BoA-PTA: A Bayesian Optimization Accelerated PTA Solver for SPICE Simulation

WEI W. XING and XIANG JIN, Beihang University
TIAN FENG, China University of Petroleum, Beijing
DAN NIU, Southeast University
WEISHENG ZHAO, Beihang University
ZHOU JIN, China University of Petroleum, Beijing

One of the greatest challenges in integrated circuit design is the repeated executions of computationally expensive SPICE simulations, particularly when highly complex chip testing/verification is involved. Recently, pseudo-transient analysis (PTA) has shown to be one of the most promising continuation SPICE solvers. However, the PTA efficiency is highly influenced by the inserted pseudo-parameters. In this work, we proposed BoA-PTA, a Bayesian optimization accelerated PTA that can substantially accelerate simulations and improve convergence performance without introducing extra errors. Furthermore, our method does not require any pre-computation data or offline training. The acceleration framework can either speed up ongoing, repeated simulations (e.g., Monte-Carlo simulations) immediately or improve new simulations of completely different circuits. BoA-PTA is equipped with cutting-edge machine learning techniques, such as deep learning, Gaussian process, Bayesian optimization, non-stationary monotonic transformation, and variational inference via reparameterization. We assess BoA-PTA in 43 benchmark circuits and real industrial circuits against other SOTA methods and demonstrate an average of 1.5x (maximum 3.5x) for the benchmark circuits and up to 250x speedup for the industrial circuit designs over the original CEPTA without sacrificing any accuracy.

CCS Concepts: • **Computing methodologies → Gaussian processes**; • **Hardware → Software tools for EDA;**

Additional Key Words and Phrases: Bayesian optimization, Gaussian process, deep learning, SPICE, PTA, CEPTA, circuit simulation

W. W. Xing and X. Jin contributed equally to this research.

This work was supported in part by the Zhongguancun open laboratory concept verification project (20210421085), the National Natural Science Foundation of China (11804016 and 62006011), the Natural Science Foundation of Jiangsu Province of China (BK20202006), and the Science Foundation of China University of Petroleum (2462020YXZZ024).

Authors' addresses: W. W. Xing, X. Jin, and W. Zhao, School of Integrated Circuit Science and Engineering, Beihang University, 37 Xueyuan Road, Haidian District, Beijing, China; emails: wxing@buaa.edu.cn, zy2141113@buaa.edu.cn, wszhao@buaa.edu.cn; T. Feng and Z. Jin (corresponding author), Department of Computer Science and Technology, University of Petroleum–Beijing, 18# Changping Restrict Fuxue Road,Beijing, China; email: fengtian@student.cup.edu.cn; D. Niu, School of Automation, Southeast University, Sipailou, Nanjing, China; email: danniu1@163.com.
Updated author affiliation: W. W. Xing, Beihang Hangzhou Innovation Institute Yuhang, Xixi Octagon City, Yuhang District, Hangzhou 310023, China.

1084-4309/2022/12-ART27 $15.00
https://doi.org/10.1145/3555805

ACM Reference format:
Wei W. Xing, Xiang Jin, Tian Feng, Dan Niu, Weisheng Zhao, and Zhou Jin. 2022. BoA-PTA: A Bayesian Optimization Accelerated PTA Solver for SPICE Simulation. *ACM Trans. Des. Autom. Electron. Syst.* 28, 2, Article 27 (December 2022), 26 pages.
https://doi.org/10.1145/3555805

1 INTRODUCTION

With increasing degrees of the integration of modern **integrated circuits (ICs)**, the reliability of a chip design is improved via a time-consuming verification process before it can be taped out [8]. The verification mainly verifies whether a designed IC is physically feasible and robust by such as Monte Carlo analysis [2] and dynamic timing analysis [25], all of which require repeated executions of an expensive SPICE (simulation program with IC emphasis) simulation (due to the large scale of an IC design) [24]. Moreover, exploring the large design space such as analog circuit synthesis [53] also relies on repeated SPICE simulations. This poses a great challenge, as the verification can take up to 80% of the development time in an IC design [41].

Due to its recent fast development, machine learning and other statistical learning methods have been utilized to resolve this challenge [11]. For instance, **Bayesian optimization (BO)** [53], multi-fidelity modeling [52], and computing budget allocation [20] are proposed to accelerate repeated simulations. Despite being efficient, direct machine learning implementations rely on a large amount of pre-computed data to work. Furthermore, almost all machine learning based methods provide no error bounds in any form, putting the verification process at great risk. Thus, machine learning methods are mainly used in academic research rather than industrial applications.

A "first principal" way to reduce the computational expense is to improve the SPICE efficiency. A SPICE solves nonlinear algebraic equations or algebraic differential equations that are constructed on a circuit based on Kirchhoff's current law (KCL) and Kirchhoff's voltage law (KVL) [9]. The first step is to provide the solution of **direct current (DC)** analysis, which supports other detailed analyses, such as transient analysis and small-signal analysis [38]. A general SPICE, such as the widely used Ngspice, utilizes **Newton-Raphson (NR)** iteration and other continuation methods, such as Gmin stepping [29] and source stepping [35], to solve the nonlinear equations due to their fast convergence properties. These classic algorithms, however, may fail to converge when the circuit scale is sufficiently large and especially with a strong nonlinearity design. In particular, NR can fail if the initial guess is not close enough to the final solution; Gmin and source stepping [49] can fail for positive feedback, high loop gain, or multiple solutions if bifurcation or ill-conditioned matrix occurs [39]. The non-convergence is not unusual in practical simulations [49]. This challenge can be solved by homotopy methods, such as fixed point homotopy, Newton homotopy, and nonlinear homotopy [37, 40, 46, 47]. Although these methods are proven to be global convergent and can improve the convergence effectively, their implementation highly depends on the device model, making them not that practical in real simulators. This challenge is well resolved by **pseudo-transient analysis (PTA)** [28], which inserts constant pseudo capacitors and inductors to original circuits and converts the original hard-to-solve nonlinear algebraic equations into ordinary differential equations that are easier to solve. In commercial SPICE simulators, PTA is commonly the last option when all other methods (including NR, Gmin, source stepping) fail to converge, because the PTA-based methods are known to be robust to large-scale problems but often suffer from slow convergence, leading to a large number of iterations [29] due to oscillation issues. **Damped PTA (DPTA)** [44] exploits a numerical integration method with artificially enlarged damping effect to deal with the oscillation; **ramping PTA (RPTA)** [13] ramps up voltage sources instead of inserting the pseudo-inductors to suppress fill-ins. **Compound element PTA (CEPTA)** [50] has demonstrated a strong capability to eliminate oscillation while maintaining high

efficiency. However, the inserted values and position, a.k.a, the the solver parameters, significantly affect its efficiency and even its convergence performance. Meanwhile the solver parameters are highly circuit dependent. As to our knowledge, until now, there has been no literature showing effective (solver) parameter strategies for CEPTA acceleration.

To harness the power of modern machine learning and meanwhile retain accuracy reliability of a SPICE simulation, we aim to equip the state-of-the-art SPICE solver, CEPTA, with machine learning power. To this end, we propose BoA-PTA, a *B*ayesian *o*ptimization error-free *A*cceleration framework for CE*PTA* (Figure 1). Specifically, we introduce a BO [7, 23, 31] to select PTA solver parameters as an optimization problem. To extend the applicability for different circuits, we utilize a special netlist characterization and a **deep neural network (DNN)** for netlist feature extraction. To further improve BoA-PTA for the highly nonlinear optimization problem, we introduce a Bayesian hierarchical warping **Beta cumulative density function (BetaCDF)**, which overcomes the stationary limitation of a general BO without complicating the geometry via a monotonic bijection transformation [33]. Parameters of the warping BetaCDF are integrated out using variational inference combined with reparameterization trick [18] to avoid overfitting. Last, the optimization constraint and scale are handled by a log-sigmoid transformation. We highlight the novelty of BoA-PTA as follows:

(1) As far as the authors are aware, BoA-PTA is the first machine learning enhanced SPICE solver.
(2) BoA-PTA provides error-free accelerations and improves convergence performance for the CEPTA SPICE solver (and potentially other SPICE solvers).
(3) BoA-PTA requires no pre-computed data. It can accelerate ongoing repeated simulations or improve new simulations of completely unseen circuits.
(4) BoA-PTA is equipped with cutting-edge machine learning techniques: deep learning for netlist feature extractions, BetaCDF for non-stationary modeling, and variational inference to avoid overfitting.
(5) BoA-PTA shows an average 1.5x (maximum 3.5x) speedup on 43 benchmark circuit simulations and up to 250x speedup for practical circuit designs.

We implement our acceleration framework for CEPTA due to its urgent need for solver parameter tuning. Nevertheless, our method is ready to combine with other SPICE solvers. As a very first work of machine learning enhanced SPICE, it is our hope that this work can inspire interesting machine learning enhanced EDA tools from different perspectives.

The rest of the article is organized as follows. In Section 2, we review the background of PTA and BO. In Section 3, BoA-PTA is derived with motivations and details. In Section 4, we assess BoA-PTA on 43 benchmark circuit simulations and four practical circuits for different tasks. We conclude this work in Section 5.

2 BACKGROUND AND PRELIMINARIES

2.1 SPICE Simulations via PTA

DC analysis computes DC operating points by solving a series of equations that describe the circuits' behavior as shown in (1), where the unknown vector $\mathbf{v} \in \mathbb{R}^Q$ denotes the node voltages to the datum node, $\mathbf{p} \in \mathbb{R}^P$ denotes the internal current in the independent voltage sources, and $P + Q = L$.

$$\mathbf{F}(\mathbf{u}) = \mathbf{0}, \mathbf{u} = (\mathbf{v}, \mathbf{p})^T, \mathbf{F} : \mathbb{R}^L \rightarrow \mathbb{R}^L \tag{1}$$

When the commonly used NR and practical continuation methods fail to converge in the circuit simulator (e.g., SPICE), the PTA is implemented as an alternative because it provides robust

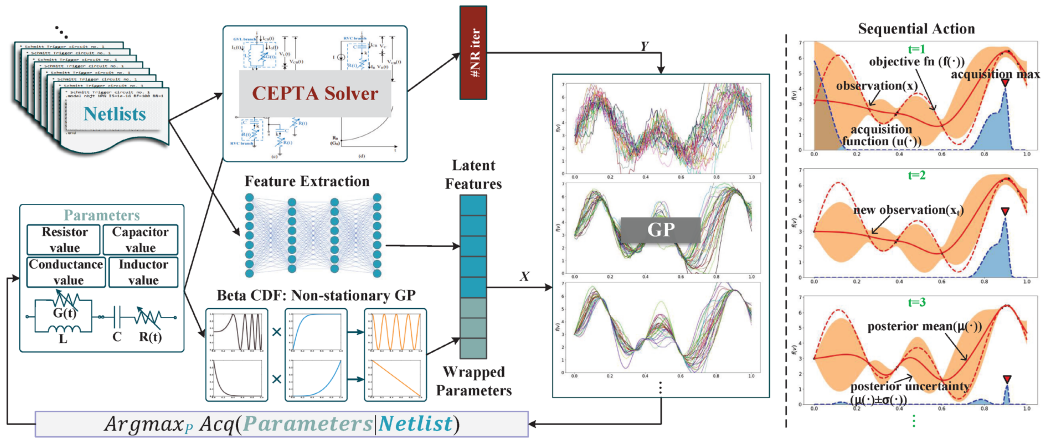


Fig. 1. The proposed BoA-PTA framework: the CEPTA solver is treated as a black-box function with the netlist and the inserted parameters being the model inputs and the CEPTA required iterations to complete the simulation as the outputs. The netlist features are extracted using a two-layer multiple-layer perceptron, and the CEPTA inserted values are transformed using the Beta cumulative density function (BetaCDF) [33] to address the non-stationary challenge. The GP-based [26] BO is conducted conditioning on a given netlist. An illustration of how the sequential BO [7] works is shown on the right-hand side.

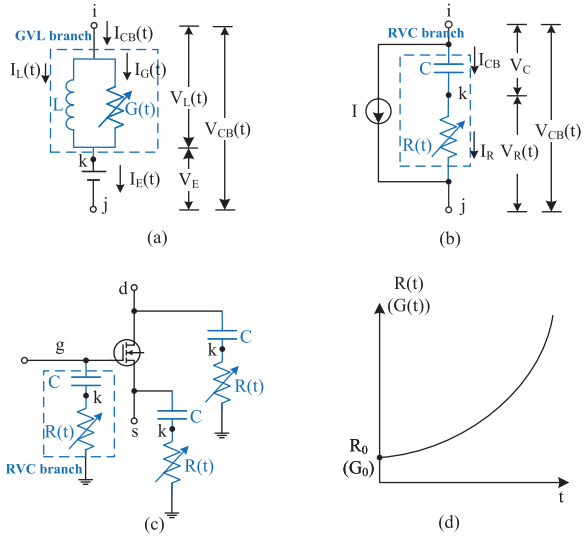
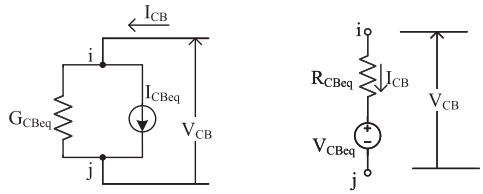


Fig. 2. Inserted pseudo-elements ((a) GVL, (b, c) RVC) and their embedding positions in CEPTA ((a) to independent voltage sources, (b) to independent current sources, (c) to MOS/BJT transistors, (d) function of inserted time-variant resistor and conductance).

solutions to nonlinear algebraic equations formed by **modified nodal analysis (MNA)**. PTA works by inserting certain dynamic pseudo-elements into the original circuits. As shown in Figure 2, CEPTA inserts a GVL branch into an independent voltage source in serial (Figure 2(a)), an RVC branch into an independent current source in parallel (Figure 2(b)), and transistors between each node to ground (Figure 2(c)). The RVC branch is composed of a constant capacitor C connected in serial with a time-variant resistor $R(t)$, whereas the GVL branch is composed of a constant



(a) RVC (b) GVL

Fig. 3. Equivalent circuit for the branches RVC and GVL.

inductor L connected in parallel with a time-variant conductance $G(t)$. With the pseudo-elements inserted, a new set of differential algebraic Equations (2) is formed. Solve it by a transient analysis with an initial guess u_0 until convergence:

$$\mathbf{g}(\mathbf{u}(t), \dot{\mathbf{u}}(t), t) = 0, \tag{2}$$

where

$$R(t) = R_0 e^{t/\tau}, \quad G(t) = G_0 e^{t/\tau}. \tag{3}$$

R_0 and G_0 is an initial resistor and conductance value and τ is the time constant. When it gets the steady-state, the inserted GVL branch can be considered as short, whereas the RVC branch can be considered as open. The converged solution (when $\dot{\mathbf{u}}(t) = 0$) is the solution to the original circuit.

2.2 Stamping

The insertion of compound elements brings additional nodes (e.g., node k in Figure 2) that require extra computations. To avoid enlarging the order of the Jacobian matrix induced by additional nodes, CEPTA can be implemented in an equivalent way, where the equivalent circuits are shown as Figure 3. For the RVC branch, the relationship between current I_{CB} and voltage V_c (Figure 2) in the continuous-time domain is

$$I_{CB} = C * dV_c/dt. \tag{4}$$

Applying the backward Euler formula to the differential part in (4), we have

$$I_{CB}^{n+1} = C * \left(\frac{{V_c}^{n+1} - {V_c}^{n}}{h} \right) = C * \left(\frac{({V_{CB}}^{n+1} - {V_R}^{n+1}) - (V_{CB}^{n} - {V_R}^{n})}{h} \right) \tag{5}$$

at discrete time point t^{n+1}. Therefore, we can obtain the following equivalent equation, which can be considered as an equivalent inserted circuit as shown in Figure 3(a):

$$I_{CB}^{n+1} = G_{CBeq} V_{CB}^{n+1} + I_{CBeq}, \tag{6}$$

where

$$\begin{aligned} G_{CBeq}^{-1} &= h^{n+1}/C + R(t^{n+1}), \\ I_{CBeq} &= G_{CBeq} \left(I_{CB}^{n} R(t^{n}) - V_{CB}^{n} \right). \end{aligned} \tag{7}$$

Similarly, the equivalent equations for branch GVL at time point t^{n+1} can be obtained by

$$V_{CB}^{n+1} = R_{CBeq} I_{CB}^{n+1} + V_{CBeq}, \tag{8}$$

where

$$\begin{aligned} R_{CBeq}^{-1} &= h^{n+1}/L + G^{n+1}, \\ V_{CBeq} &= R_{CBeq} \left(-I_{CB}^{n} + G^{n} \left(V_{CB}^{n} - E \right) \right) + E. \end{aligned} \tag{9}$$

Table 1. The Stamping for Branch RVC for SPICE3 Implementation

	V_i	V_j	RHS
i	G_{CBeq}	$-G_{CBeq}$	$-I_{CBeq}$
j	$-G_{CBeq}$	G_{CBeq}	I_{CBeq}

Table 2. The Stamping for Branch GVL for SPICE3 Implementation

	V_i	V_j	I_E	RHS
i			1	
j			−1	
BR	1	−1	$-R_{CBeq}$	V_{CBeq}

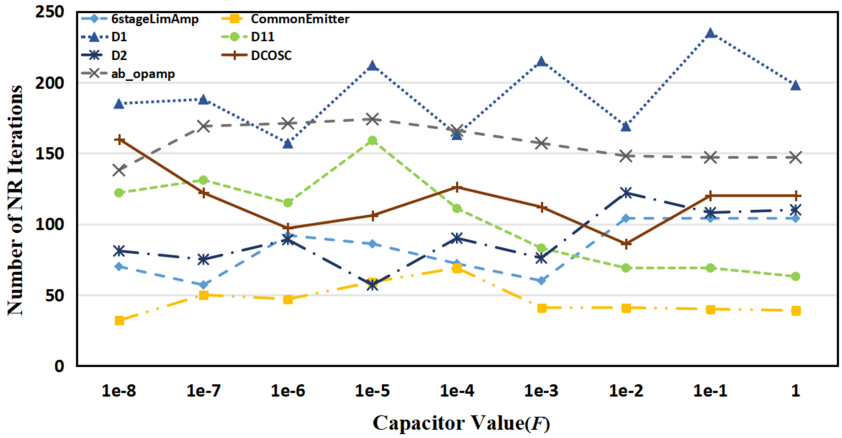


Fig. 4. Simulation performance (the number of NR iterations) of CEPTA with different capacitor values for seven different circuits.

In other words, to implement CEPTA in SPICE, the solver needs to stamp the these equivalent resistances into the original MNA matrix as shown in Table 1 and Table 2 for RVC and GVL branches, respectively, which are composed of pseudo capacitor value C, resistor R, inductor L, and conductance G.

Despite CEPTA's great success, its performance is highly influenced by the inserted pseudo-elements—that is, the values of the inserted pseudo capacitor, inductor, and the initial values of resistor and conductance. Figure 4 shows the simulation efficiency with different inserted capacitor values for seven different circuit simulations, which are from the benchmark discussed later in Section 4.2. It is thus important to quickly find a set of optimal inserted pseudo-elements that accelerates the convergence and thus the repeated SPICE simulations. The circuit-dependent and sensitive property make solver parameter tuning a still open challenge.

2.3 Problem Formulation

Consider a CEPTA solver g with solver parameters $\boldsymbol{x} \in \mathbb{R}^4$ (indicating the value of inserted capacitor, inductor, resistor, and conductance) that operates on a netlist file denoted as $\boldsymbol{\xi}$ and generate the steady state $\boldsymbol{u} = g(\boldsymbol{x}, \boldsymbol{\xi})$. We are interested in reducing the number of iterations, denoted as $\eta(\boldsymbol{x}, \boldsymbol{\xi}) + \varepsilon$, for $g(\boldsymbol{x}, \boldsymbol{\xi})$. Here, ε captures the model inadequacy and randomness that are not fully

captured by $\boldsymbol{x}$ and ξ when we execute the CEPTA solver. We aim to seek a function

$$\boldsymbol{x}^*(\xi) = \underset{\boldsymbol{x}\in\mathcal{X}}{\operatorname{argmin}}\, \eta(\boldsymbol{x}, \xi), \tag{10}$$

where $\boldsymbol{x}^*(\xi)$ is the optimal CEPTA solver parameters for any given netlist ξ and $\operatorname{argmin}_{\boldsymbol{x}\in\mathcal{X}} \eta(\boldsymbol{x}, \xi)$ is the variable $\boldsymbol{x}$ in the feasible domain $\mathcal{X}$ that makes function $\eta(\boldsymbol{x}, \xi)$ achieve the minimum value.

2.4 Bayesian Optimization

BO is an optimization framework generally for expensive black-box functions that are noisy or noise-free [23]. Since the black-box function is expensive to evaluate, it is approximated by a probabilistic surrogate model, which provides useful derivative information for the classic optimization approaches. The surrogate model is a data-driven probabilistic regression that is calibrated to fit the black-box function with available data. Since conducting the black-box function to collect data is an expensive procedure, the surrogate model along with the goal of optimization is conducted in a sequential manner. A candidate point that produces the largest uncertainty can be proposed, which is known as exploration; in contrast, a candidate focusing on maximizing the prediction is known as exploitation. The tradeoff between exploration and exploitation is handled by the acquisition function, which should reflect our reference for the tradeoff.

2.5 Gaussian Process

The **Gaussian process (GP)** is a common choice for the surrogate model of BO due to its model capacity for complex black-box function and for uncertainty quantification, which naturally quantifies the tradeoff. We briefly review GP in this section.

For the sake of clarity, let us consider a case where the circuit is fixed and its index ξ is thus omitted. Assume that we have observation $y_i = \eta(\boldsymbol{x}_i) + \varepsilon$, $i = 1, \ldots, N$ for design points $\boldsymbol{x}_i$, where y is the (determined) iteration number needed for convergence. In a GP model, we place a prior distribution over $\eta(\boldsymbol{x})$ indexed by $\boldsymbol{x}$:

$$\eta(\boldsymbol{x})|\boldsymbol{\theta} \sim \mathcal{GP}\left(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'|\boldsymbol{\theta})\right), \tag{11}$$

with mean and covariance functions:

$$\begin{aligned} m_0(\boldsymbol{x}) &= \mathbb{E}[\eta(\boldsymbol{x})], \\ k(\boldsymbol{x}, \boldsymbol{x}'|\boldsymbol{\theta}) &= \mathbb{E}[(\eta(\boldsymbol{x}) - m_0(\boldsymbol{x}))(\eta(\boldsymbol{x}') - m_0(\boldsymbol{x}'))], \end{aligned} \tag{12}$$

in which $\mathbb{E}[\cdot]$ is the expectation operator. The hyperparameters $\boldsymbol{\theta}$ are estimated during the learning process. The mean function can be assumed to be an identical constant, $m_0(\boldsymbol{x}) \equiv m_0$, by virtue of centering the data. Alternative choices are possible, such as a linear function of $\boldsymbol{x}$, but rarely adopted unless *a priori* information on the form of the function is available. The covariance function can take many forms, the most common being the **automatic relevance determinant (ARD)** kernel:

$$k(\boldsymbol{x}, \boldsymbol{x}'|\boldsymbol{\theta}) = \theta_0 \exp\left(-(\boldsymbol{x} - \boldsymbol{x}')^T \operatorname{diag}(\theta_1, \ldots, \theta_l)(\boldsymbol{x} - \boldsymbol{x}')\right), \tag{13}$$

where l is the input dimensionality and $\operatorname{diag}(\theta_1, \ldots, \theta_l)$ is a diagonal matrix with diagonal elements $\theta_1, \ldots, \theta_l$, and the ARD Matern 5/2 kernel [32]:

$$k(\boldsymbol{x}, \boldsymbol{x}'|\boldsymbol{\theta}) = \theta_0 \left(1 + \sqrt{5d^2} + \frac{5}{3}d^2\right) \exp\left(-\sqrt{5d^2}\right), \tag{14}$$

where $d^2 = -(\boldsymbol{x} - \boldsymbol{x}')^T \operatorname{diag}(\theta_1, \ldots, \theta_l)(\boldsymbol{x} - \boldsymbol{x}')$ is the effective distance that is also used in Equation (13). The hyperparameters $\theta_1^{-1}, \ldots, \theta_l^{-1}$ are referred to as the square correlation lengths controlling the contribution of each input dimension in both cases, whereas θ_0 manipulates the overall magnitude. Denote all of the hyperparameters $\boldsymbol{\theta} = (\theta_0, \ldots, \theta_l)^T$.

For any fixed $\boldsymbol{x}$, $\eta(\boldsymbol{x})$ is a random variable before we actually execute the simulation $\eta(\boldsymbol{x})$. A collection of observations $\eta(\boldsymbol{x}_i)$, $i = 1, \ldots, N$, however, is a partial realization of the random process indexed by $\boldsymbol{x}$, which is assumed GP prior in Equation (11). The main property of GPs is that the joint distribution of $\eta(\boldsymbol{x}_i)$, $i = 1, \ldots, N$, is multivariate Gaussian. Assuming the model inadequacy $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ is also a Gaussian, with the prior (11) and available data $\boldsymbol{y} = (y_1, \ldots, y_N)^T$, we can derivative the model likelihood

$$\begin{aligned}\mathcal{L} \triangleq p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}) &= \int_{\eta} (\eta(\boldsymbol{x}) + \varepsilon) d\eta = \mathcal{N}(\boldsymbol{y}|m_0\mathbf{1}, \mathbf{K}(\boldsymbol{\theta}) + \sigma^2\mathbf{I}) \\ &= -\frac{1}{2}(\boldsymbol{y} - m_0\mathbf{1})^T (\mathbf{K}(\boldsymbol{\theta}) + \sigma^2\mathbf{I})^{-1} (\boldsymbol{y} - m_0\mathbf{1}) - \frac{1}{2}\ln|\mathbf{K}(\boldsymbol{\theta}) + \sigma^2\mathbf{I}| - \frac{N}{2}\log(2\pi),\end{aligned} \tag{15}$$

where the covariance matrix $\mathbf{K}(\boldsymbol{\theta}) = [K_{ij}]$, in which $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j|\boldsymbol{\theta})$, $i, j = 1, \ldots, N$; $\mathbf{1}$ is a vector with element 1, and $\mathbf{I}$ is the identity matrix. The hyperparameters $\boldsymbol{\theta}$ are normally obtained from point estimates [16] by maximum likelihood estimate (MLE) of (15) w.r.t. $\boldsymbol{\theta}$. The joint distribution of $\boldsymbol{y}$ and $\eta(\boldsymbol{x})$ also form a joint Gaussian distribution with mean value $m_0\mathbf{1}$ and covariance matrix

$$\mathbf{K}'(\boldsymbol{\theta}) = \left[\begin{array}{c|c} \mathbf{K}(\boldsymbol{\theta}) + \sigma^2\mathbf{I} & \boldsymbol{k}(\boldsymbol{x}) \\ \hline \boldsymbol{k}^T(\boldsymbol{x}) & k(\boldsymbol{x}, \boldsymbol{x}|\boldsymbol{\theta}) + \sigma^2 \end{array}\right], \tag{16}$$

in which $\boldsymbol{k}(\boldsymbol{x}) = (k(\boldsymbol{x}_1, \boldsymbol{x}|\boldsymbol{\theta}), \ldots, k(\boldsymbol{x}_N, \boldsymbol{x}|\boldsymbol{\theta}))^T$. Conditioning on $\boldsymbol{y}$ provides the conditional predictive distribution at $\boldsymbol{x}$ [26]:

$$\begin{aligned}\hat{\eta}(\boldsymbol{x})|\boldsymbol{y}, \boldsymbol{\theta} &\sim \mathcal{N}\left(\mu(\boldsymbol{x}|\boldsymbol{\theta}), v(\boldsymbol{x}, \boldsymbol{x}'|\boldsymbol{\theta})\right), \\ \mu(\boldsymbol{x}|\boldsymbol{\theta}) &= m_0\mathbf{1} + \boldsymbol{k}(\boldsymbol{x})^T \left(\mathbf{K}(\boldsymbol{\theta}) + \sigma^2\mathbf{I}\right)^{-1} (\boldsymbol{y} - m_0\mathbf{1}), \\ v(\boldsymbol{x}|\boldsymbol{\theta}) &= \sigma^2 + k(\boldsymbol{x}, \boldsymbol{x}|\boldsymbol{\theta}) - \boldsymbol{k}^T(\boldsymbol{x}) \left(\mathbf{K}(\boldsymbol{\theta}) + \sigma^2\mathbf{I}\right)^{-1} \boldsymbol{k}(\boldsymbol{x}).\end{aligned} \tag{17}$$

The expected value $\mathbb{E}[\eta(\boldsymbol{x})]$ is given by $\mu(\boldsymbol{x}|\boldsymbol{\theta})$ and the predictive variance by $v(\boldsymbol{x}|\boldsymbol{\theta})$.

2.6 Acquisition Function

For simplicity, let us consider the maximization of the black-box function without particular constraints. Based on the GP model posterior in (17), we can simply calculate the improvements for a new input $\boldsymbol{x}$ as $I(\boldsymbol{x}) = \max(\hat{\eta}(\boldsymbol{x}) - y^\dagger, 0)$, where $y^\dagger$ is the current optimal and $\hat{\eta}(\boldsymbol{x})$ is the predictive posterior in (17). The **expected improvement (EI)** [14] over the probabilistic space is

$$\begin{aligned}EI(\boldsymbol{x}) &= \mathbb{E}_{\hat{\eta}(\boldsymbol{x}) \sim \mathcal{N}(\mu(\boldsymbol{x}), v(\boldsymbol{x}))}[\max(\hat{\eta}(\boldsymbol{x}) - y^\dagger, 0)] \\ &= (\mu(\boldsymbol{x}) - y^\dagger)\psi\left(\frac{\mu(\boldsymbol{x}) - y^\dagger}{v(\boldsymbol{x})}\right) + v(\boldsymbol{x})\phi\left(\frac{\mu(\boldsymbol{x}) - y^\dagger}{v(\boldsymbol{x})}\right),\end{aligned} \tag{18}$$

where $\psi(\cdot)$ and $\phi(\cdot)$ are the probabilistic density function (PDF) and cumulative density function (CDF) of a standard normal distribution, respectively. It is clear that the EI acquisition function (18) favors regions with larger uncertainty or regions with larger predictive mean values and naturally handles the tradeoff between exploitation and exploration. The candidates for the next iteration are selected by

$$\operatorname*{argmax}_{\boldsymbol{x} \in \mathcal{X}} EI(\boldsymbol{x}), \tag{19}$$

which is normally optimized with classic non-convex optimizations (e.g., L-BFGS-B) [54].

Rather than looking into the EI, we can approach the optimal by exploring the areas with higher uncertainty toward the maximum

$$\operatorname*{argmax}_{\boldsymbol{x} \in \mathcal{X}} \left(\mu(\boldsymbol{x}) + \beta^{\frac{1}{2}} v(\boldsymbol{x})\right), \tag{20}$$

where β reflects our preference of the tradeoff of exploration and exploitation. This is known as the **upper confidence bound (UCB)** [34], which is simple and easy to implement yet powerful and effective. However, the choice for β is nontrivial, which hinders its further applications.

Both EI and UCB acquisition functions try to extract the best from the current status. The **max-value entropy search (MES)** acquisition function is introduced by Wang and Jegelka [42] to take a further step to inquire at a location that produces maximum information gain (based on information theory) about the black-box function optimal,

$$\begin{aligned} MES(\boldsymbol{x}) &= -\mathbb{E}_{y(\boldsymbol{x})}[h(y^*|\mathcal{D} \cup \{\boldsymbol{x}, \hat{\eta}(\boldsymbol{x})\})] - H(y^*|\mathcal{D}) \\ &= -\mathbb{E}_{\hat{\eta}^*}[H(\hat{\eta}(\boldsymbol{x})|y^*)] + H(\hat{\eta}(\boldsymbol{x})), \end{aligned} \tag{21}$$

where y^* indicates the black-box function optimal, $\mathcal{D}$ means the current dataset, and $H(\hat{\eta}) = -\int p(\hat{\eta}) \log(p(\hat{\eta}))d\hat{\eta}$ is the entropy for $p(\hat{\eta})$. The first term in (21) is generally achieved using sampling method, whereas the second one has a closed-form solution. The readers are referred to the work of Wang and Jegelka [42] for more details.

BO is an active research area, and there are many other acquisition functions, such as knowledge gradient [30] and predictive entropy search [10]. Ensembles of multiple acquisition function are also possible [21]. In this work, we focus on BO accelerated SPICE and test it with the classic EI, UCB, and MES acquisition functions. However, our method is ready to combine with any other acquisition functions or portfolio strategy.

3 PROPOSED BOA-PTA

3.1 Circuits Characterization via Deep Learning

The most challenging part of this work is the characterization of the circuit on which the SPICE solver is executed. Recently, machine learning techniques have been implemented in the EDA community to accelerate the design/verification process [11]. Directly introducing a powerful model such as deep learning that directly uses netlist as inputs is feasible in some cases [22]. However, this approach is unlikely to address our problem because we do not have a large amount of data nor great computational budget for model training. Even if we have, the overwhelming computational overhead required will make the approach impractical for real problems. Instead, we follow the work of Zhang et al. [51] and use the seven key factors (the number of nodes, MNA equations, capacitors, resistors, voltage sources, bipolar junction transistor, and MOS field-effect transistor) to characterize a netlist as raw inputs for BoA-PTA. These features are denoted as a column input $\boldsymbol{\xi}$. A GP with a commonly used kernel e.g., (13) is unlikely to be able to capture the complex correlations between different netlists. The DNN has been shown to be a powerful automatic feature extraction for various practical applications [19]. Thus, we further introduce deep learning transform as automatic feature extraction for $\boldsymbol{\xi}$ (i.e., $\Phi(\boldsymbol{\xi})$), before the GP surrogate:

$$\Phi(\boldsymbol{\xi}) = \mathbf{h}(\mathbf{W}^l \Phi^l(\boldsymbol{\xi}) + \boldsymbol{b}^l), \tag{22}$$

$$\Phi^l(\boldsymbol{\xi}) = \mathbf{h}(\mathbf{W}^{l-1} \Phi^{l-1}(\boldsymbol{\xi}) + \boldsymbol{b}^{l-1}), \tag{23}$$

where $\Phi^1(\boldsymbol{\xi}) = \boldsymbol{\xi}$ and $\mathbf{h}(\cdot)$ is an element-wise nonlinear transformation known as the active function in this scenario. This is a classic DNN structure known as the **multiple-layer perceptron (MLP)**, which is commonly used to process features in a deep model. In this work, we use the same dimension of $\boldsymbol{\xi}$ to be the output dimension of $\Phi(\boldsymbol{\xi})$. The extracted features are then passed to a GP for further feature selections by an ARD kernel and for model predictions. The DNN with the follow-up kernel together can be seen as a kernel that learns the complex correlations automatically through DNN. For this reason, this approach is also known as deep kernel learning [43].

3.2 Non-Stationary GP for CEPTA

The efficiency and effectiveness of BO is highly determined by the accuracy of the surrogate model; for a GP model, its model capacity is largely influenced by choice of the kernel function. Consider the function $\eta(\boldsymbol{x}, \boldsymbol{\xi})$ for a fixed $\boldsymbol{\xi}$, according to our experiments, $\eta(\boldsymbol{x}, \boldsymbol{\xi})$ is a highly nonlinear function w.r.t. $\boldsymbol{x}$, making the commonly used stationary ARD kernel ineffective for modeling such a complex function.

Unlike the previous section where the complex correlations of $\boldsymbol{\xi}$ can be captured automatically using a complex model such as DNN [51], latent space mapping [5], and GP [1], modeling of $\boldsymbol{x}$ requires extra care because (1) despite the strong model capacity, introducing a complex model is likely to introduce extra model parameters (particularly when a DNN is implemented), which makes the model training difficult and potentially requires more data for the surrogate to perform well, and (2) even worse, introducing another complex model can complicate the geometry, making the optimization of the non-convex acquisition function w.r.t. $\boldsymbol{x}$ more difficult. Note that the DNN we implement in Section 3.1 does not suffer from this issue because it is not involved in the optimization of the acquisition function. We will show the details in later sections.

For modeling $\boldsymbol{x}$, we believe that the rule of thumb is to follow Occam's razor and introduce a simple yet effective transformation for the solver parameter $\boldsymbol{x}$. To this end, we follow the work of Snoek et al. [33] and introduce a bijection BetaCDF,

$$w_d(x_d) = \int_0^{x_d} \frac{u^{\alpha_d - 1}(1-u)^{\beta_d - 1}}{B(\alpha_d, \beta_d)} du, \tag{24}$$

where α_d and β_d are the positive functional parameters, and $B(\alpha_d, \beta_d)$ is the normalization constant. This transformation is monotonic (thus does not complicate the optimization geometry), and it comes with only two extra parameters for each input dimension. To further reduce the probability of overfitting with $w_d(x_d)$, we use a hierarchical Bayesian model by placing priors

$$\log(\alpha_d) \sim \mathcal{N}\left(\mu_a^d, \sigma_a^d\right), \quad \log(\beta_d) \sim \mathcal{N}\left(\mu_b^d, \sigma_b^d\right), \tag{25}$$

for the BetaCDF. The introduced hyperparameters $\{\alpha_d, \beta_d\}_{d=1}^D$ can be obtained via point estimations. To avoid overfitting, Snoek et al. [33] integrate them out by using Markov chain Monte Carlo slice sampling, which significantly increases the model training time and will make the acceleration via BoA-PTA impractical because the BO itself consumes too many computational resources.

In this work, the reparameterization trick [18] is utilized to conduct a fast posterior inference for $\{\alpha_d, \beta_d\}_{d=1}^D$, which is later integrated out. We use a log-Gaussian variational posterior

$$\log\left(q(\boldsymbol{\gamma})\right) \sim \mathcal{N}(\boldsymbol{\mu}_\gamma, \Sigma_\gamma), \tag{26}$$

where $\boldsymbol{\gamma} = [\alpha_1, \ldots, \alpha_D, \beta_1, \ldots, \beta_D]^T$. This formulation allows us to capture the complex correlation between any α_d and $\beta_{d'}$.

3.3 Handling Constraints and Scales

The CEPTA solver parameters are practically in the range of $[10^{-7}, 10^7]$. This poses two challenges. First, it turns the unconstrained optimization into a constrained one that requires extra care. Second, in its original space, $[10^{-7}, 0]$ takes almost zero volume of the whole domain $[10^{-7}, 10^7]$. This makes an optimization either ignore the $[10^{-7}, 0]$ range completely or fail to search the whole domain with a small searching step.

To resolve these issues simultaneously, we introduce a log-sigmoid transformation,

$$x_d = \left(7 \cdot \text{sigmoid}(z_d)\right)^{10}, \tag{27}$$

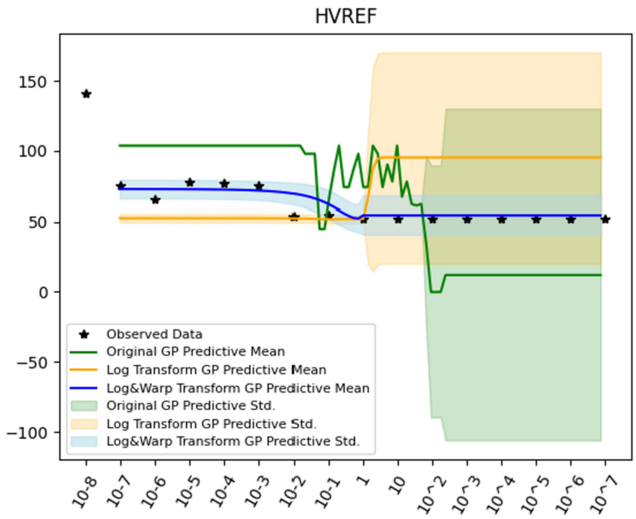


Fig. 5. Simulation performance (the number of NR iterations) with different capacitor values for HVREF circuit design.

where $\text{sigmoid}(z_d) = 1/(1+\exp(z_d))$ is the sigmoid function. In this equation, the base-10 logarithm scales x_d such that the optimization focuses on the magnitude of x_d rather the particular value, whereas the sigmoid function naturally bounds x_d to the range of $[10^{-7}, 10^{7}]$. This log-sigmoid transformation is applied to each x_d independently. When the optimization of the acquisition function is conducted, it is optimized w.r.t. z_d instead of x_d. Note that this log-sigmoid transformation does not change the monotone of $\eta(\boldsymbol{x}, \boldsymbol{\xi})$ nor affect the non-stationary transformation (24), which is designed to tweak the space $\mathcal{X}$ to resolve the non-stationary issue.

To illustrate the benefit of the log-sigmoid and the aforementioned BetaCDF transformation, we show the GP fitting for modeling the capacitor vs. the number of NR iterations for the HVREF circuit design using a vanilla GP, a GP with log-sigmoid transformation, and a GP with log-sigmoid and BetaCDF transformation in Figure 5. Note that the x-axis is on a logarithmic scale for a compact visualization, whereas the logarithmic scale is not naturally applied to the vanilla GP. It is clear that the vanilla GP cannot capture the wide domain issue and creates unstable oscillated predictions, which we frequently encountered during our experiments. The log-sigmoid transformation can improve the problem in a wide domain but still generates unreliable predictions and uncertainty estimations for large capacitor values. The log-sigmoid and BetaCDF together significantly improve the fitting.

3.4 BoA-PTA Training and Updating

With the features extracted from the netlist through the DNN Φ, the GP likelihood $\mathcal{L}$ is the same as (15) but with a composite covariance kernel function

$$k\left([\boldsymbol{w}(\boldsymbol{x}), \Phi(\xi)], [\boldsymbol{w}(\boldsymbol{x}'), \Phi(\xi')]\right). \tag{28}$$

The challenge presents itself due to the posterior of $\boldsymbol{\gamma}$, which hinders the joint model likelihood from taking a closed-form solution. Thus, we follow Bishop [4] and utilize Jensen's inequality to derive **evidence low bound (ELBO)** of the joint model log-likelihood given the observation set

$\{\boldsymbol{x}_i, \xi_i, y_i\}_{i=1}^N$,

$$\log \int p(\boldsymbol{y}, \boldsymbol{\gamma}|\boldsymbol{x}, \boldsymbol{\theta}) d\boldsymbol{\gamma} \geq \int q(\boldsymbol{\gamma}) \log p(\boldsymbol{y}|\boldsymbol{\gamma}, \boldsymbol{x}, \boldsymbol{\theta}) d\boldsymbol{\gamma} - \mathrm{KL}\left(q(\boldsymbol{\gamma})||p(\boldsymbol{\gamma})\right) \triangleq \mathcal{L}, \tag{29}$$

where $\mathrm{KL}\left(q(\boldsymbol{\gamma})||p(\boldsymbol{\gamma})\right) = \int q(\boldsymbol{\gamma}) \log \frac{q(\boldsymbol{\gamma})}{p(\boldsymbol{\gamma})} d\boldsymbol{\gamma}$ is the KL distance between $q(\boldsymbol{\gamma})$ of Equation (26) and $p(\boldsymbol{\gamma})$ of Equation (25), which has a closed-form solution given our prior and variational posterior. The details of the derivation are preserved in the appendix. However, the first term in Equation (29) is not tractable, and thus we cannot easily derive the derivative w.r.t. to the model hyperparameters and variational parameters. To resolve this issue, we take advantage of the latest development in machine learning of the reparameterization trick, which is proven to be more robust than other general black-box variational approaches. Specifically, we use the one-liner transformation,

$$\boldsymbol{\gamma}_s = \exp\left(\boldsymbol{\mu}_\gamma + \epsilon_i \cdot \mathbf{L}\right), \quad s = 1, \ldots, S, \tag{30}$$

where $\boldsymbol{\gamma}_s$ denotes an i.i.d. sampled of $\boldsymbol{\gamma}$, ϵ_s is independently sampled from a standard normal distribution $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and $\mathbf{L}\mathbf{L}^T = \Sigma_\gamma$ is the Cholesky decomposition of Σ_γ. The first term in Equation (29) can then be approximated using the samples $\{\boldsymbol{\gamma}_s\}_{s=1}^S$

$$\int q(\boldsymbol{\gamma}) \log p(\boldsymbol{y}|\boldsymbol{\gamma}, \boldsymbol{x}, \boldsymbol{\theta}) d\boldsymbol{\gamma} \approx \frac{1}{S} \sum_{s=1}^{S} \log p(\boldsymbol{y}|\boldsymbol{\gamma}_s, \boldsymbol{x}, \boldsymbol{\theta}). \tag{31}$$

This approximation allows us to write the joint likelihood $\mathcal{L}$ in an analytical form and to derive the exact derivative w.r.t. the model parameters. We can then maximize $\mathcal{L}$ using any gradient-based optimization method, such as Adam [17] and L-BFGS-B [54].

The DNN Φ parameters $\mathbf{W}^l$ and $\boldsymbol{b}^l$ are updated easily (because they do not affect by $\boldsymbol{\gamma}$) by using the following gradient

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^l} = \frac{1}{S} \sum_{s=1}^{S} \frac{\partial \mathcal{L}_s}{\partial k_s} \cdot \frac{\partial k_s}{\partial \Phi} \cdot \frac{\partial \Phi}{\partial \mathbf{W}^l}, \tag{32}$$

where $\mathcal{L}_s = \log p(\boldsymbol{y}|\boldsymbol{\gamma}_s, \boldsymbol{x}, \boldsymbol{\theta})$ is the log-likelihood with $\boldsymbol{\gamma}_s$, and k_s is the kernel function with input data augmented by the BetaCDF with parameter instance $\boldsymbol{\gamma}_s$. The DNN parameter derivatives need to be computed only once to save computation. Similarly, the derivative w.r.t. $\boldsymbol{b}^l$ is conducted in the same way,

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{b}^l} = \frac{1}{S} \sum_{s=1}^{S} \frac{\partial \mathcal{L}_s}{\partial k_s} \cdot \frac{\partial k_s}{\partial \Phi} \cdot \frac{\partial \Phi}{\partial \boldsymbol{b}^l}. \tag{33}$$

For the variational posterior $\boldsymbol{\gamma}$, the variational parameter derivatives are computed by

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_\gamma} = \frac{1}{S} \sum_{s=1}^{S} \frac{\partial \mathcal{L}_s}{\partial k_s} \cdot \frac{\partial k_s}{\partial \boldsymbol{w}_s} \cdot \frac{\partial \boldsymbol{w}_s}{\partial \boldsymbol{\gamma}_s} \cdot \frac{\partial \boldsymbol{\gamma}_s}{\partial \boldsymbol{\mu}_\gamma} \tag{34}$$

and

$$\frac{\partial \mathcal{L}}{\partial \mathbf{L}} = \frac{1}{S} \sum_{s=1}^{S} \frac{\partial \mathcal{L}_s}{\partial k_s} \cdot \frac{\partial k_s}{\partial \boldsymbol{w}_s} \cdot \frac{\partial \boldsymbol{w}_s}{\partial \boldsymbol{\gamma}_s} \cdot \frac{\partial \boldsymbol{\gamma}_s}{\partial \mathbf{L}}. \tag{35}$$

In practice, the surrogate model will be updated iteratively, whereas the reparameterization trick is indeed an unbiased estimation with low variance, and we opt for the single-point estimation by setting $S = 1$ to save computational resources as in the work of Kingma and Welling [18].

ALGORITHM 1: BoA-PTA Cold Start

Input: Netlists $[\xi_1, \ldots, \xi_M] = \Xi$, number of epoch N_{epoch}

```
Execute CEPTA with default setting on any netlist ξ_i
for j = 1 to N_epoch do
    for i = 1 to M do
        Update surrogate model M by maximizing (29)
        Update and optimize acquisition function (18), (20), or (21) given ξ_i and get candidate x
        Execute CEPTA and collect iteration η(x, ξ_i)
    end for
end for
return Best record of {x*, η(x*, ξ_i)} for i = 1, ..., M; surrogate model M
```

3.5 Simplified Optimization of Acquisition Functions

Unlike a general BO process where all input parameters of the surrogate model are optimized simultaneously, in our application, we always optimize the solver parameters $\boldsymbol{x}$ conditioned on a given netlist ξ. This makes the optimization much easier and faster without repeated forward and backward propagation through the DNN. Specifically, conditioned on a netlist ξ, the kernel (28) can be decomposed as $k_1(\boldsymbol{w}(\boldsymbol{x}), \boldsymbol{w}(\boldsymbol{x}')) \cdot k_2(\Phi(\xi), \Phi(\xi'))$, where k_1 is an ARD kernel for $\boldsymbol{w}(\boldsymbol{x})$ and k_2 for $\Phi(\xi)$ with their original hyperparameters, due to the separate structure of an ARD kernel. This decomposition significantly simplifies the optimization of acquisition function because $k_2(\Phi(\xi), \Phi(\xi'))$ need to be computed only once until a new target netlist ξ_* is given.

3.6 BoA-PTA for Solver Parameter Optimization

Most surrogate model based acceleration techniques require pre-computed data for pseudo-random inputs [45]. In contrast, BoA-PTA can be immediately deployed to explore the potential improvements for a netlist set $\Xi = [\xi_1, \ldots, \xi_L]$. We call this *cold start*, as the surrogate has yet seen any observations for any netlists and solver parameters to make an accurate posterior prediction. For this situation, we use a sequential iteration scheme to run BoA-PTA, which is described in Algorithm 1.

It might seem unnecessary to run Algorithm 1 because it requires repeated executions of the CEPTA solver and consumes extra computational resources. This process indeed provides little value for the task of solving netlists Ξ for once. However, BoA-PTA provides three significant extended values. First, it explores the potential improvements for the considered netlists. Unlike a random or grid search scheme, it provides a systematic and efficient way to continuously optimize CEPTA for future usage. In practice, those optimal solver parameters can be reused when slight modifications are made to the original netlist, which is a common situation in circuit optimization or yield estimation. We will discuss this further for practical acceleration in Section 3.7. Second, for some netlists, CEPTA does not converge with the default solver parameters. In this case, BoA-PTA has the potential to seek solver parameters that leads to convergence. This is practically useful for performance improvements for CEPTA. Third, the process is an effective and efficient way to conduct offline training for the surrogate model to directly predict optimal solver parameters for an unseen circuit/netlist in future usage.

3.7 BoA-PTA for Monte Carlo Acceleration

One of the most common situations for the repeated SPICE simulations is Monte Carlo analysis, where many modest variations of a given netlist are simulated. Denote Q as a netlist sampler that generates a netlist ξ based on a pre-defined distribution. Here we propose a possible method for BoA-PTA to accelerate such a Monte Carlo analysis in Algorithm 2. In this algorithm, we set

$y = 9{,}999$ as the penalty for a non-convergence case, $2y^*$ as the threshold to halt the BO process, and 20 epochs as a convergence threshold. This hyperparameter needs to be adjusted for different situations and computational allowance. Note that the pre-trained model of Algorithm 1 can be used for Algorithm 2 as a "warm start" that provides prior knowledge.

ALGORITHM 2: BoA-PTA Monte Carlo Acceleration

Input: Netlists sampler Q, number of samples N_{mc}

```
Sample a netlist ξ from Q and execute η(x, ξ)
Update the record of the best x* and the best iteration y*
for i = 2 to N_mc do
    Sample a netlist ξ from Q
    if y* does not converge then
        Update surrogate model M by maximizing (29)
        Optimize acquisition function and get optimal x
        Execute η(x, ξ)
        while Executing η(x, ξ) do
            if SPICE iteration reaches 2y* then
                Stop the current execution
                re-execute η(x*, ξ) and collect results
            end if
        end while
        Update record of best parameters x* and iteration y*
    else
        Execute η(x*, ξ) and collect results
    end if
end for
return Monte Carlo analysis for Q
```

3.8 Error Analysis and Computation Complexity

Unlike many verification/design acceleration solutions that are purely based on machine learning techniques [11] introducing unquantified error and uncertainty, BoA-PTA introduces no extra error or uncertainty. Specifically, as long as the CEPTA converges, the error is bounded by the error of PTA, which is ($\dot{u} < 10^{-12}$) by default. BoA-PTA is thus an error-free approach. When BoA-PTA fails to improve CEPTA and leads to a non-convergence situation, it is fully aware of such an error and can roll back to use the default setting.

Once the GP is trained, it takes only $O(N)$ and $O(N^2)$ (N is number of observations) for the computation of $\mu(\boldsymbol{x})$ and $v(\boldsymbol{x})$, respectively. The complexity of the DNN (depending on the network structure) is approximately $O(\sum_{l=1}^{L} M_l^2)$, where M_l is the number of units in the hidden layer l. The BetaCDF transformation computational cost is negligible.

For the training of a GP, the major computational cost is the matrix inversion $(\mathbf{K}+\sigma^2\mathbf{I})^{-1}$, which is $O(N^3)$, and the DNN forward computation for all observations, which is $O(N\sum_{l=1}^{L} M_l^2)$. We can see that BoA-PTA scales poorly with N, which hinders its further applications. In such a case, a variational sparse GP [36] can be implemented to resolve this issue, which is outside the scope of this work; we thus leave it as future work.

For practical SPICE simulations that can take up to several hours, BoA-PTA brings almost zero computational overhead until the number of samples grows very large. As discussed earlier, a scalable GP is then required.

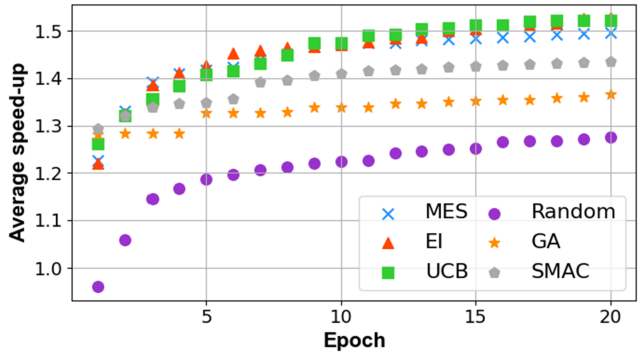


Fig. 6. Average speedup over 43 simulations for BoA-PTA with different acquisition functions (MES, EI, and UCB), GA, SMAC, and a random search scheme.

4 EXPERIMENTAL RESULTS

4.1 Benchmark Circuits and Experimental Setups

Most SPICE solvers have certain advantages on some particular circuits. To assess BoA-PTA thoughtfully, we test it on the circuit simulator benchmark set known as CircuitSim93 [3], which contains 43 classic circuits including BJT-, MOS2-, and MOS3-type circuits.

In this work, we have tested several features extracting DNNs, such as MLPs with two to five hidden layers, each of which consists of {4, 8, 16} neurons with sigmoid or ReLU activation functions. We did not find a significant performance difference, possibly due to the relatively small size of the training data for this problem. We believe that a ReLU activation function is more suitable for classification, and a deeper network is harder to train. Thus, we opt for a simple MLP architecture with two hidden layers containing 16 neurons to sustain enough model capacity. We use L-BFGS-B [54] with five iterations for both the GP fitting optimization and the acquisition optimization. We evaluate BoA-PTA with three acquisition functions (i.e., EI, MES, and UCB) with common $\beta = 0.1$. The implementation of BoA-PTA is based on PyTorch and BoTorch.[1]

4.2 BoA-PTA Acceleration Efficiency

Since BoA-PTA is intended to reduce the number of iterations of CEPTA, the speedup against CEPTA with default solver parameters is used as the performance metric. To further assess the acceleration efficiency through the BO framework, we compare the speedup of CEPTA when using **genetic algorithm (GA)**, **sequential model-based optimization for general algorithm configuration (SMAC)** [12], a SOTA baseline BO based on random forest, and a vanilla random search method as a reference. GA is implemented based on Pymoo,[2] whereas SMAC is implemented on SMAC.[3] The implementation details are given in the appendix for the sake of clarity. All optimization methods are given the default parameters and a randomly generated parameter as initial data, whereas the random search is not. We ran each method up to 20 epochs and recorded the average speedups over the 43 classic benchmark circuits for each epoch. Each method is repeated five times with different random seeds and thus the initial data. After excluding the non-convergence simulations, the average speedups over the 43 benchmark simulations for each epoch are shown in Figure 6.

[1] https://pytorch.org;https://botorch.org.
[2] https://pymoo.org.
[3] https://scikit-optimize.github.io.

Table 3. Solver Iterations for Benchmark Circuits

NAME	Properties								Performance (NR Iterations)					Speedup
bjt	**nodes**	**eqn**	**bjt**	**mos2**	**mos3**	**c**	**r**	**v**	**PPTA**	**CEPTA**	**MES**	**EI**	**UCB**	**Best**[4]
astabl	6	12	2	0	0	2	4	2	108	55	**46**	50	48	1.20
bias	12	55	13	0	0	0	5	4	N/A[5]	839	**239**	581	355	3.51
bjtff	48	177	41	0	0	1	26	6	N/A	169	102	90	**86**	1.97
bjtinv	26	40	12	0	0	0	24	2	125	186	104	**52**	121	3.85
latch	19	65	14	0	0	0	10	4	153	130	86	**84**	88	1.55
loc	326	739	96	0	0	12	276	5	N/A	N/A	N/A	N/A	N/A	N/A
nagle	26	54	23	0	0	1	11	5	2,440	**306**	**306**	**306**	**306**	1.00
opampal	71	518	148	0	0	4	28	3	2,335	N/A	830	1,000	**635**	∞[6]
optrans	270	1,860	528	0	0	19	148	6	N/A	2,206	**1,561**	2,118	2,283	1.41
rca	18	32	11	0	0	0	12	3	76	82	**55**	57	82	1.49
ring11	34	101	22	44	0	11	1	41	63	63	63	**51**	**41**	1.24
schmitecl	8	18	4	0	0	1	8	2	48	52	47	50	**44**	1.18
vreg	19	20	20	0	0	0	10	1	N/A	**22**	**22**	**22**	**22**	1.00
mos2	**nodes**	**eqn**	**bjt**	**mos2**	**mos3**	**c**	**r**	**v**	**PPTA**	**CEPTA**	**MES**	**EI**	**UCB**	
ab_ac	25	28	0	31	0	22	1	3	3,568	90	82	**79**	84	1.14
ab_integ	28	32	0	31	0	24	3	4	4,644	499	**460**	477	472	1.08
ab_opamp	28	31	0	31	0	24	4	3	5,765	150	130	**126**	**126**	1.19
cram	32	44	0	60	0	42	0	12	162	91	**90**	91	**90**	1.01
e1480	145	204	0	28	0	17	130	3	5,280	179	165	134	**118**	1.52
g1310	66	97	0	14	0	21	56	3	73	76	56	**48**	55	1.58
gm6	7	20	0	5	0	0	0	3	N/A	69	**44**	46	45	1.57
hussamp	14	17	0	16	0	2	1	3	193	91	88	**86**	91	1.06
mosrect	6	10	0	4	0	0	2	2	828	65	**54**	58	59	1.20
mux8	30	42	0	64	0	29	0	12	177	122	**90**	93	100	1.36
nand	17	19	0	25	0	0	0	2	N/A	83	55	56	**54**	1.54
pump	3	4	0	1	0	2	1	1	N/A	**22**	**22**	**22**	**22**	1.00
reg0	15	16	0	0	0	13	30	1	**22**	**22**	**22**	**22**	**22**	1.00
ring	18	19	0	34	0	1	0	1	**46**	N/A	N/A	N/A	N/A	N/A
schmitfast	5	19	0	6	0	0	0	2	5,647	82	71	69	**67**	1.22
schmitslow	7	25	0	8	0	0	0	2	N/A	127	96	**93**	108	1.37
slowlatch	12	37	0	0	14	0	1	5	9,445	169	163	163	**135**	1.25
toronto	25	36	0	0	58	33	0	11	N/A	**277**	**277**	**277**	**277**	1.00
mos3	**nodes**	**eqn**	**bjt**	**mos2**	**mos3**	**c**	**r**	**v**	**PPTA**	**CEPTA**	**MES**	**EI**	**UCB**	
arom	57	62	0	0	116	23	2	5	N/A	N/A	N/A	N/A	N/A	N/A
b330	163	856	0	0	330	0	0	33	N/A	N/A	N/A	N/A	N/A	N/A
counter	93	96	0	0	220	0	0	3	**22**	**22**	**22**	**22**	**22**	1.00
gm1	31	129	0	0	46	8	7	6	N/A	76	**74**	**74**	**74**	1.03
gm2	31	148	0	0	7	5	0	2	N/A	70	**47**	51	**47**	1.49
gm3	89	428	0	0	30	1	0	2	111	66	53	54	**51**	1.29
gm17	5	21	0	0	56	7	3	5	N/A	212	**185**	197	192	1.15
gm19	17	79	0	0	162	83	1	15	N/A	N/A	**256**	2983	N/A	∞
jge	180	243	0	0	348	157	1	63	N/A	1215	829	841	**801**	1.52
mike2	11	38	0	0	12	1	0	5	103	189	78	80	**57**	3.32
rich3	51	56	0	0	106	12	2	5	N/A	N/A	N/A	N/A	N/A	N/A
todd3	13	43	0	0	13	0	1	6	9428	554	219	**105**	133	5.28

We can see that BoA-PTA outperforms other competitors with a large margin. In detail, BoA-PTA with different acquisition functions shows similar performance; SMAC and GA show similar performance to BoA-PTA at the beginning of low epochs. As the epoch grows, the difference is enlarged. Note that GA improves speedup slowly, possibly because GA needs a large amount of

[4]Best speedup of BoA-PTA with MSE, EI, and UCB against CEPTA.
[5]N/A indicates that the SPICE solver does not converge.
[6]∞ indicates that BoA-PTA turns a non-convergence into convergence.

Table 4. Solver Iterations for Benchmark Circuits Compared with a Commercial Simulator

NAME	Spectre-ptran	Spectre-dptran	MES	EI	UCB
opampal	1,238	1,240	830	1,000	635
optrans	1,946	1,881	1,561	2,118	2,283
gm19	245	247	256	2,983	N/A

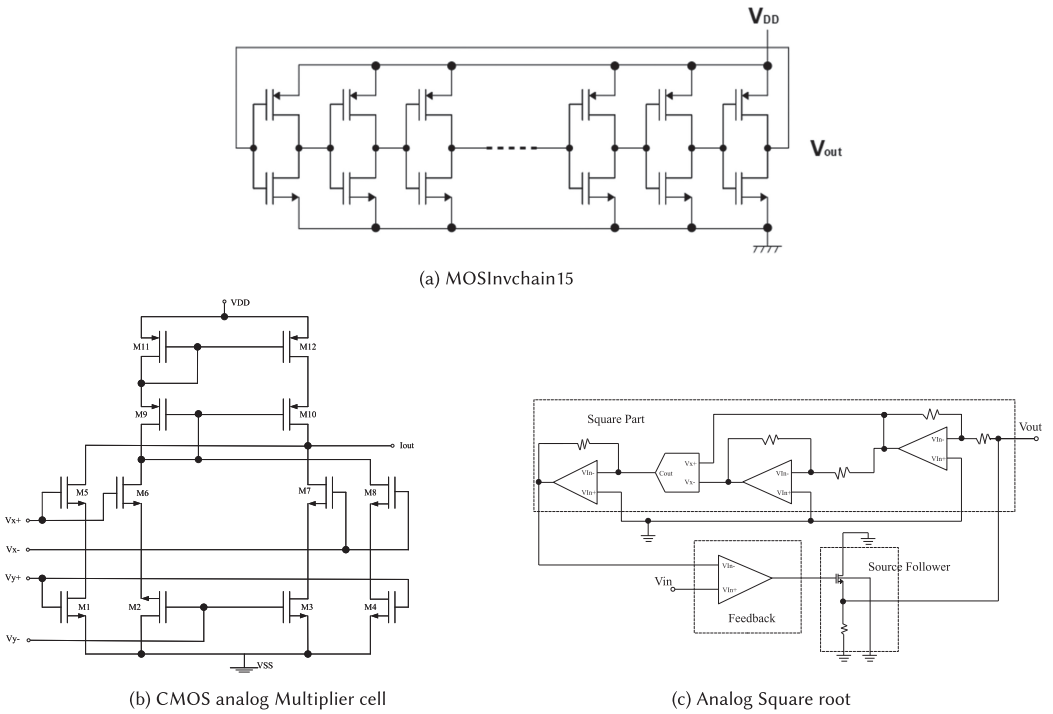
epochs to fully test its mutations and is thus not very efficient in this scenario, where we increase the number of simulations slowly.

We show the detailed acceleration with 20 epochs for all benchmark simulations in Table 3. All DPTA and RPTA results are worse than CEPTA and are not presented due to limited space. We also highlight the best speedup of BoA-PTA by different acquisition functions. The first thing we notice in Table 3 is that BoA-PTA outperforms the best PTA method CEPTA in all cases except for pump and reg0, in which the performance is equal, indicating that no improvements can be made by searching the solver parameter space. This clearly demonstrates that BoA-PTA improves the CEPTA solver without degeneration. Another interesting thing to point out is that for the CEPTA non-convergence cases of {opampal, optrans, gm19}, BoA-PTA makes them converge! This is particularly useful for PTA-based SPICE, as they often suffer from non-convergence issues. In general, tuning a PTA solver to converge is extremely difficult because no gradient or space geometry information can be inferred from the non-convergence data. Even with expert knowledge, this process is time consuming and there is no guarantee for success. We also notice that the speedup for these 43 circuits has quite a large variance, indicating the difficulty in accelerating some circuits. This does not weaken the practicality of BoA-PTA, especially when we are facing complex simulations that can take a large number of iterations to converge.

We have also compared to Newton's method, Gmin stepping, and the commonly used SPICE solver, Ngspice, which uses Newton's method as the first attempt and Gmin stepping when Newton's method fails. One may notice that PTA-based methods are outperformed many times. This is not surprising because homotopy and Newton methods are particularly good for small-scale simple circuits but scale poorly to large-scale real-world circuits due to non-convergence issues [48]. PTA-based methods are often used as the safeguard when most ordinary methods, such as homotopy methods, fail the DC analysis; they are known to be robust to large-scale problems but often suffer from slow convergence issues, leading to a large number of iterations. BoA-PTA can improve CEPTA so much that it can match or outperform Ngspice (e.g., in mike2). This is a remarkable advances for the PTA-based methods.

To also compare with the SOTA commercial PTA solver, we run Spectre, which implements two PTA-type methods—ptran and dptran—for the benchmark circuits. As we cannot control the behaviors of Spectre, we only compare the results where ptran and dptran are executed when NR and Gmin fail. The results are shown in Table 4. We can see that BoA-PTA with an MES acquisition outperforms Spectre for opampal and optrans and shows approximately good performance for gm19. BoA-PTA with EI and UCB acquisition do not have as good performance, indicating the importance of choosing a proper acquisition function.

To access the usability in real applications, we test the improvements over four classic analog circuit designs, namely MOSInvchain15, Multiplier, Square root, and Suntraction, where MOSInvchain15 is a CMOS transistor inverter circuit, whereas the other three circuits are from a classic automobile intake system [27], which are designed to be highly accurate under high-speed processing calculations. Similarly, we are interested in the speedup performance within 20 epochs for the testing circuits for BoA-PTA, GA, SMAC, and random search.



(a) MOSInvchain15

(b) CMOS analog Multiplier cell

(c) Analog Square root

Fig. 7. Schematic illustration of testing circuits.

The MOSInvchain15 is a 15-stage connected CMOS transistor inverter chain circuit that is a typical ring oscillator with 30 MOS transistors and $V_{DD} = 5$V. Each stage consists of two complementary MOS transistors: an NMOS and a PMOS. It has a very high loop gain with positive feedback and is quite easy to oscillate by using the pseudo-transient analysis method. The circuit design of MOSInvchain15 is shown in Figure 7(a) and the performance in Figure 8(a). For the MOSInvchain15, BoA-PTA with the UCB acquisition function can improve the performance up to about 250x just with 10 iterations, whereas BoA-PTA with the EI acquisition function has a much earlier improvement (e.g., 210x with only 4 iterations). In contrast, the random search, as well as GA and SMAC, fails to improve any performance even with 20 iterations, which suggests that BoA-PTA indeed is a powerful method for optimizing solver parameters.

The Multiplier is a highly accurate multiplying calculation circuit essentially used in an automobile intake system. The circuit is a real design for the Toyota automobile's control engine, which is implemented by a high linear CMOS four-quadrant multiplier cell as shown in Figure 7(b). The Multiplier circuit includes four voltages sources, five capacitors, 14 resistors, and 57 MOSFETs (33 NMOS and 24 PMOS). The improved performances are shown in Figure 8(b). It is clear that BoA-PTA improves the performance significantly especially with the UCB and EI acquisition functions. This result is slightly different from the others because the EI normally was outperformed by the MES and UCB. However, this phenomenon is consistent with the BO literature that no acquisition is suitable for all problems. The random search method seems to work well in this case, as it converges to a similar level of speedup eventually. However, we should point out that the random search takes about 13 epochs to converge, whereas BoA-PTA with UCB takes about 2 epochs, indicating an almost 7x higher efficiency. It is also noticeable that random search outperforms GA and

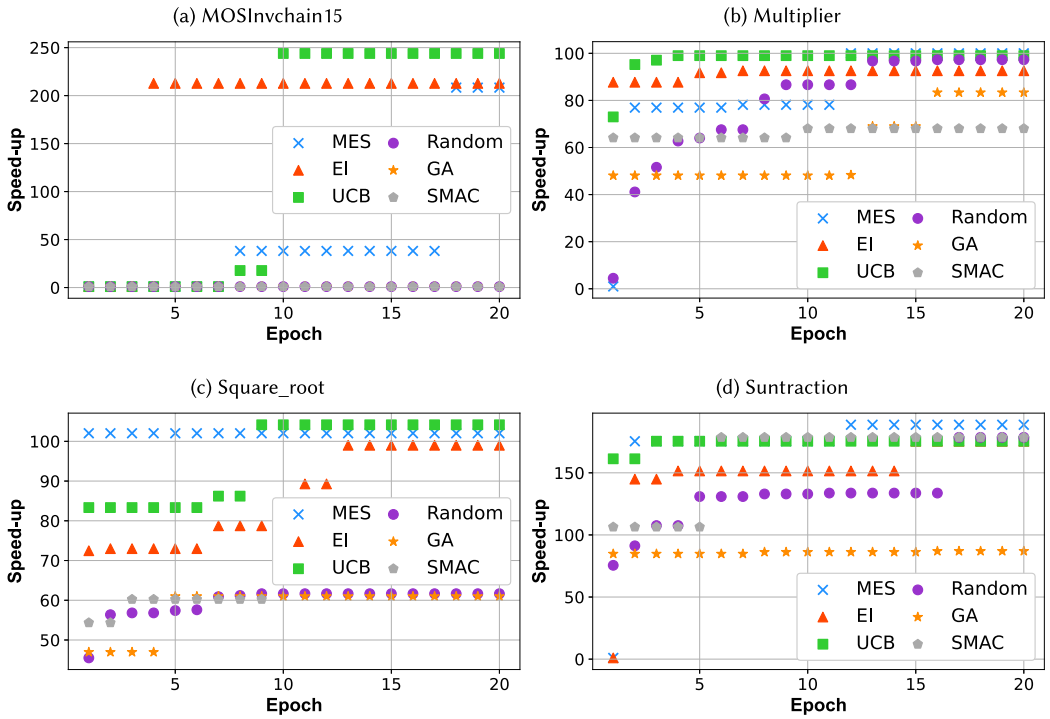


Fig. 8. Speedup for four testing circuits using BoA-PTA with acquisition functions (MES, EI, and UCB), GA, SMAC, and a random search scheme.

SMAC with a high epoch, possibly due to local optimal issues particularly for a heuristic method such as GA.

The Square root is a highly accurate root square calculation circuit also used in the automobile intake system. The circuit includes the square part, operational amplifier, and source follower to ensure the output of square root to always be correct even if the input signal enters the negative region. It has 76 devices including three voltage sources, 14 capacitors, 10 resistors, and 49 MOSFETs. The circuit design of the Square root circuit is illustrated in Figure 7(c) and the improved performance in Figure 8(c). In this experiment, BoA-PTA with MES acquisition outperforms the other methods significantly with just 1 epoch. The UCB achieves similar performance with 9 epochs. Despite the slower convergence rate, BoA-PTA with UCB keeps providing stable speedup as in other experiments. The random search method, GA, and SMAC again fail this task by converging to a much lower speedup improvement with 20 epochs.

The Suntraction circuit [27] is an analog circuit designed for high-speed and highly accurate subtraction operation, which includes 31 devices composed of four voltages sources, seven resistors, two capacitors, and 18 MOSFETs. The improved performance is shown in Figure 8(d). We can see that BoA-PTA with the MES acquisition function converges to the best speedup, whereas BoA-PTA with UCB and EI acquisition provide slightly worse performances, which are the same as the one achieved by the random search method with 17 epochs. However, note that this performance is achieved by BoA-PTA with UCB with 3 epochs and BoA-PTA with MES with 2 epochs. Despite that the random search method converges to the same speedup level, it shows a significantly slower convergence rate, which is fatal for improving SPICE performance. GA struggles to improve performance despite that the first attempt with 1 epoch shows very good performance.

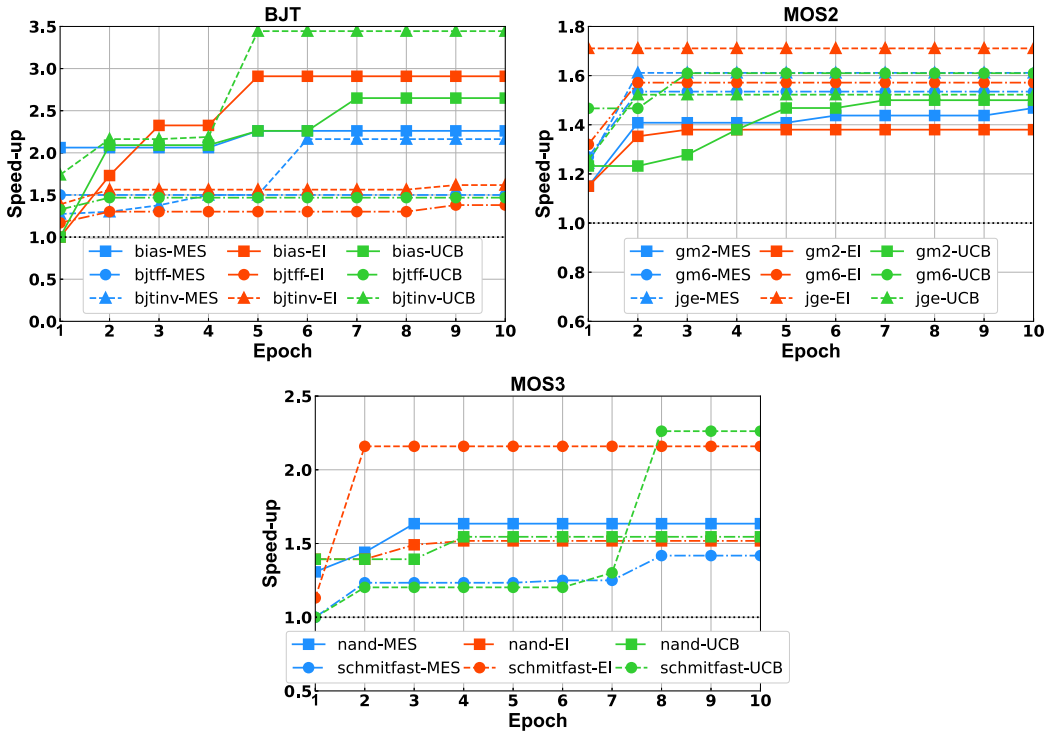


Fig. 9. Speedup for unseen BJT-, MOS2-, and MOS3-type circuit simulations.

SMAC also shows high speedup at the beginning and improves itself slowly toward the possible best performance. It is, however, consistently slower than BoA-PTA with MES and UCB.

Overall, for the four practical circuit designs, BoA-PTA, no matter which acquisition functions it used, showed significant improvement in terms of convergence rate and consistency over random search, GA, and SMAC. BoA-PTA with UCB provides the best overall performance—that is, a 250x speedup for the MOSInvchain15 with just 10 epochs and about 80x to 160x speedup for the other classic calculation circuit design with only 1 epoch. In contrast, the random search and GA either fail to improve the performance or show a much slower convergence rate. SMAC is a potential candidate acceleration framework. However, without a proper redesign of its surrogate model for this particular problem, the vanilla SMAC shows inferior performance in terms of the final improvement, converging speed, and robustness.

4.3 Optimal Predictions for Unseen Simulations

In this experiment, we pick the circuit with large potential for improvements among all types of circuits (i.e., $\mathcal{T}$ ={bias, bjtff, bjtinv, gm2, gm6, jge, nand, schmitfast}) and use them as testing simulations for BoA-PTA. Specifically, simulations that are not in $\mathcal{T}$ are used as the training simulations and used as input for Algorithm 1. At the end of each epoch, we use BoA-PTA to predict solver parameters for $\mathcal{T}$ and evaluate their speedups. We emphasize that the evaluations of $\mathcal{T}$ are never updated to BoA-PTA. They are strictly treated as testing data. The results are shown in Figure 9.

In this case, we do not compare BoA-PTA with a random search optimization because there is no way for it to predict the optimal solver parameters. This is indeed one of the main novelties of BoA-PTA. As we can see in Figure 9, BoA-PTA can further improve the SPICE speedup with an increasing number of epochs even the circuits have never been seen by the system. This

Table 5. Monte Carlo Simulation Statistics

Circuit		DPTA	RPTA	PPTA	CEPTA	MES	EI	UCB
bias	#NC	3,913	88	5,710	9	**0**	**0**	**0**
	Mean	9,993	722	5,644	913	**346**	359	484
	STD	29,980	4,342	8,395	**51**	191	465	2,964
bjtinv	#NC	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	Mean	73.9	73.8	124.5	55.3	**49.5**	53.8	48.8
	STD	14.4	14.8	10.4	**3.3**	12.3	13.9	10.3
bjtff	#NC	6,000	6,000	6,000	**0**	**0**	**0**	**0**
	Mean	N/A	N/A	N/A	138	**89**	93	93
	STD	N/A	N/A	N/A	**0**	13.5	20.0	4.8

essentially indicates that the DNN feature extractions indeed work with BoA-PTA as expected such that knowledge from the training circuits can be transferred to the testing circuits. The BJT-type and MOS3-type circuit simulations show a larger potential for improvements, whereas the improvement for the MOS2-type circuit simulation is less significant. The EI acquisition shows stable improvement with only three epochs for all three types of simulations in this experiment. The MES, however, shows slower improvement.

4.4 Monte Carlo Accelerations

Last, we assess BoA-PTA in Monte Carlo accelerations as described in Algorithm 2. Similarly, we use the BJT circuit with potential for improvements (i.e., bias, bjtinv, and bjtff) as testing examples. Monte Carlo simulation is designed to analyze the statistical properties when all registers in a circuit have independent {1%, 2%, 5%, 10%, 20%} variation of normal distribution (i.e., $R = R_{\text{original}} * \mathcal{N}(1, \text{variation}^2)$). For each variation set, each method is tested on the same 1,000 random sampled netlists to provide a fair comparison. Since BoA-PTA is error free as discussed, we did not show the statistical results but focused on the runtime statistics. We first show the number of non-convergence (#NC), the mean, and the **standard deviation (STD)** iterations for 6,000 total simulations in Table 5. We can see clearly that BoA-PTA always converges and always provides minimal iterations. Among different acquisition functions, BoA-PTA with MES consistently shows the best performance, which is consistent with the observation in previous experiments. CEPTA always has the lowest STD of iterations. We argue that what matters most is the total iterations, not the deviation for the runtime. In addition, BoA-PTA can overcome a few non-converge simulations in the bias circuits. The other PTA solvers are way worse than BoA-PTA and CEPTA, which is consistent with previous results.

The average iterations number (over 6,000 simulations) is shown in Figure 10 without DPTA, RPTA, and PPTA due to their high non-convergence rate. We can see that any acquisition function can improve the standard CEPTA for a large margin using the BoA-PTA framework. BoA-PTA with MES overall obtains the most stable and good performance with approximately 2x speedup over CEPTA.

5 CONCLUSION

In this article, a BO SPICE acceleration is proposed, and it is demonstrated using BoA-PTA, a combination with the CEPTA solver. By harnessing the advantages of modern machine learning techniques, BoA-PTA demonstrates substantial improvement over the original CEPTA solver—up to 3.5x speedup for 43 benchmark circuit simulations, up to 250x speedup for four practical circuit

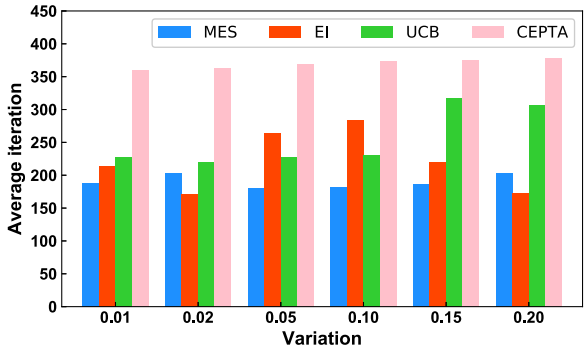


Fig. 10. Average number of iterations (over bias, bjtinv, and bjtff) for Monte Carlo simulations with different variations.

designs, and up to 2x speedup for Monte Carlo simulations based on circuits from the 43 benchmark circuits. Further improvements of this work include a graph neural network that extracts features directly from a netlist, scalable GP surrogate, and efficient BO via portfolio strategies. Direct acceleration for CEPTA and other PTA-based methods may be explored by improving its step scheme using local optimization [6] and reinforcement learning [15].

APPENDICES

A DERIVATION OF THE ELBO

We utilize Jensen's inequality to derive an ELBO of the joint model log-likelihood given the observation set $\{\boldsymbol{x}_i, \boldsymbol{\xi}_i, y_i\}_{i=1}^N$,

$$\begin{aligned}
&\log \int p(\boldsymbol{y}, \boldsymbol{\gamma}|\boldsymbol{x}, \boldsymbol{\theta}) d\boldsymbol{\gamma} \\
&= \log \int q(\boldsymbol{\gamma}) \frac{p(\boldsymbol{y}, \boldsymbol{\gamma}|\boldsymbol{x}, \boldsymbol{\theta})}{q(\boldsymbol{\gamma})} d\boldsymbol{\gamma} \\
&\geq \int q(\boldsymbol{\gamma}) \log \frac{p(\boldsymbol{y}, \boldsymbol{\gamma}|\boldsymbol{x}, \boldsymbol{\theta})}{q(\boldsymbol{\gamma})} d\boldsymbol{\gamma} \\
&= \int q(\boldsymbol{\gamma}) \log p(\boldsymbol{y}, \boldsymbol{\gamma}|\boldsymbol{x}, \boldsymbol{\theta}) d\boldsymbol{\gamma} - \int q(\boldsymbol{\gamma}) \log q(\boldsymbol{\gamma}) d\boldsymbol{\gamma} \\
&= \int q(\boldsymbol{\gamma}) \log p(\boldsymbol{y}|\boldsymbol{\gamma}, \boldsymbol{x}, \boldsymbol{\theta}) d\boldsymbol{\gamma} - \mathrm{KL}\left(q(\boldsymbol{\gamma})||p(\boldsymbol{\gamma})\right) \triangleq \mathcal{L},
\end{aligned} \tag{36}$$

where $\mathrm{KL}\left(q(\boldsymbol{\gamma})||p(\boldsymbol{\gamma})\right) = \int q(\boldsymbol{\gamma}) \log \frac{q(\boldsymbol{\gamma})}{p(\boldsymbol{\gamma})} d\boldsymbol{\gamma}$ is the KL distance between $q(\boldsymbol{\gamma})$ of Equation (26) and $p(\boldsymbol{\gamma})$ of Equation (25), which has a closed-form solution given our prior and variational posterior.

B IMPLEMENTATION DETAILS OF GA AND SMAC

For GA, the solid multi-objective algorithm (NSGA2) function in the Pymoo library is utilized. We use a uniform distribution for the sampling, a simulated binary crossover (SBX) with $p = 0.9$ and $\eta = 15$, and a polynomial mutation with $\eta = 20$ for the configuration of GA. The rest of the parameters use the default settings in Pymoo. For SMAC, the forest minimizing function of the scikit-optimize library is used. Extra trees regressor is utilized as the surrogate model. We use EI as the acquisition function, whereas the initial point generator is set to random with two initial points. The rest of the parameters use the default parameters in SMAC.

Table 6. Solver Iterations for Benchmark Circuits

NAME	Performance (NR Iterations)							
bjt	**Ngspice**	**NR**	**Gmin**	**PPTA**	**CEPTA**	**MES**	**EI**	**UCB**
astabl	10	10	35	108	55	46	50	48
bias	69	69	81	N/A4	839	239	581	355
bjtff	64	64	126	N/A	169	102	90	86
bjtinv	N/A	N/A	N/A	125	186	104	52	121
latch	13	13	47	153	130	86	84	88
loc	25	25	105	N/A	N/A	N/A	N/A	N/A
nagle	7	7	38	2,440	306	306	306	306
opampal	8	8	40	2,335	N/A	830	1,000	635
optrans	163	N/A	178	N/A	2,206	1,561	2,118	2,283
rca	38	38	37	76	82	55	57	82
ringll	38	38	202	63	63	63	51	41
schmitecl	13	13	142	48	52	47	50	44
vreg	2,357	N/A	438	N/A	22	22	22	22
mos2	**Ngspice**	**NR**	**Gmin**	**PPTA**	**CEPTA**	**MES**	**EI**	**UCB**
ab ac	19	19	52	3,568	90	82	79	84
ab integ	19	19	63	4,644	499	460	477	472
ab opamp	142	N/A	165	5,765	150	130	126	126
cram	10	10	38	162	91	90	91	90
e1480	9	9	35	5,280	179	165	134	118
g1310	N/A	N/A	N/A	73	76	56	48	55
gm6	17	17	39	N/A	69	44	46	45
hussamp	15	15	45	193	91	88	86	91
mosrect	23	23	44	828	65	54	58	59
mux8	6	6	25	177	122	90	93	100
nand	3	3	25	N/A	83	55	56	54
pump	18	18	46	N/A	22	22	22	22
reg0	28	28	51	22	22	22	22	22
ring	89	89	62	46	N/A	N/A	N/A	N/A
schmitfast	13	13	41	5,647	82	71	69	67
schmitslow	7	7	N/A	N/A	127	96	93	108
slowlatch	9	9	58	9,445	169	163	163	135
toronto	37	37	37	N/A	277	277	277	277
mos3	**Ngspice**	**NR**	**Gmin**	**PPTA**	**CEPTA**	**MES**	**EI**	**UCB**
arom	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
b330	16	16	28	N/A	N/A	N/A	N/A	N/A
counter	9	9	34	22	22	22	22	22
gm1	9	9	39	N/A	76	74	74	74
gm2	50	50	58	N/A	70	47	51	47
gm3	34	34	60	111	66	53	54	51
gm17	9	9	34	N/A	212	185	197	192
gm19	11	11	28	N/A	N/A	256	2,983	N/A
jge	N/A	N/A	N/A	N/A	1,215	829	841	801
mike2	180	N/A	N/A	103	189	78	80	57
rich3	36	36	29	N/A	N/A	N/A	N/A	N/A
todd3	156	N/A	55	9,428	554	219	105	133

C COMPARISON TO NGSPICE

We have also compared to Newton's method, Gmin stepping, and the commonly used SPICE solver, Ngspice, which uses Newton's method as the first attempt and Gmin stepping where Newton's method fails. The results are shown in Table 6. The experimental setup is the same as that in Section 4.2. We can see that Ngspice, NR, and Gmin outperform BoA-PTA, and certainly CEPTA, many times. This is an expected result because the PTA-based methods are known to be slow but

stable. They are designed to provide reliable DC analysis when ordinary methods fail; they serve as a complement instead of a replacement for the ordinary and well-known methods. A side evidence we can see is that Gmin is also not as good as the most fundamental NR. By formulating the DC analysis from a different perspective, Gmin shows its advantage of convergence for particular circuits at the price of higher iteration. Thus, Gmin is a very important complement to Ngspice as its second-choice candidate when NR fails. The same philosophy applies to PTA-based methods, which serve as backups where most of the ordinary methods fail. Because the PTA-based method often suffers from slow convergence, improving the PTA-based methods using machine learning is practical and useful (as we have shown in the practical MC experiment in Section 4.4).

REFERENCES

[1] Ryan Prescott Adams and Oliver Stegle. 2008. Gaussian process product models for nonparametric nonstationarity. In *Proceedings of the 25th International Conference on Machine Learning*. 1–8.

[2] Meysam Akbari, Omid Hashemipour, and Farshad Moradi. 2018. Input offset estimation of CMOS integrated circuits in weak inversion. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 26, 9 (2018), 1812–1816.

[3] J. A. Barby and R. Guindi. 1993. CircuitSim93: A circuit simulator benchmarking methodology case study. In *Proceedings of the 6th Annual IEEE International ASIC Conference and Exhibit*. 531–535.

[4] Christopher M. Bishop. 2007. *Pattern Recognition and Machine Learning*. Springer.

[5] Luke Bornn, Gavin Shaddick, and James V. Zidek. 2012. Modeling nonstationary processes through dimension expansion. *Journal of the American Statistical Association* 107, 497 (2012), 281–289.

[6] Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press.

[7] Eric Brochu, Vlad M. Cora, and Nando De Freitas. 2010. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*.

[8] Xiaoming Chen, Yu Wang, and Huazhong Yang. 2017. *Parallel Sparse Direct Solver for Integrated Circuit Simulation*. Springer.

[9] Michael Günther and Uwe Feldmann. 1995. The DAE-index in electric circuit simulation. *Mathematics and Computers in Simulation* 39, 5–6 (1995), 573–582.

[10] José Miguel Hernández-Lobato, Matthew W. Hoffman, and Zoubin Ghahramani. 2014. Predictive entropy search for efficient global optimization of black-box functions. In *Advances in Neural Information Processing Systems* 27 (2014), 918–926.

[11] Guyue Huang, Jingbo Hu, Yifan He, Jialong Liu, Mingyuan Ma, Zhaoyang Shen, Juejian Wu, et al. 2021. Machine learning for electronic design automation: A survey. *ACM Transactions on Design Automation of Electronic Systems* 26, 5 (2021), Article 40, 46 pages.

[12] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2011. Sequential model-based optimization for general algorithm configuration. In *Proceedings of the International Conference on Learning and Intelligent Optimization*. 507–523.

[13] Zhou Jin, Xiao Wu, Dan Niu, Xiaoli Guan, and Yasuaki Inoue. 2015. Effective ramping algorithm and restart algorithm in the SPICE3 implementation for DPTA method. *Nonlinear Theory and Its Applications, IEICE* 6, 4 (2015), 499–511.

[14] Donald R. Jones, Matthias Schonlau, and William J. Welch. 1998. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization* 13, 4 (1998), 455–492.

[15] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. 1996. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* 4 (1996), 237–285.

[16] Marc C. Kennedy and Anthony O'Hagan. 2001. Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63, 3 (2001), 425–464.

[17] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[18] Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational Bayes. In *Proceedings of the International Conference on Learning Representations (ICLR'14)*.

[19] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

[20] Bo Liu, Francisco V. Fernández, and Georges Gielen. 2010. An accurate and efficient yield optimization method for analog circuits based on computing budget allocation and memetic search technique. In *Proceedings of the 2010 Design, Automation, and Test in Europe Conference and Exhibition (DATE'10)*. 1106–1111.

[21] Wenlong Lyu, Fan Yang, Changhao Yan, Dian Zhou, and Xuan Zeng. 2018. Batch Bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design. In *Proceedings of the International Conference on Machine Learning*. 3306–3314.

[22] Yuzhe Ma, Haoxing Ren, Brucek Khailany, Harbinder Sikka, Lijuan Luo, Karthikeyan Natarajan, and Bei Yu. 2019. High performance graph convolutional networks with applications in testability analysis. In *Proceedings of the 56th Annual Design Automation Conference (DAC'18)*.

[23] Jonas Mockus. 2012. *Bayesian Approach to Global Optimization: Theory and Applications*. Vol. 37. Springer Science & Business Media.
[24] L. W. Negel. 1975. *A Computer Program to Simulate Semiconductor Circuits*. College of Engineering, University of California.
[25] Bipul Chandra Paul and Kaushik Roy. 2002. Testing cross-talk induced delay faults in static CMOS circuit through dynamic timing analysis. In *Proceedings of the International Test Conference*. IEEE, Los Alamitos, CA, 384–390.
[26] Carl Edward Rasmussen and Christopher K. I. Williams. 2006. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA.
[27] Haoxing Ren, George F. Kokai, Walker J. Turner, and Ting-Sheng Ku. 2020. ParaGraph: Layout parasitics and device parameter prediction using graph neural networks. In *Proceedings of the 2020 57th ACM/IEEE Design Automation Conference (DAC'20)*. IEEE, Los Alamitos, CA, 1–6.
[28] Mohammad Ali Rezvani, Mokhtar Alinia Asli, Sahar Khandan, Hossein Mousavi, and Zahra Shokri Aghbolagh. 2017. Synthesis and characterization of new nanocomposite CTAB-PTA@ CS as an efficient heterogeneous catalyst for oxidative desulphurization of gasoline. *Chemical Engineering Journal* 312 (2017), 243–251.
[29] Jaijeet Roychowdhury and Robert Melville. 2005. Delivering global DC convergence for large mixed-signal circuits via homotopy/continuation methods. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25, 1 (2005), 66–78.
[30] Warren Scott, Peter Frazier, and Warren Buckler Powell. 2011. The correlated knowledge gradient for simulation optimization of continuous parameters using Gaussian process regression. *SIAM Journal on Optimization* 21, 3 (2011), 996–1026.
[31] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando De Freitas. 2015. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE* 104, 1 (2015), 148–175.
[32] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. 2012. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*. 2951–2959.
[33] Jasper Snoek, Kevin Swersky, Rich Zemel, and Ryan Adams. 2014. Input warping for Bayesian optimization of non-stationary functions. In *Proceedings of the International Conference on Machine Learning*. 1674–1682.
[34] Niranjan Srinivas, Andreas Krause, Matthias Seeger, and Sham M. Kakade. 2010. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on Machine Learning*. 1015–1022.
[35] E. J. W. ter Maten, Theo G. J. Beelen, Alex de Vries, and Maikel van Beurden. 2012. Robust time-domain source stepping for DC-solution of circuit equations. In *Proceedings of Scientific Computing in Electrical Engineering (SCEE'12)*. 39–40.
[36] Michalis K. Titsias. 2009. Variational learning of inducing variables in sparse Gaussian processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*. 567–574.
[37] Ljiljana Trajkovic. 2012. DC operating points of transistor circuits. *Nonlinear Theory Its Applications, IEICE* 3, 3 (2012), 287–300.
[38] Diego Ernesto Cortés Udave, Jan Ogrodzki, and G. M. A. de Anda. 2012. DC large-scale simulation of nonlinear circuits on parallel processors. *International Journal of Electronics and Telecommunications* 58, 3 (2012), 285–295.
[39] Akio Ushida, Yoshihiro Yamagami, Yoshifumi Nishio, Ikkei Kinouchi, and Yasuaki Inoue. 2002. An efficient algorithm for finding multiple DC solutions based on the SPICE-oriented Newton homotopy method. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 21, 3 (2002), 337–348.
[40] A. Ushida, Y. Yamagami, Y. Nishio, I. Kinouchi, and Y. Inoue. 2006. An efficient algorithm for finding multiple DC solutions based on the SPICE-oriented Newton homotopy method. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 21, 3 (2006), 337–348.
[41] Laung-Terng Wang, Yao-Wen Chang, and Kwang-Ting Tim Cheng. 2009. *Electronic Design Automation: Synthesis, Verification, and Test*. Morgan Kaufmann.
[42] Zi Wang and Stefanie Jegelka. 2017. Max-value entropy search for efficient Bayesian optimization. In *Proceedings of the 34th International Conference on Machine Learning, Volume 70*. 3627–3635.
[43] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. 2016. Deep kernel learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*. 370–378.
[44] Xiao Wu, Zhou Jin, Dan Niu, and Yasuaki Inoue. 2014. A PTA method using numerical integration algorithms with artificial damping for solving nonlinear DC circuits. *Nonlinear Theory and Its Applications, IEICE* 5, 4 (2014), 512–522.
[45] W. W. Xing, V. Triantafyllidis, A. A. Shah, P. B. Nair, and Nicholas Zabaras. 2016. Manifold learning for the emulation of spatial fields from computational models. *Journal of Computational Physics* 326 (2016), 666–690.
[46] Kiyotaka Yamamura and Wataru Kuroki. 2006. An efficient and globally convergent homotopy method for finding DC operating points of nonlinear circuits. In *Proceedings of the 2006 Asia and South Pacific Design Automation Conference (ASP-DAC'06)*. IEEE, Los Alamitos, CA, 408–415. https://doi.org/10.1145/1118299.1118400

[47] K. Yamamura and T. Sekiguchi. 1999. A fixed-point homotopy method for solving modified nodal equations. *IEEE Transactions on Circuits and Systems I Fundamental Theory Applications* 46, 6 (1999), 654–665.

[48] E. Yilmaz and M. M. Green. 1999. Some standard SPICE DC algorithms revisited: Why does SPICE still not converge? In *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems (ISCAS'99)*, Vol. 6. 286–289. https://doi.org/10.1109/ISCAS.1999.780151

[49] Ecevit Yilmaz and Michael M. Green. 1999. Some standard SPICE DC algorithms revisited: Why does SPICE still not converge? In *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems (ISCAS'99)*, Vol. 6. IEEE, Los Alamitos, CA, 286–289.

[50] Hong Yu, Yasuaki Inoue, Kazutoshi Sako, Xiaochuan Hu, and Zhangcai Huang. 2007. An effective SPICE3 implementation of the compound element pseudo-transient algorithm. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 90-A, 10 (2007), 2124–2131.

[51] Shuhan Zhang, Wenlong Lyu, Fan Yang, Changhao Yan, Dian Zhou, and Xuan Zeng. 2019. Bayesian optimization approach for analog circuit synthesis using neural network. In *Proceedings of the 2019 Design, Automation, and Test in Europe Conference and Exhibition (DATE'19)*. 1463–1468.

[52] Shuhan Zhang, Wenlong Lyu, Fan Yang, Changhao Yan, Dian Zhou, Xuan Zeng, and Xiangdong Hu. 2019. An efficient multi-fidelity Bayesian optimization approach for analog circuit synthesis. In *Proceedings of the 2019 56th ACM/IEEE Design Automation Conference (DAC'19)*. IEEE, Los Alamitos, CA, 1–6.

[53] Shuhan Zhang, Fan Yang, Dian Zhou, and Xuan Zeng. 2020. An efficient asynchronous batch Bayesian optimization approach for analog circuit synthesis. In *Proceedings of the 2020 57th ACM/IEEE Design Automation Conference (DAC'20)*. IEEE, Los Alamitos, CA, 1–6.

[54] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. 1997. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software* 23, 4 (1997), 550–560.

Received 11 March 2022; revised 31 May 2022; accepted 24 July 2022