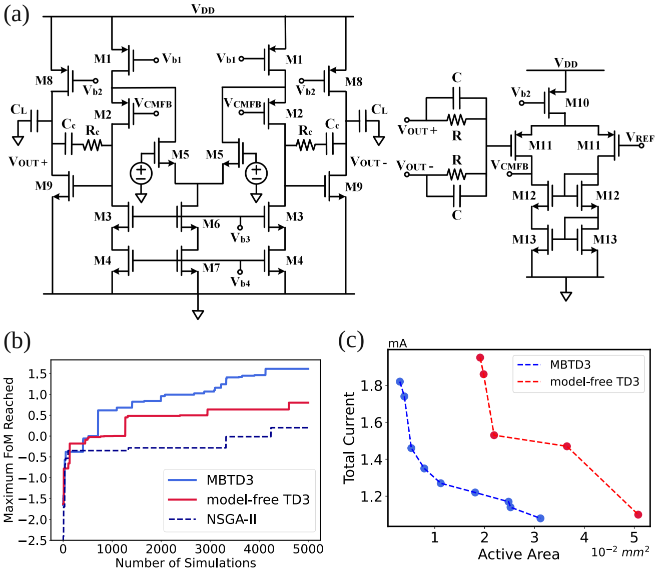

Figure 5: a) Schematic of the folded-cascode opamp, b) learning curves, c) comparison of the optimal solution fronts

achieved optimization targets, the FoM reward achieved, the number of simulations executed , and the used runtime. Our reward shaping helps in first encouraging the agent to mostly focus on satisfying the hard constraints and afterwards (instead of over-optimizing all specifications) to mainly search for solutions with less total current and area. The sample efficiency results of our MBTD3 algorithm, compared to the baseline model-free TD3 and the Genetic Algorithm, are plotted in Figures 4-b and 5-b. As discussed in the previous section, we keep track of the Pareto front with optimal solutions during the process. Therefore, in Figures 4-c and 5-c, we show a comparison of the optimal current-area Pareto fronts achieved by the model-free TD3 and our MBTD3 after 5k simulations. Furthermore, we compare the results and performances between our MBTD3 and the model-free TD3, at the point of maximum FoM achieved by the model-free TD3. The corresponding point selected for comparison from the MBTD3 results already has a higher FoM than that of the model-free TD3. Therefore, the sample efficiency and runtime improvements (as can be seen from Figures 4-b, 5-b) may even be better at other points. Yet, in this comparison, MBTD3 reaches better solutions with 2.01 (3.37) times fewer simulations and 26.3% (48.7%) less runtime for the two-stage (folded-cascode) opamp. One environment step in MBTD3 requires more time than in the model-free TD3 because of the computation overload. Therefore, for more complex circuits where the simulation time is higher, the runtime benefits of MBTD3 are even higher.

B. Sizing a complex Data Receiver circuit

In the next step, we consider sizing a full data receiver to showcase the scalability of our method with the multi-agent approach. The data receiver circuit of Figure 6-a is designed to be employed at the front end of an off-chip DRAM (DDR4) for receiving and amplifying data transmitted by the processor. The circuit is composed of three sub-blocks and therefore we leverage MA-MBTD3 for sizing it (Figure 3). We compare our results with the baseline model-free MATD3 and the single-agent TD3, as well as our single-agent MBTD3. Because of the longer simulation times for the data receiver circuit, we run each algorithm for 20 hours.

The data receiver circuit has a total of 66 design parameters. The system must function at a maximum frequency of 2.133 GHz with other hard constraints summarized in Table 2. The receiver topology begins with a pre-amplifier block, followed by RCV1 as a buffer stage with CMFB, and RCV2 (with a NAND gate) for gain maximization. There are two hard constraints that need more specific considerations: 1) input common-mode voltage (ICMV): the sub-blocks need to be functional in a range of input common-mode voltages. For handling this, we consider the lower bound (V_l), the upper bound (V_u), and the middle value (V_m) of the specified ICMV. Then, at each environment step, we randomly select one of these voltages for simulation and reward evaluation. Whenever the hard constraints are met, the other two voltages are also tested and the reward is averaged across these ICMV scenarios.
2) output common-mode voltage (OCMV): The output common-mode voltage specification is also a range of voltages (V_{lb}, V_{ub}). To address this kind of specification, we consider the following normalization and rewarding term:

$$z_{ocm} = \frac{V_{ocm}}{V_{DD}} - 1, r_{ocm} = \begin{cases} 0 & ;V_{lb} < V_{ocm} < V_{ub} \\ -\left|\frac{\left(\frac{2\times V_{ocm}}{V_{ub}+V_{lb}}\right)-1}{\frac{V_{DD}}{V_{ub}}-1}\right| & \text{otherwise} \end{cases} \qquad (6)$$

in which V_{ocm} is the output voltage measured after simulation, z_{ocm} is the normalized term used in the state definition, and r_{ocm} is the reward assigned to this specification in the total reward of the hard constraints (r_H in Eq. (1)).

In addition, for this receiver circuit, there is one more optimization target; the gain of the last stage. Therefore, there are three optimization targets for this system in our experiments. Figure 6-b shows the FoM results reached using model-based and model-free MATD3, as well as their single-agent counterparts. As can be seen, MA-MBTD3 reaches the first solution (all hard constraints satisfied) in approximately half of the simulations and runtime. Also, we show the three (two by two) optimal fronts of the targets (after 10k simulations) (Figure 6-c, d, e). Finally, in Table 2, a comparison is shown of the results and simulation counts at the maximum FoM reached by the model-free MATD3. These

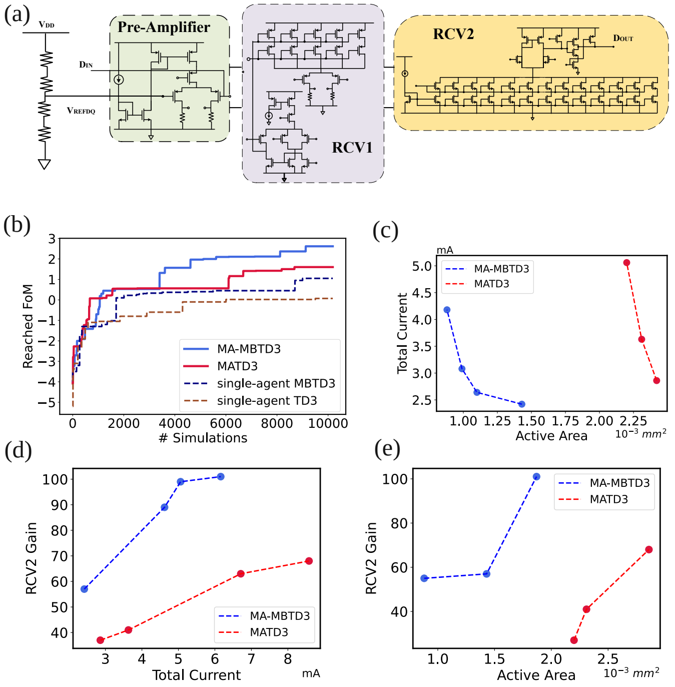

Figure 6: a) Schematics of the data receiver circuit: b) learning curves; c) current vs. area optimal fronts; d) RCV2 gain vs. current optimal fronts; e) RCV2 gain vs. area optimal fronts

results all indicate that for complex circuits, MA-MBTD3 requires significantly fewer simulations and computation time to reach the optimal solutions.

4 CONCLUSION

A model-based policy optimization with twin-delayed deep deterministic policy gradient has been proposed for boosting the sample efficiency in analog circuit sizing optimization. We have shown that using an ensemble of probabilistic models for generating short synthetic rollouts and exploring the design space results in significantly faster convergence towards power/area-optimal fronts. Also, a multi-agent reinforcement learning scheme with probabilistic model rollouts has been proposed for the sizing of complex circuits consisting of multiple sub-blocks. Our methods yield ~3x improvement in sample efficiency and ~2x in computation time, paving the way for more efficient reinforcement learning-based design automation of analog circuits.

Table 2: Per-block design constraints of the data receiver and general performance results

Spec.	Pre-Amp.	RCV1	RCV2
Max. Freq. (*GHz*)	>2.133	>2.133	>2.133
Gain	>1	>1	Maximize
ICMV (*V*)	0.15-0.3	0.15-0.3	$\leq$ OCMV$_{RCV1}$
OCMV (*V*)	0.15-0.3	0.28-0.32	0.3-0.4
Performance	Target	MATD3	MA-MBTD3
Frequency (*GHz*)	>2.133	2.193	2.487
Differential Slew Rate (*V/nsec*)	>6	6.6	7.3
RCV2 Gain	-	75	92
Total Current (*mA*)	-	6.8	5.4
Area ($\times 10^{-3} mm^2$)	-	3.1	2.3
FoM	-	1.52	**1.94**
#Simulations	-	8620	**4598**
Runtime (hours)	-	14.33	**7.9**

ACKNOWLEDGMENTS

This work was funded by the European Research Council (ERC) under the Horizon 2020 program (grant n° 101019982 - AnalogCreate). The authors would also like to thank Jan Lappas and Professor Norbert Wehn from the University of Kaiserslautern-Landau for providing the data receiver circuit.

REFERENCES

[1] Manuel Barros et al. *Analog Circuits and Systems Optimization based on Evolutionary Computation Techniques*, volume 294 of *Studies in Computational Intelligence*. Springer Berlin Heidelberg.

[2] Georges Gielen et al. Computer-aided design of analog and mixed-signal integrated circuits. *Proceedings of the IEEE*, 88(12):1825–1854, 2000.

[3] Glenn Wolfe et al. Extraction and use of neural network models in automated synthesis of operational amplifiers. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(2):198–212, 2003.

[4] Tiago Pessoa et al. Enhanced analog and RF IC sizing methodology using PCA and NSGA-II optimization kernel. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 660–665. IEEE.

[5] Wenlong Lyu et al. Batch bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design. In *International conference on machine learning*, pages 3306–3314. PMLR, 2018.

[6] Bo Liu et al. Gaspad: A general and efficient mm-wave integrated circuit synthesis method based on surrogate model assisted evolutionary algorithm. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(2):169–182, 2014.

[7] Keertana Settaluri et al. Autockt: Deep reinforcement learning of analog circuit designs. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 490–495. IEEE, 2020.

[8] Bobak Shahriari et al. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[9] Peter Frazier et al. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[10] Hanrui Wang et al. Learning to design circuits. In *NeurIPS Machine Learning for Systems Workshop*, 2018.

[11] Hanrui Wang et al. GCN-RL circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

[12] N.S. Karthik Somayaji, Hanbin Hu, and Peng Li. Prioritized reinforcement learning for analog circuit optimization with design knowledge. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 1231–1236. IEEE.

[13] Kai Yang et al. Trust-region method with deep reinforcement learning in analog design space exploration. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE.

[14] Wei Shi et al. Robustanalog: Fast variation-aware analog circuit design via multi-task rl. In *Proceedings of the 2022 ACM/IEEE Workshop on Machine Learning for CAD*, pages 35–41, 2022.

[15] Minjeong Choi et al. Reinforcement learning-based analog circuit optimizer using gm/id for sizing. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2023.

[16] Jinxin Zhang et al. Automated design of complex analog circuits with multiagent based reinforcement learning. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2023.

[17] Ahmet 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)*, pages 1219–1224. IEEE, 2021.

[18] Youngchang Choi et al. MA-Opt: Reinforcement learning-based analog circuit optimization using multi-actors. In *2023 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–5, 2023.

[19] Kurtland Chua et al. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

[20] Michael Janner et al. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

[21] Wei Zhao et al. New generation of predictive technology model for sub-45 nm early design exploration. *IEEE Transactions on electron Devices*, 53(11):2816–2823, 2006.