BoA-PTA, A Bayesian Optimization Accelerated Error-Free SPICE Solver

1st Wei W. Xing
School of Integrated Circuit Science and Engineering
Beihang University
Beijing, China
wxing@buaa.edu.cn

2nd Xiang Jin
School of Integrated Circuit Science and Engineering
Beihang University
Beijing, China
jinxiang1114@163.com

3rd Yi Liu
Department of Computer Science and Technology
University of Petroleum-Beijing
Beijing, China
enchantedlllll717@163.com

4th Dan Niu
School of Automation
Southeast university
Changsha, China
danni1@163.com

5th Weishen Zhao
School of Integrated Circuit Science and Engineering
Beihang University
Beijing, China
wxing@buaa.edu.cn

6th Zhou Jin
Department of Computer Science and Technology
University of Petroleum-Beijing
Beijing, China
jinzhou@cup.edu.cn

Abstract—One of the greatest challenges in IC 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 solver. 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 be implemented to speed up ongoing repeated simulations immediately or to improve new simulations of completely different circuits. BoA-PTA is equipped with cutting-edge machine learning techniques, e.g., deep learning, Gaussian process, Bayesian optimization, non-stationary monotonic transformation, and variational inference via reparameterization. We assess BoA-PTA in 43 benchmark circuits against other SOTA SPICE solvers and demonstrate an average 2.3x (maximum 3.5x) speed-up over the original CEPTA.

Index Terms—Bayesian optimization, Gaussian process, Deep learning, SPICE, PTA, CEPTA, Circuit simulation

With increasing degrees of the integration of modern integrated circuits (IC), the reliability of a chip design is improved via a time-consuming verification process before it can be taped-out!!FIXME!! [5]. This poses a great challenge as the verification can take up to 80% of the development time in an IC design [6].

Due to its recent fast development, machine learning and other statistical learning methods have been utilized to resolve this challenge [7]. For instance, Bayesian optimization [4], multi-fidelity modelling [8], and computing budget allocation [9] 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 forms, putting the verification process in great risk. Thus, the 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 differential algebraic equations that are constructed on a circuit base on Kirchhoff’s current law (KCL) and Kirchhoff’s voltage law (KVL) [10]. The solution provides direct current (DC) analysis, which supports other detailed analysis, e.g., transient analysis and small signal analysis [11]. A general SPICE utilizes Newton-Raphson (NR) iteration and some continuation methods, e.g., Gmin stepping [12] and source stepping [13], to solve the nonlinear equations due to their fast convergence properties. However, they may fail to converge when the circuit scale is sufficiently large and especially with strong nonlinearity design, which brings high loop gain, positive feedback, or multiple solutions [14]. This challenge is well resolved by pseudo-transient analysis

---

The text continues with further details about the proposed method, its implementation, and the results of the benchmark tests.
(PTA) [15], which inserts constant pseudo capacitors and inductors to original circuits. However, PTA can cause oscillation issues. Damped PTA (DPTA) [16], exploits a numerical integration method with artificially enlarged damping effect to deal with oscillation; Ramping PTA (RPTA) [17] ramps up voltage sources instead of inserting the pseudo-inductors to suppress fill-ins. Compound element PTA (CEPTA) [18] has demonstrated a strong capability to eliminate oscillation while maintain a high efficiency. As to our knowledge, till now, there has been no literature showing effective (solver) parameter strategies for CEPTA acceleration.

To harness the power of modern machine learning and meanwhile retains 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 Bayesian optimization error-free acceleration framework using CEPTA (Fig. 1). Specifically, we introduce a Bayesian optimization (BO) to select PTA solver parameters as an optimization problem. To extend the capability 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 BetaCDF, which overcomes the stationary limitation of a general BO without complicating the geometry via a monotonic bijection transformation. Parameters of the warping BetaCDF are integrated out using variational inference to avoid overfitting.

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 SPICE solvers.
3) BoA-PTA requires no pre-computed data. It can accelerate ongoing repeated simulations or to improve new simulations of completely different circuits.
4) BoA-PTA is equipped with cutting-edge machine learning techniques: deep learning for netlist feature extractions, BetaCDF for non-stationary modelling, and variational inference to avoid overfitting.
5) BoA-PTA shows an average 2.2x (maximum 3.5x) speed-up on 43 benchmark circuit simulations and Monte-Carlo simulations.

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 solver. As a very first work of machine learning enhanced SPICE, we hope this work can inspire interesting machine learning enhanced EDA tool from different perspectives.

The rest of the paper is organized as follows. In Section 2, 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 for different tasks. We conclude this work in Section 5.

I. BACKGROUND AND PRELIMINARIES

A. SPICE Simulations Via PTA

When the commonly used NR and practical continuation methods fail to converge in a SPICE, the PTA is implemented as an alternative because it provides robust solutions to nonlinear algebraic equations from modified nodal analysis (MNA). PTA works by inserting certain dynamic pseudo-elements into the original circuits. As shown in Fig. 2 CEPTA inserts a GVL branch into an independent voltage source in serial (Fig. 2(a)), a RVC branch into an independent current source in parallel (Fig. 2(b)), and a transistors between each node to ground (Fig. 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 inductor $L$ connected in parallel with a time-variant conductance $G(t)$. With the pseudo-elements inserted, the differential algebraic equations (1) is solved with an initial guess $u_0$ until converge

$$
g(u(t), \dot{u}(t), t) = 0, \quad R(t) = R_0 e^{t/\tau}, \quad G(t) = G_0 e^{t/\tau}. \quad (1)$$

The converged solution (when $\dot{u}(t) = 0$) is the solution to the original circuit. The insertion of compound elements brings additional nodes (such as node $k$ in Fig. 2(b)) that require extra computations. To avoid enlarging the size of the Jacobian matrix induced by additional nodes, CEPTA can be implemented in an equivalent way, where the equivalent circuits are shown as Fig. 3}

![Fig. 1. The proposed BoA-PTA framework.](image1)

![Fig. 2. Inserted pseudo-elements ((a)GVL, (b,c)RVC) and their embedding positions in CEPTA.](image2)
branch RVC at time $t^{n+1}$ after discretization by the backward Euler is

$$I_{CB}^{n+1} = G_{CB} \cdot V_{CB}^{n+1} + I_{CBeq},$$  \hspace{1cm} (2)

where $G_{CB}^{-1} = h^{n+1}/C + R(t^{n+1})$ and $I_{CBeq} = G_{CB} \cdot (I^n_B \cdot R(t^n) - V_{CB}^n)$. Similarly, the equivalent equations for branch GVL at time point $t^{n+1}$ can be obtained by

$$V_{CB}^{n+1} = R_{CB} \cdot I_{CB}^{n+1} + V_{CBeq},$$  \hspace{1cm} (3)

where, $R_{CB}^{-1} = h^{n+1}/L + G^{n+1}$ and $V_{CBeq} = R_{CB} \cdot (-I_{CB}^n + G^n(V_{CB}^n - E)) + E$. Despite CEPTA’s great success, its performance is highly influenced by the inserted pseudo elements, i.e., the values of inserted pseudo capacitor, inductor, and the initial values of resistor and conductance. 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 makes solver parameter tuning an still open challenge.

B. Problem Formulation

Consider a CEPTA solver $g$ with solver parameters $x$ (indicating the value of inserted capacitor, inductor, resistor and conductance) that operates on a netlist file denoted as $\xi$ and generate the steady state $u = g(x, \xi)$. We are interested in reducing the number of iteration, denoted as $\eta(x, \xi) + \varepsilon$, for $g(x, \xi)$. Here $\varepsilon$ captures the model inadequacy and randomness that are not fully captured by $x$ and $\xi$. We aim to seek a function

$$x^*(\xi) = \argmin_{x \in X} \eta(x, \xi),$$  \hspace{1cm} (4)

where $x^*(\xi)$ is the optimal CEPTA solver parameters for any given netlist $\xi$ and $X$ is the feasible domain for $x$.

C. Bayesian Optimization

Bayesian optimization (BO) is an optimization framework generally for expensive black-box functions that are noisy or noise-free [19]. 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. Thus, we need as many data from the black-box function. Since the data are expensive to collect, we need to have a good strategy to update the surrogate model to gain maximum improvement with each new observation. This is known as the exploration. Keep in mind that the ultimate goal is the optimization of the black-box function rather than fitting a surrogate; this is known as the exploitation. The tradeoff between exploration and exploitation is handled by the acquisition function, which should reflect our reference for the tradeoff.

D. Gaussian Process

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 $\xi$ is thus omitted. Assume that we have observation $y_i = \eta(x_i) + \varepsilon$, $i = 1, \ldots, N$ and design points $x_i$, where $y$ is the (determined) iteration number needed for convergence. In a GP model we place a prior distribution over $\eta(x)$ indexed by $x$:

$$\eta(x)|\theta \sim \mathcal{GP} \left(m(x), k(x, x')|\theta\right),$$  \hspace{1cm} (5)

with mean and covariance functions:

$$m_0(x) = \mathbb{E}[\eta(x)],$$

$$k(x, x') = \mathbb{E}[\eta(x) - m_0(x)](\eta(x') - m_0(x'))],$$

where $m_0$, $\theta$ are estimated during the learning process. The mean function can be assumed to be an identical constant, $m_0(x) \equiv m_0$, by virtue of centering the data. Alternative choices are possible, e.g., a linear function of $x$, 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:

$$k(x, x'|\theta) = \theta_0 \exp(- (x - x')^T \text{diag}(\theta_1, \ldots, \theta_l)(x - x')).$$  \hspace{1cm} (6)

The hyperparameters $\theta = (\theta_0, \ldots, \theta_l)^T$, $\theta_1^{-1}, \ldots, \theta_l^{-1}$ in this case are called the square correlation lengths. For any fixed $x$, $\eta(x)$ is a random variable. A collection of values $\eta(x_i)$, $i = 1, \ldots, N$, on the other hand, is a partial realization of the GP. Realizations of the GP are deterministic functions of $x$.

The main property of GPs is that the joint distribution of $\eta(x_i)$, $i = 1, \ldots, N$, is multivariate Gaussian. Assuming the model inadequacy $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ is also a Gaussian, with the prior [5] and available data $y = (y_1, \ldots, y_n)^T$, we can derive the model likelihood

$$L \triangleq p(y|x, \theta) = \int_{\eta} p(y | \eta, m_0 1, K(\theta) + \sigma^2 I) d\eta = \mathcal{N}(y|m_0 1, K(\theta) + \sigma^2 I),$$

$$= - \frac{1}{2} (y - m_0)^T (K(\theta) + \sigma^2 I)^{-1} (y - m_0 1)$$

$$- \frac{1}{2} \ln[K(\theta) + \sigma^2 I] - \frac{N}{2} \log(2\pi),$$

where the covariance matrix $K(\theta) = [K_{ij}]$, in which $K_{ij} = k(x_i, x_j|\theta)$, $i, j = 1, \ldots, N$. The hyperparameters $\theta$ are normally obtained from point estimates [20] by maximum likelihood estimate (MLE) of (8) w.r.t. $\theta$. The joint distribution
uncertainty towards the maximum can approach the optimal by exploring the areas with higher mizations, e.g., L-BFGS-B.

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

E. Acquisition Function

For simplicity, let us consider the maximization of the black-box function without particular constraints. Based on the GP model posterior in (10), we can simply calculate the improvements for a new input $x$ as $I(x) = \max(\hat{y}(x) - y^*, 0)$, where $y^*$ is the current optimal and $\hat{y}(x)$ is the predictive posterior in (10). The expected improvement (EI) [23] over the probabilistic space is

$$EI(x) = \mathbb{E}_{\hat{y}(x) \sim \mathcal{N}(\mu(x|\theta),v(x|\theta))}[\max(\hat{y}(x) - y^*, 0)]$$

$$= (\mu(x) - y^*)\psi\left(\frac{\mu(x) - y^*}{v(x)}\right) + v(x)\phi\left(\frac{\mu(x) - y^*}{v(x)}\right),$$

(11)

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 (11) favors regions with larger uncertainty or regions with larger predictive mean values and naturally handles the tradeoff between exploitation and exploration. The candidates for next iteration is selected by

$$\arg\max_{x \in \mathcal{X}} EI(x),$$

(12)

which is normally optimized with classic non-convex optimizations, e.g., L-BFGS-B.

Rather than looking into the expected improvement, we can approach the optimal by exploring the areas with higher uncertainty towards to the maximum

$$\arg\max_{x \in \mathcal{X}} \left(\mu(x) + \beta^{\frac{1}{2}}v(x)\right),$$

(13)

where $\beta$ reflects our preference of the tradeoff of exploration and exploitation. This is known as the upper confidence bound (UCB) [23], which is simple and easy to implement yet powerful and effective. However, the choice for $\beta$ 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 [24] to take a further step to inquire at a location that produce maximum information gain (based on information theory) about the black-box function optimal,

$$MES(x) = -\mathbb{E}_{\hat{y}(x)}[h(y^*|\mathcal{D} \cup \{x, \hat{y}(x)\})] - H(y^*|\mathcal{D})$$

$$= -\mathbb{E}_{\hat{y}}[H(\hat{y}(x)|y^*)] + H(\hat{y}(x)),$$

(14)

where $y^*$ indicates the black-box function optimal, $\mathcal{D}$ means the current data set, and $H(\hat{y}) = -\int p(\hat{y}) \log(p(\hat{y})) d\hat{y}$ is the entropy for $p(\hat{y})$. The first term in (14) is generally achieved using sampling method whereas the second one has a closed-form solution. The readers are referred to [24] for more details.

BO is an active research area and there are many other acquisition functions, e.g., knowledge gradient [25] and predictive entropy search [26]. Ensembles of multiple acquisition function are also possible [27]. In this work, we focus on BO accelerated SPICE and we test it with EI, UCB, and MES. However, our method can be combined with any existing acquisition strategy.

II. PROPOSED BOE-PTA

A. Circuits Characterization Via Deep Learning

The most challenging part in this work is the characterization of the circuit where the SPICE solver is executed on. Recently, machine learning techniques have been implemented in theEDA community to accelerate the design/verification process [7]. Directly introducing a powerful model such as deep learning that directly uses netlist as inputs is feasible in some cases [28]. 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 [29] 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 $\xi$. A GP with commonly used kernel (e.g., (7)) is unlikely to be able to capture the complex correlations between different netlists. The DNN has shown to be a powerful automatic feature extraction for various practical applications [30]. Thus, we further introduce a deep learning transform as an automatic feature extraction for $\xi$, i.e., $\Phi(\xi)$, before the GP surrogate.

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

(15)

where $\Phi^{l}(\xi) = h(W^{l-1}\Phi^{l-1}(\xi) + b^{l-1})$, $\Phi^{1}(\xi) = \xi$, and $h(\cdot)$ 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 $\xi$ to be the output dimension of $\Phi(\xi)$. 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 learning complex correlation automatically through DNN. For this reason, this approach is also known as deep kernel learning [31].
B. Non-Stationary Gaussian Process 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 the choice of the kernel function. Consider the function $\eta(x, \xi)$ for a fixed $\xi$, according to our experiments, $\eta(x, \xi)$ is a highly nonlinear function w.r.t. $x$, making the commonly used stationary ARD kernel ineffective for modeling such a complex function.

Unlike the previous section where the complex correlations of $\xi$ can be captured automatically using a complex model such as DNN [29], latent space mapping [32], GP [33], modeling of $x$ requires extra cares 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. 2) Even worse, introducing another complex model can complicate the geometry, making the optimization of the non-convex acquisition function w.r.t. $x$ more difficult. Note that the DNN we implement in Section II-A does not suffer from the issue because it is not involved in the optimization of acquisition function. We will show the details in later sections.

For modelling $x$, we believe the rule of thumb is to follow the Occam’s razor and introduce a simple yet effective transformation for the solver parameter $x$. To this end, we follow the work of [34] and introduce a bijection Beta cumulative density function (BetaCDF),

$$w_d(x_d) = \int_{0}^{x_d} \frac{u^\alpha_d - 1(1-u)^\beta_d - 1}{B(\alpha_d, \beta_d)} du, \quad (16)$$

where $\alpha_d$ and $\beta_d$ 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

$$\log(\alpha_d) \sim N(\mu_d^a, \sigma_d^a), \quad \log(\beta_d) \sim N(\mu_d^b, \sigma_d^b), \quad (17)$$

for the BetaCDF. The introduced hyperparameters $\{\alpha_d, \beta_d\}^D_{d=1}$ can be obtained via point estimations. To avoid overfitting, [34] 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 much computational resource.

In this work, reparameterization trick [35] is utilized to conduct a fast posterior inference for $\{\alpha_d, \beta_d\}^D_{d=1}$, which is later integrated out. We use a log-Gaussian variational posterior $\log(\gamma) \sim N(\mu_\gamma, \Sigma_\gamma)$, where $\gamma = [\alpha_1, \ldots, \alpha_D, \beta_1, \ldots, \beta_D]^T$. This formulation allows us to capture the complex correlation between any $\alpha_d$ and $\beta_d$.

C. 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 cares. 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 small searching step.

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

$$x_d = (7 \cdot \text{sigmoid}(z_d))^{10}, \quad (18)$$

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 acquisition 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 $\eta(x, \xi)$ nor affect the non-stationary transformation [10], which is designed to tweak the space $X$ to resolve the non-stationary issue.

D. Boa-PTA Training and Updating

Given observation set $\{x_i, \xi_i, y_i\}^N_{i=1}$, the hyperparameters $\theta$, the DNN $\Phi$ parameters $\{W^i, b^i\}^L_{l=1}$, and the variational posterior parameters $\{\mu_\gamma, \Sigma_\gamma\}$ are updated using gradient descent. The joint model likelihood $L$ is the same as (8) but with a composite covariance kernel function

$$k((w(x), \Phi(\xi)), [w(x'), \Phi(\xi')]). (19)$$

The GP hyperparameters are updated by maximizing $L$ as in a general GP. The DNN $\Phi$ parameters $W^l$ and $b^l$ are updated by

$$\frac{\partial L}{\partial W^l} = \frac{\partial L}{\partial k} \frac{\partial k}{\partial W^l}, \quad \frac{\partial L}{\partial b^l} = \frac{\partial L}{\partial k} \frac{\partial k}{\partial b^l}. (20)$$

where $\frac{\partial}{\partial W^l}$ represents the gradient of the model log-likelihood w.r.t. the DNN. For the variational posterior $\gamma$, we use the parameterization trick to update the variational parameters $\mu_\gamma$ and $\Sigma_\gamma$. Specifically, we sample $\gamma_i$ for $i = 1, \ldots, S$ by

$$\gamma_i = \text{exp}(\mu_\gamma + \epsilon \cdot L), \quad (21)$$

where $\epsilon \sim \mathcal{N}(0, I)$ is sampled from i.i.d. standard normal distributions and $L^TT = \Sigma_\gamma$ is the Cholesky decomposition of $\Sigma_\gamma$. Given solver parameter $x$, the output of the BetaCDF warping becomes a distribution. We simplify this process by taking its expectation as the output, we have

$$w(x|\gamma) \approx \frac{1}{S} \sum_{i=1}^S w(x|\gamma_i). (22)$$

The variational parameters $\mu_\gamma$ and $\Sigma_\gamma$ can be now updated using back propagation. Taking $\mu_\gamma$ for instance, we have

$$\frac{\partial L}{\partial \mu_\gamma} = \frac{1}{S} \sum_{i=1}^S \frac{\partial L}{\partial k} \frac{\partial k}{\partial w} \frac{\partial w}{\partial \gamma_i} \frac{\partial \gamma_i}{\partial \mu_\gamma}. (23)$$
In practice, since the surrogate model is updated consequently, we set \( S = 1 \) to save computational resource as in [35]. Also, when updating the posterior, we also include the KL distance \( \text{KL}(q(\gamma)||p(\gamma)) \) in the likelihood function.

E. 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 \( x \) conditioned on a given netlist \( \xi \). This makes the optimization much easier and faster without repeated forward and backward propagation through the DNN. Specifically, conditioned on a netlist \( \xi \), the kernel \( k_1(w(x), w(x')) \cdot k_2(\Phi(\xi), \Phi(\xi')) \), where \( k_1 \) is an ARD kernel for \( w(x) \) 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 \( \xi \) is given.

F. BoA-PTA For Solver Parameter Optimization

Most surrogate model based acceleration techniques require pre-computed data for pseudo-random inputs [36]. 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” because the surrogate has no prior knowledge. For this situation, we use a batch iteration scheme to run BoA-PTA which is described in Algorithm 1.

**Algorithm 1 BoA-PTA Cold Start**

**Input:** Netlists \( \{\xi_1, \ldots, \xi_L\} = \Xi \), number of epoch \( N_{\text{epoch}} \)

1: Execute CEPTA with default setting on any netlist \( \xi_i \)
2: for \( j = 1 \) to \( N_{\text{epoch}} \) do
3: for \( i = 1 \) to \( M \) do
4: Update surrogate model \( \mathcal{M} \) by maximizing (8)
5: Update and optimize acquisition function (11), (13), or (14) given \( \xi_i \) and get candidate \( x \)
6: Execute CEPTA and collect iteration \( \eta(x; \xi_i) \)
7: end for
8: end for
9: return Best record of \( \{x^*, \eta(x^*, \xi_i)\} \) for \( i = 1, \ldots, M \); surrogate model \( \mathcal{M} \)

It might seem unnecessary to run Algorithm 1 because it requires repeated executions of the CEPTA solver and consumes extra computational resources. Indeed, this process provides little value for the task of solving netlists \( \Xi \) 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 II-G. 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 the future usage.

G. BoA-PTA For Monte-Carlo Acceleration

One of the most common situations for repeated SPICE simulation is the Monte-Carlo analysis, where a large number of modest variations of a given netlist is simulated. Denote \( Q \) as a netlist sampler which generates a netlist \( \xi \) 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 \( y = 9999 \) as the penalty for a non-convergence case, \( 2y^* \) as the threshold to halt a solver, 20 epoch as a convergence threshold. This hyperparameters 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 provide prior knowledges.

**Algorithm 2 BoA-PTA Monte-Carlo Acceleration**

**Input:** Netlists sampler \( Q \), number of samples \( N_{mc} \)

1: Sample a netlist \( \xi \) from \( Q \) and execute \( \eta(x; Q) \)
2: Update record of best \( x^* \) and best iteration \( y^* \)
3: for \( i = 1 \) to \( N_{mc} \) do
4: Update surrogate model \( \mathcal{M} \) by maximizing (8)
5: Sample a netlist \( \xi \) from \( Q \)
6: Optimize acquisition function and get optimal \( x \)
7: Execute \( \eta(x; \xi) \); if SPICE iteration reaches \( 2y^* \), stop the execution and set \( y = 9999 \); re-execute \( \eta(x^*, Q) \) and collect results.
8: Update record of best parameters \( x^* \) and iteration \( y^* \)
9: if \( y^* \) has no improvements over 20 iterations, stop BO and use constant \( x^* \) for the rest of simulations.
10: end for
11: return Monte-Carlo analysis for \( Q \)

H. Error Analysis And Computation Complexity

Unlike many verification/design acceleration solutions that are purely based on machine learning techniques [7] 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 \( (\hat{u} < 10^{-12}) \) by default. BoA-PTA thus an error-free approach. When BoA-PTA fails to improve CEPTA and lead to non-convergence situation, we are fully aware of such an error and can roll back to use the default setting. Once the GP is trained, it only takes \( O(N) \) and \( O(N^2) \) \( (N \) is number of observations) for the computation of \( \mu(x) \) and \( v(x) \), respectively. The complexity of the DNN (depending on the network structure) is approximately \( O(\sum_{i=1}^{L} M_i^2) \), where
$M_l$ is the number of unite 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 $(K + \sigma^2 I)^{-1}$, 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 \cite{37} can be implemented to resolve this
issue, which is outside the scope of this paper; we thus leave
it as a future work.

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

III. EXPERIMENTAL RESULTS

A. Benchmark Circuits And Experimental Setups

Most SPICE solvers have certain advantages on some particu-
lar circuits. To assess BoA-PTA thoughtfully, we test it on the
circuit simulator benchmark set known as CircuitSim93 \cite{38},
which contains 43 classic circuits including BJT-, MOS2-, and
MOS3-type circuits. We compare BoA-PTA to Ngspice, the
widely used open source SPICE based on Gmin stepping, and
the other PTA family SPICE including PPTA, DPTA, RPTA,
and CEPTA.

For BoA-PTA, we utilize a DNN of two hidden layers, each
of which contains 16 hidden units and a sigmoid activation
function. We use L-BFGS-B with five iterations for both the
GP fitting optimization and the acquisition optimization. We
evaluate BoA-PTA with three acquisition functions, i.e., EI,
UCB, and MES. For the UCB, we set the common $\beta = 0.1$.
The implementation of BoA-PTA is based on PyTorch and
BoTorch\footnote{https://pytorch.org; https://botorch.org}.

B. BoA-PTA Acceleration Efficiency

To show that BoA-PTA actually improves the SPICE effi-
ciency, we firstly compare BoA-PTA with a vanilla random
search method, which run each simulation with randomly
sampled solver parameters from a uniform distribution over $X$. We use Algorithm 1 to run BoA-PTA for all benchmark
simulations with 20 epochs. This experiment is repeated five
times with different random seeds to ensure robustness and
fairness of the results. After excluding the non-convergence
simulations, the average best records of speed-ups over the
43 benchmark simulations (and five repeat tests) are shown in
Fig. 4. We can see that BoA-PTA with MES clearly
outperforms BoA-PTA with other acquisition functions with
a large margin, particularly with only a few epochs. The
UCB converges to the same speed-up with slightly more
iterations. In contrast, the EI struggles to improve quickly.
Nevertheless, BoA-PTA with any acquisition is clearly better
than the random search scheme.

We show the detailed acceleration with 20 epochs for all
benchmark simulations in Table I. All DPTA and RPTA results
are worse than CEPTA and are not presented due to the
limited space. First thing we notice in Table I is that BoA-PTA
outperforms the best PTA method CEPTA in all cases except
for pump and reg0, in which the performance are equaled. This
clearly demonstrates that BoA-PTA improves CEPTA solver
without degeneration. Another interesting thing to point out
is that for the CEPTA non-convergence cases of \{opampal,
opus, ring, gm19\}, BoA-PTA makes them converge! This
is a particularly useful for PTA based SPICE as they sometime
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 non-
convergence data.

One may notice that Ngspice also demonstrates a good
performance as it outperforms BoA-PTA a few times. This is
not surprising because Ngspice utilizes the Gmin stepping,
which is particularly good for small-scale simple circuits but
scales poorly to large-scale real-world circuits due to non-
convergence issues \cite{39}. We can already see this in Table I as
there are eight non-convergence cases for the Gmin stepping
and only four cases for BoA-PTA-MES. In contrast, PTA
based methods are known to be robust to large-scale problems
but often suffer from slow convergence issues, leading to
a large number of iterations. We highlight that BoA-PTA
improves CEPTA so much that it can match Gmin in many
small-scale benchmark circuits, e.g., ring11 and ab-ac. For this
reason, we also highlight the best of our results in Table I
and evaluate their

C. Optimal Predictions For Unseen Simulations

In this experiment, we pick the circuit with large potential
for improvements among all types of circuits, i.e., $T$=
{bias, bjtff, bjtinv, gm2, gm6, jge, nand, schmitfast} and use them
as testing simulations for BoA-PTA. Specifically, simulations
that are not in $T$ 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 $T$ and evaluate their
speed-ups. We emphasis that the evaluations of $T$ are never
updated to BoA-PTA. They are strictly treated as testing data.
The results are shown in Fig. 5. 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 novelty of BoA-PTA.
As we can see in Fig. 5, BoA-PTA can further improve the SPICE speed-up with increasing number of epochs even the circuits have never been seen by the system. This 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 a stable improvement with only three epochs for all three types of simulations in this experiment. The MES, on the other hand, shows a slower improvement.

D. Monte-Carlo Accelerations

Lastly, 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 example. The 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., \(R = R_{\text{original}} \times N(1, \text{variation}^2)\). For each variation set, each method is tested on the same 1000 random sampled netlists to provide a fair comparison. Since BoA-PTA is error-free as discussed, we did not show the statistical results but focus on the run-time statistics. We firstly show the number of non-convergence (#NC), the mean, and the standard deviation (STD) iterations for 6000 total simulations in Table II. We can see clearly that BoA-PTA always converges and always provides minimal iterations. Among different acquisition functions, BoA-PTA with MES consistently show the best performance, with is consistent with the observation in previous experiments. CEPTA always has the lowest standard deviation of iterations. We argue that what matters most is the total iterations not the deviation for the run-time. Also, 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 6000 simulations) are shown in Fig. 6 without DPTA, RPTA, and PPTA due to their high non-convergence rate. Compared to CEPTA, we can see that whichever acquisition function improves BoA-PTA for a large margin. BoA-PTA with MES overall obtains the most stable and good performance with approximately 2x speed-up over CEPTA.

IV. CONCLUSION

In this paper, a Bayesian optimization SPICE acceleration is proposed and it is demonstrated using BoA-PTA, a combination with PTA solver. By harnessing the advantages of modern machine learning techniques, BoA-PTA demonstrate a substantial improvement (up to 3.5x speed-up) over the original CEPTA solver with little extra cost. The improvements are validated through 43 benchmark circuit simulations and Monte-Carlo simulations.
### TABLE I
Solver iterations for benchmark circuits

| Circuit | Ngspice | PPTA | CEPTA | MES | EI | UCB |
|---------|---------|------|-------|-----|----|-----|
| astabl  | 3359    | 108  | 55    | 46  | 50 | 46  |
| bias    | 86      | N/A  | 839   | 239 | 294| 355 |
| bjtinv  | 116     | N/A  | 169   | 102 | 90 | 86  |
| latch   | 46      | 148  | 130   | 86  | 84 | 83  |
| loc     | N/A     | N/A  | N/A   | N/A | N/A| N/A |
| nagle   | 117     | 2440 | 306   | 306 | 306| 306 |
| opamp   | 168     | 2335 | N/A   | 794 | 866| 635 |
| optrans | N/A     | N/A  | 2206  | 1561| 2118|2283|
| rca     | 47      | 76   | 82    | 55  | 57 | 64  |
| ring11  | 41      | 102  | 63    | 63  | 63 | 51  |
| schmitcl| N/A     | 48   | 52    | 45  | 47 | 44  |
| vreg    | 37      | N/A  | 22    | 22  | 22 | 22  |

| Circuit | Ngspice | PPTA | CEPTA | MES | EI | UCB |
|---------|---------|------|-------|-----|----|-----|
| ab_ac   | 53      | N/A  | 90    | 79  | 79 | 82  |
| ab_integ| 64      | N/A  | 499   | 460 | 454| 471 |
| ab_opamp| 583     | N/A  | 150   | 121 | 126| 124 |
| cram    | 40      | N/A  | 91    | 90  | 91 | 87  |
| e1480   | 975     | 3213 | 179   | 165 | 134| 118 |
| g1310   | 1254    | N/A  | 76    | 48  | 48 | 51  |
| gm6     | N/A     | N/A  | 63    | 42  | 43 | 45  |
| hussamp | 46      | N/A  | 91    | 88  | 85 | 82  |
| mosrect | 44      | 251  | 65    | 54  | 51 | 53  |
| mux8    | 25      | 8579 | 122   | 90  | 93 | 91  |
| nand    | 25      | N/A  | 83    | 55  | 56 | 54  |
| pump    | 47      | N/A  | 22    | 22  | 22 | 22  |
| reg0    | 52      | 22   | 22    | 22  | 22 | 22  |
| ring    | 63      | N/A  | 70    | N/A | 1126|N/A |
| schmitfast | 46 | 71 | 82 | 69 | 67 | 64 |
| schimitslow | N/A | N/A | 127  | 96  | 93 | 108 |
| slowlatch | 58     | N/A  | 169   | 108 | 163| 135 |
| toronto | 38      | N/A  | 277   | 258 | 277| 273 |

### TABLE II
Monte-Carlo simulation statistics

| Circuit | DPTA | RPTA | PPTA | CEPTA | MES | EI | UCB |
|---------|------|------|------|-------|-----|----|-----|
| bias    | #NC  | N/A  | N/A  | N/A   | N/A | N/A| N/A |
| bjtinv  | 0    | 0    | 0    | 0     | 0   | 0  | 0   |
| bjtff   | 6000 | 6000 | 6000 | 0     | 0   | 0  | 0   |

| Circuit | Mean | STD  | Mean | STD  | Mean | STD  | Mean | STD  |
|---------|------|------|------|------|------|------|------|------|
| bias    | 3913 | 88   | 5710 | 9    | 0    | 0   | 0    | 0    |
| bjtinv  | 73.9 | 73.8 | 124.5| 55.3 | 49.5 | 53.8| 48.8 |
| bjtff   | 14.4 | 14.8 | 10.4 | 3.3  | 12.3 | 13.9| 10.3 |

| Circuit | Mean | STD  | Mean | STD  | Mean | STD  |
|---------|------|------|------|------|------|------|
| bias    | 9993 | 722  | 5644 | 913  | 346  | 359  | 484  |
| bjtinv  | 14.4 | 14.8 | 10.4 | 3.3  | 12.3 | 13.9| 10.3 |
| bjtff   | 6000 | 6000 | 6000 | 0    | 0    | 0    | 0    |
