Local Bayesian Optimization For Analog Circuit Sizing

Konstantinos Touloupas, Nikos Chouridis and Paul P. Sotiriadis
National Technical University of Athens, Greece
E-mail: ktouloupas@mail.ntua.gr

2021 58th ACM/IEEE Design Automation Conference (DAC) | 978-1-6654-3274-0 /21/$31.00 ©2021 EU | DOI: 10.1109/DAC18074.2021.9586172

***Abstract*—This paper proposes a Bayesian Optimization (BO) algorithm to handle large-scale analog circuit sizing. The proposed approach uses a number of separate Gaussian Process (GP) models approximating the objective and constraint functions locally in the search space. Unlike mainstream BO approaches, it is able to traverse high dimensional problems with ease and provide multiple query points for parallel evaluation. To extend the method to large sample budgets, GP regression and sampling are enhanced by using kernel approximations and GPU acceleration. Experimental results demonstrate that the proposed method finds better solutions within given budgets of total evaluations compared to state-of-the-art approaches.**

I. Introduction

While the demand for modern electronic systems with advanced functionalities increases, reducing the design time is becoming important. Despite the progress made in digital Integrated Circuits (IC) design, which is facilitated by automatic synthesis tools, analog design automation has not yet fully developed. This problem is exacerbated by the continuous scaling of transistor dimensions, since manual analog IC design is becoming increasingly difficult. Therefore, new and robust automation methods are needed to accelerate the design of modern electronic systems.

Most approaches to automated analog design aim to discover optimal device sizes, given a fixed circuit topology [1]. This can be formulated as a constrained optimization problem, where user-defined optimization goals and constraints drive the selection of device sizing. There are two main approaches to this optimization problem, namely equation based and simulation based approaches. The equation based approach requires the designer's intervention, either by defining explicit equations of circuit performances, or by using regression methods to acquire them [1]. Analytic expressions however, cannot capture the higher order effects stemming from small channel dimensions and lead to inaccurate sizing results. Simulation based approaches, on the other hand, require excessive computational resources, but allow for higher accuracy. In this work, we focus on simulation-based analog sizing.

Several methods have addressed the simulation based analog sizing, including Evolutionary Algorithms (EAs) and Simulated Annealing (SA) [2]. However, their slow convergence rate makes them unsuitable for sizing large scale analog and RF circuits. Bayesian Optimization (BO) [3], a method that has attracted attention in the machine learning community, promises to alleviate this limitation. BO incrementally constructs a Gaussian Process (GP) regression model which is used to select the next evaluation (query) point. This algorithm and some of its variants have been applied to analog and RF circuit sizing [2], [4]–[6], demonstrating promising results.

While BO has emerged as a competitive method for optimizing black box functions, scalability to large sample budgets and high-dimensional search spaces remains a major concern. For n training samples, $\mathcal{O}(n^3)$ time is required to compute GP predictive means and variances [7], which are used by the BO to determine each query point. In addition, GP regression, relying on the Euclidean distance to define sample correlations, becomes inefficient in high dimensions, which is a problem known as the curse of dimensionality. When constraints apply, such as in the case of analog IC sizing, BO should be able to identify feasible query points, which is a non-trivial task by itself.

This work intends to address the automatic sizing of analog and RF building blocks, using a new variant of BO that scales well in high-dimensional and constrained problems. To address the aforementioned shortcomings of classic BO, we use a local-based scheme that maintains a number of separate GP models, each one capturing the objective and constraint functions inside a sub-region of the whole design space. To take full advantage of multi-core workstations, the proposed BO variant provides a batch of query points at each iteration, using a modified Thomson Sampling acquisition function. GP regression is scaled to large datasets using sparse approximations for the GP covariance matrix and Blackbox Matrix-Matrix Gaussian Process inference [8], which allows for GPU acceleration. Compared to BO implementations for constrained optimization and state-of-the-art EAs, the proposed method provides better results and a speedup of $\times 40$.

The rest of this paper is organized as follows. Section II provides an overview of the analog circuit sizing problem and the classic BO method. Section III demonstrates the proposed approach. Section IV provides experimental results on two real-world circuits and Section IV concludes the paper.

II. Background

A. Problem Formulation

Analog and RF circuit sizing can be cast to a constrained optimization problem:

$$\begin{aligned} \min \quad & f(\mathbf{x}), \quad \mathbf{x} = [x_1, x_2, \ldots, x_d] \\ \text{s.t.} \quad & g_j(\mathbf{x}) \leq 0, \quad j = 1, \ldots, l \\ & L_i \leq x_i \leq U_i, \quad i = 1, \ldots, d \end{aligned} \tag{1}$$

where vector $\mathbf{x}$ contains the design variables, L_i and U_i are the lower and upper bounds of the i-th variable, $S = \prod_{i=1}^{d}[L_i, U_i]$ is the variable space, f is the objective (fitness) function and g_j is the j-th constraint. For a given parameter vector $\mathbf{x}$, its degree of constraint violation is defined as

$$CV(\mathbf{x}) = \sum_j max[0, g_j(\mathbf{x})].$$

In the context of simulation-based sizing, parametrized testbenches are simulated in iterations using a commercial simulator and software platforms that automate the procedure of data processing and simulation automation.

B. Gaussian Processes

Assume a dataset $\mathcal{D}$ comprised of n d-dimensional parameter vectors $X = \{\mathbf{x}_i\}_{i=1}^{n}$ and their corresponding 1-dimensional observation value set $\mathbf{y} = \{y_i\}_{i=1}^{n}$, generated by an unknown function, f and an uncorrelated additive noise $\epsilon_i \sim \mathcal{N}(0, \sigma_n^2)$, as $y_i = f(\mathbf{x}_i) + \epsilon_i$. A Gaussian Process (GP) is a stochastic process of infinitely many random variables, any finite set of which jointly follows a Gaussian distribution. A GP can be used to provide a probability distribution over functions that approximate f, such that any collection of function values $f(X)$ are random variables that have a joint Gaussian distribution

$$f(X) = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)]^{\mathrm{T}} \sim \mathcal{N}(\boldsymbol{\mu}, K). \tag{2}$$

Here, vector $\boldsymbol{\mu}$ is the GP mean, defined by a mean function $m(\mathbf{x})$, and K is the covariance matrix, constructed by a kernel function $k(\mathbf{x}, \mathbf{x}')$, such that $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) + \sigma_n^2 \delta_{ij}$, where δ_{ij} is the Kronecker delta. In cases when no prior information about f is available, the mean function is set to a constant μ. Therefore, the expressive capabilities of the GP model are determined by the kernel function. Popular kernel functions include the RBF and Matèrn kernel families [7]. In this work, we use the Matèrn 5/2 kernel with automatic relevance determination,

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2 \left(1 + \sqrt{5}r + \frac{5}{3}r^2\right) e^{-\sqrt{5}r} \tag{3}$$

where

$$r = \left(\sum_{k=1}^{d} \frac{(x_{i,k} - x_{j,k})^2}{\lambda_k^2}\right)^{1/2}. \tag{4}$$

Parameters σ, the lengthscales λ_k, the constant mean μ and the noise variance σ_n are called the hyperparemeters of the GP model. For the following, we adopt the common approach [7] and fix $\mu = 0$.

To predict f at a point $\mathbf{x}^\star \notin \mathcal{D}$, one uses the predictive distribution $p(f(\mathbf{x}^\star)|X, \mathbf{y})$. This is a Gaussian distribution with mean and variance

$$\begin{aligned} \mu_{f|D}(\mathbf{x}^\star) &= \mathbf{k}^{\mathrm{T}} K^{-1} \mathbf{y} \\ \sigma^2_{f|D}(\mathbf{x}^\star) &= c - \mathbf{k}^{\mathrm{T}} K^{-1} \mathbf{k}. \end{aligned} \tag{5}$$

Here, $\mathbf{k}^{\mathrm{T}}$ is a $(1 \times n)$ vector with values $k(\mathbf{x}_i, \mathbf{x}^\star)$ for $i = 1, \ldots, n$ and $c = k(\mathbf{x}^\star, \mathbf{x}^\star)$. The predictive distribution provides estimates not only for point-wise query points, but for multiple inputs $X^\star$ as well. In the case of m query points, equations in (5) provide the mean and the diagonal elements of the covariance matrix of a multivariate Gaussian distribution $p([f(\mathbf{x}_1^\star), \ldots, f(\mathbf{x}_m^\star)]\,|X, \mathbf{y})$. The covariance between two points $\mathbf{x}^\star, \mathbf{x}^{\star\prime}$ is given by

$$\mathrm{Cov}(\mathbf{x}^\star, \mathbf{x}^{\star\prime}) = k(\mathbf{x}^\star, \mathbf{x}^{\star\prime}) - \mathbf{k}^{\mathrm{T}}_{X,\mathbf{x}^\star} K^{-1} \mathbf{k}_{X,\mathbf{x}^{\star\prime}}, \tag{6}$$

where $\mathbf{k}^{\mathrm{T}}_{X,\mathbf{x}^\star} = k(\mathbf{x}_i, \mathbf{x}^\star)$ for $i = 1, \ldots, n$. Sampling from this joint distribution allows for sampling functions from the GP model [7].

To adapt a GP model to $\mathcal{D}$, its hyperparameters must be learned from the data. This is done by minimizing the negative log marginal likelihood

$$L(\theta) = \frac{1}{2}\mathbf{y}^T K^{-1} \mathbf{y} + \frac{1}{2}\log(|K|) + \frac{n}{2}\log(2\pi). \tag{7}$$

This expression is used to learn parameters θ, using gradient based optimization. The derivatives with respect to each θ_i are given by

$$\frac{dL}{d\theta_i} = \frac{1}{2}\mathrm{Tr}\left(K^{-1}\frac{dK}{d\theta_i}\right) - \frac{1}{2}\mathbf{y}^T K^{-1} \frac{dK}{d\theta_i} K^{-1} \mathbf{y}. \tag{8}$$

C. Bayesian Optimization

Bayesian Optimization (BO) [3] is a sample efficient method to solve global optimization problems, particularly aiming expensive-to-evaluate cost functions. In the unconstrained regime, a real valued, unknown function f is provided, and BO learns a fast to evaluate surrogate model from past evaluations. It selects next query points for evaluation sequentially, by balancing exploration and exploitation to find the global optimum.

The BO framework consists of two main components; the probabilistic surrogate model that aims to approximate function f and an *acquisition function* that provides a score of utility for evaluating a candidate query point, based on the probabilistic model. In most approaches, the surrogate model is a GP, which is trained and used for predictive point evaluations as discussed in the previous subsection. Popular acquisition functions include the expected improvement (EI), probability of improvement (PI), entropy search (ES) and Thompson sampling (TS) [3].

Starting from an initial set of evaluations, BO incrementally builds a GP model based on historic data, and selects a new query point, as the point $\mathbf{x}$ that optimizes the acquisition function. This is an auxiliary optimization problem, but since the acquisition function is fast to evaluate, off-the-shelf optimization methods such as CMA-ES [3] can be used.

In the constrained optimization case, such as (1), the most prominent approach is to construct l additional GP models, each approximating a single constraint function. The query points are selected in a similar manner, but the acquisition functions are different. In [2], a weighted expected improvement acquisition function was used, which is an EI function weighted by the probability of feasibility for each query point. This is computed using the additional GP models.

III. Proposed Approach

In this section, the proposed approach for local BO based analog circuit sizing is presented, along with implementation details for computational efficiency.

A. Local Bayesian Optimization

Local based approaches for global optimization are extensively studied in the context of EAs. A surrogate within a restricted (trust) region is trained and used to suggest query points. A similar approach is proposed in [9], where a sequential model building optimization algorithm uses GP models inside a trust region to model the objective function, and defines an acquisition function to select future query points. This BO variant alleviates the problem of heterogeneity of objective functions, since query points are selected based on only the local dynamics of the problem at hand.

To achieve global optimization using local-based GP models, however, multiple trust regions are employed in parallel. In the case of k trust regions, $k \times (1+l)$ GP models (objective and constraints) must be maintained and trained throughout the procedure. Starting from an initial sampling of the design space using Latin Hypercube Sampling, each trust region is assigned a number of observations to build and initial set of GP models. While they are allowed to overlap, they maintain separate historic data archives used for training their respective GP models.

Each trust region is a hyper-rectangle and its center is chosen to be the maximum utility point in their respective historic data archive, meaning either minimum objective function value or constraint violation, if no feasible solution is yet to be found. The centre, as well as the length of each trust region are updated in each iteration. In particular, the length of each hyper-rectangle is denoted by $\mathcal{L}$ and is updated according to the trust region progress. To quantify this progress, we borrow an approach from EAs [10]. Given p_i as the current centre of the i-th hyper-rectangle and $q_{i,\text{best}}$ as the best query point probed in it in a particular iteration, quantities

$$I_{opt} = \frac{f(p_i) - f(q_{i,\text{best}})}{\hat{f}(p_i) - \hat{f}(q_{i,\text{best}})}, I_{\text{inf}} = \frac{CV(p_i) - CV(q_{i,\text{best}})}{\hat{CV}(p_i) - \hat{CV}(q_{i,\text{best}})} \quad (9)$$

define the optimization progress, where $\hat{}$ denotes values predicted using GP models and CV is the constraint violation function from Eq. (1). Parameters, c_1, c_2 and the boundary values for $\mathcal{L}$, $\mathcal{L}_{max}$ and $\mathcal{L}_{min}$ control its adaptation, with $0 < c_1 < 1 < c_2$. The indicator for the i-th trust region is given by

$$\mathcal{I}_i = \begin{cases} I_{opt} & \text{for } p_i, q_{i,\text{best}} \text{ feasible} \\ I_{\text{inf}} & \text{for } p_i, q_{i,\text{best}} \text{ infeasible} \\ c_1 & \text{for } p_i \text{ feasible}, q_{i,\text{best}} \text{ infeasible} \\ c_2 & \text{for } p_i \text{ infeasible}, q_{i,\text{best}} \text{ feasible} \end{cases}. \quad (10)$$

At the end of each iteration, $\mathcal{L}_i$ is updated by following rule:

$$\mathcal{L}_i = \begin{cases} \max(c_1 \times \mathcal{L}_i, \mathcal{L}_{min}) & \text{for } \mathcal{I}_i \leq c_1 \\ \min(c_2 \times \mathcal{L}_i, \mathcal{L}_{max}) & \text{for } \mathcal{I}_i \geq c_2 \\ \mathcal{L}_i & \text{otherwise} \end{cases}.$$

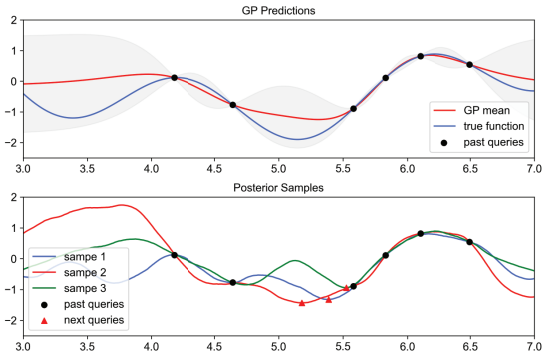

Fig. 1. Top: A GP model's pointwise predictions on a test function (mean and 95% confidence bounds shown). Bottom: 3 functions sampled from the GP model, along with the selected query points for next evaluation.

The algorithm operates in a transformed variable space, where the upper and lower bounds for each variable lie within the unit cube, therefore it holds $\mathcal{L}_{max} < 2$ and $\mathcal{L}_{min} > 0$.

To select the next query points for evaluation, a modified Thomson Sampling [3] acquisition function is employed. Consider the case when a single trust region is used. Thomson sampling uses a Sobol sequence [9] to select r candidate query points $\{\mathbf{x}_i\}_{i=1}^{r}$ residing in the trust region. For each one of the $l+1$ (objective and constraints) GP models employed, a sample is taken from their respective joint posterior distributions. This results in vectors $\left[\hat{f}(\mathbf{x}_i), \hat{g}_1(\mathbf{x}_i), \ldots, \hat{g}_l(\mathbf{x}_i))\right]$ used to compute $\left[\hat{f}(\mathbf{x}_i), \hat{CV}(\mathbf{x}_i)\right]$ for every $\mathbf{x}_i$ produced by the Sobol sequence. The $\mathbf{x}_i$ of maximum utility is chosen using the feasibility rule [11], which compares all candidates in pairs and selects the best as follows:

- Feasible candidate solutions are preferred than infeasible ones
- amongst feasible solutions, the ones with better fitness function are preferred and,
- amongst infeasible solutions, the ones with the least constraint violation are preferred.

In the case of more than one trust regions, the sampling procedure is repeated for each one of them, and the maximum utility point is selected from the pool of all candidate query points and all trust regions.

The above procedure extends naturally to batched query point selection; from the joint GP posterior on all candidate points, one can have multiple samples. Sequentially, the aforementioned selection scheme picks the maximum utility point, corresponding to a single posterior sample, and makes sure not to pick the same candidate point again. A demonstration (no constraints apply) is given in Fig. 1, where 3 samples are drawn from the GP posterior, and the next batch of query points are given as the minimum of each sample.

B. Sparse Kernel Approximation

In practice, GP training and prediction become intractable for large data sets [12]. This has motivated the research for approximate GP inference. Perhaps the most widespread

approach includes the *inducing point* methods [6], [7], [12], which employ a set of $m \ll n$ inputs $Z = \{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$ to form a Nystrom approximation of the covariance matrix,

$$K \approx K_{Nystrom} = K_{xz} K_{zz}^{-1} K_{zx}, \tag{11}$$

where $K_{xz} = K_{zx}^{\mathrm{T}}$ is a $(n \times m)$ covariance matrix between training and inducing points, evaluated using the exact kernel k. By adding a diagonal correction term,

$$K \approx K_{Nystrom} + \Lambda,$$

where $\Lambda_{(i,i)} = k(\mathbf{x}_i, \mathbf{x}_i) - k_{zx_i} K_{zz}^{-1} k_{zx_i}$, the approximate covariance matrix is ensured to be full rank [13]. Using this approximation, GP inference time complexity is reduced to $\mathcal{O}(nm^2)$. A comprehensive review of inducing point methods is provided in [12].

The selection of inducing points determines the predictive performance of the approximate GP. A naive approach would be to select a subset of the training samples to construct the covariance matrix either in random, or by means of an expensive combinatorial optimization [7]. A more elegant approach regards the inducing point locations as additional $(m \times d)$ GP hyperparameters, and provides solutions during the gradient based likelihood optimization of Eq. (7). We use the latter approach and modify it to fit the local-based BO scheme better.

In the context of local-based BO, trust region sizes and locations alter during the optimization, resulting in GP models trained with archived samples that do not reside in the current trust region. The reader is reminded that the acquisition function is restricted to select query points strictly within the trust regions. Thus, one should select inducing point locations that provide higher predictive accuracies within the trust regions. It has been noted that GPs with sparse kernels provide good accuracy in regions where the inducing points are densely concentrated [6]. We therefore restrict the inducing points to lie within each trust region.

Using the Adam [8] optimizer for gradient based optimization, we empirically found that the optimized locations of inducing points depend on two factors; the initial locations used and the optimizer's learning rate. We therefore impose the starting locations to be within or in proximity of the trust regions, and select their learning rate to be an order of magnitude less than that of the kernel function hyperparameters. Algorithm 1 explains the initial location selection for the inducing points.

C. Scalable Gaussian Process Regression

While using approximate kernels provide a remedy to the sample budget bottleneck of GPs, further performance gains can be achieved in terms of matrix computations. Inference and training for GPs require the evaluating terms $K^{-1}\mathbf{y}$, $\log|K|$ and $\mathrm{Tr}\left(K^{-1}\frac{dK}{d\theta_i}\right)$ in expressions (5), (7) and (8). The straightforward approach for this is Cholesky decomposition, which scales cubically with sample size [7]. The Black-box matrix multiplication method [8] is a recently proposed alternative that reduces the asymptotic complexity of GP inference to $\mathcal{O}(n^2)$. This is a modified conjugate gradient algorithm that allows for GPU acceleration. The software package `GPytorch` implements this method and it is used in our implementation. Moreover, to scale GP sampling, which is required in the acquisition function of the proposed BO, we use the LanczOs Variance Estimates (LOVE) [14] method.

Algorithm 1: Inducing Point Initial Location Selection

Input: $\mathcal{L}_i$ (trust region length), X (trust region archive), p_i (trust region center), m (inducing point count)
Output: loc (inducing point locations)
$A \leftarrow X \in$ Hyper-rectangle$(\mathcal{L}_i, p_i)$ `// Subset of training vectors lying inside the TR hyper-rectangle;`
if *len(A)* $> m$ **then**
  $loc \leftarrow$ KMeans(A, m clusters) `// m centroids;`
else
  $loc \leftarrow arg\min_{i=1,\ldots,m} ||p_i - X||$ `// m training vectors closest to` p_i;
end
Return loc

TABLE I
GP TRAINING AND SAMPLING RUNTIMES FOR ROSENBROCK FUNCTION

Operation	Exact GP		SGP (m=200)		SGP (m=100)	
	CPU	GPU	CPU	GPU	CPU	GPU
Training (s)	18.54	5.26	10.53	5.08	7.9	4.79
Sampling (s)	27.29	0.33	23.06	0.257	17.07	0.23
Sampling (LOVE)(s)	1.59	0.17	0.41	0.153	0.34	0.158

The gains of the aforementioned practices are demonstrated empirically using a toy experiment; one exact and 2 sparse GP models approximate a 20D Rosenbrock function. For a training dataset of 3000 samples, and 5000 candidate points to jointly sample from, the times required for training and sampling are given in Table I. For all GP models, 100 gradient descent iterations were used. Both LOVE and GPU acceleration provide a combined speedup of $\times 8$ for exact GPs and $\times 6$ for approximate ones.

IV. EXPERIMENTAL RESULTS

To test the performance of the proposed approach on large-scale problems, we perform experiments on two circuits, each one having more than 20 parameters. For comparison, we use the popular Differential Evolution (DE) algorithm with the feasibility rule for constrained optimization [15], and the BO variant WEIBO [2] proposed for automated analog design. For each experiment, a set of trial and error attempts were made to find the most suitable hyperparameters for DE. All algorithms are implemented in Python, and all GP regression models were set using `GPytorch`, using the same set of kernel and mean functions. For the proposed method, the number of inducing points is $m = 150$. The optimization runs are repeated 5 times to account for random effects. An in-house tool was used to automate simulations and result processing from Cadence Spectre. All circuits were implemented using a TSMC 90nm PDK, on a 8 core machine with a Quadro P5000 GPU.

TABLE II
THREE STAGE AMPLIFIER SPECIFICATIONS

Performance	Description	Specification
Load C_L (nF)	Capacitive Load	15
PM (^{o})	Phase Margin	≥ 52.3
DC Gain (dB)	Voltage Gain	≥ 100
GM (dB)	Gain Margin	≥ 18.1
Average SR ($V/\mu s$)	Average Slew Rate	≥ 0.22
UGF (MHz)	Unity Gain Frequency	≥ 0.95
P_{dc} (μW)	Total Power @ 2V V_{dd}	minimize

TABLE III
OPTIMIZATION RESULTS FOR THE THREE STAGE AMPLIFIER

Method	Best Solution	Time	Success
DE	$131.63 \pm 3.94\ \mu W$	27.7 ± 0.2min	3/5
WEIBO	$114.98 \pm 0.96\ \mu W$	350.3 ± 1.94min	5/5
TR-1	$112.42 \pm 0.7\ \mu W$	24.33 ± 0.31 min	5/5
TR-3	$112.50 \pm 0.95\ \mu W$	27.25 ± 0.24 min	5/5
TR-1 (GPU)	$111.76 \pm 1.19\ \mu W$	8.71 ± 0.39 min	5/5
TR-3 (GPU)	$110.54 \pm 1.33\ \mu W$	9.43 ± 0.41 min	5/5

A. Three-Stage Amplifier

A Three-Stage Amplifier shown in Fig. 2 [16] is sized in this example. This circuit was originally implemented in a .35um process, offering good driveability for large capacitive loads.

For this circuit, the search variables include transistor widths, the lengths for the devices in each stage of the amplifier and for the biasing transistors, resistors R_1, R_2, R_z, capacitors C_m and C_z and two biasing currents, based on the original sizing flow [16]. Note that symmetry restrictions are taken into account, resulting in a 23-dimensional optimization problem. We use a 2V power supply and a 15nF capacitive load.

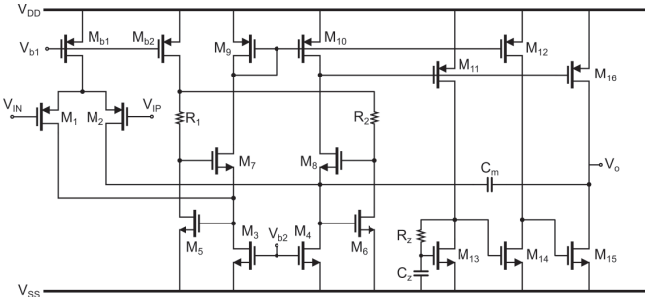

Fig. 2. Three stage amplifier proposed in [16].

To make fair comparisons, each algorithm is restricted to search within the same variable space, which consists of the minimum and maximum ranges for device sizes imposed by the PDK, while resistors are restricted to be less than 200kΩ, capacitors less than 2pF and biasing currents less than 15uA. For specifications, we use the ones provided in the original implementation in the case of 15nF load capacitor, which are shown in Table II. The sizing goal is to minimize the total power dissipation, P_{dc}. DE population and generations are 100, the maximum number of evaluations for the other two approaches is 1500 and the initial sampling size is 100.

To examine the effect of the trust-region count, two separate experiments were executed with a single and three trust regions, all of which have batch size of 15. The sizing results are shown in Table III, where best solution refers to lowest P_{dc} acquired with constraints met (averages and standard deviations shown). The proposed method consistently outperforms WEIBO and DE in terms of acquired solutions, with DE finding feasible solutions only 3 out of 5 times. Using 3 trust regions provides slightly better results, with an overhead on runtime, as training and sampling may occur for multiple GP models in certain iterations.

Special notice must be given on the runtimes; circuit evaluations are done in parallel, using 8 instances of the simulator each one running batched simulations. For this example, which includes two separate testbenches (small signal analysis, slew rate measurement), the averaged runtimes over 5 evaluations for a batch of 100, 15 and a single simulation are 16.3sec, 3.8sec and 1.3sec respectively. Therefore, batched parallel evaluation, provides significant runtime gains compared to sequential evaluations, which is the main reason for WEIBO to be so slow. Besides this, GPU acceleration provides an overall speedup of $\times 2.8$ for the proposed method (over total runtime), and achieves a $\times 40$ overall speedup compared to WEIBO.

B. High Linearity LNA

An LNA shown in Fig. 3 [17] is sized in this example. This circuit uses a complementary transconductance input stage and a set of axillary devices to achieve good linearity. Power supply is set to 1.2V. Following an initial manual design, the specifications for this circuit are based on the original implementation, with the exception of the working frequency, which is 2.4GHz instead of 1GHz, and they are given in Table IV. Maximum IIP3 is the optimization goal.

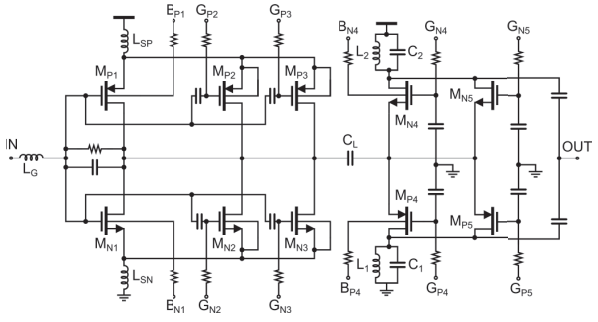

Fig. 3. High Linearity LNA proposed in [17].

Design variables include device widths, capacitances, inductances, resistances and bias voltages. Two separate variables correspond to pmos and nmos device lengths. This amounts to a total of 33 design variables. Their ranges are the maximum and minimum allowed by the PDK for device geometries, while the others are set by an initial rough design.

The batch size of the proposed approach is 15 and its simulation budget is 2000, same with WEIBO. Both methods

TABLE IV
HIGH LINEARITY LNA SPECIFICATIONS

Performance	Description	Specification
S_{11} (dB)	Input Matching @ 2.4GHz	≤ -10
S_{22} (dB)	Output Return Loss @ 2.4GHz	≤ -10
Gain (dB)	Voltage Gain @ 2.4GHz	≥ 15
NF (dB)	Noise Figure	≤ 1
P_{dc} (mW)	Power dissipation	≤ 50
IIP3 (dBm)	Third-order intercept point	maximize

TABLE V
OPTIMIZATION RESULTS FOR THE HIGH LINEARITY LNA

Method	Best Solution	Time	Success
DE	3.12 dBm	58min	1/5
WEIBO	10.87 ± 1.63 dBm	623 ± 4.2min	5/5
TR-1	19.81 ± 0.97 dBm	37 ± 1.3 min	5/5
TR-3	21.41 ± 0.44 dBm	41 ± 1.9 min	5/5
TR-1(GPU)	19.06 ± 0.69 dBm	14.78 ± 0.9 min	5/5
TR-3(GPU)	22.34 ± 0.553 dBm	15.97 ± 1.1 min	5/5

sample initially a batch of 100 candidate vectors. DE population and generations are 100 and 200. The experimental results are shown in Table V. In this demanding problem, the proposed approach clearly outperforms the other methods in terms of feasible IIP3 outcomes. While WEIBO is able to find feasible solutions all times, it provides close to 10dBm less IIP3 compared to the proposed approach, within the same sample budget. This highlights the performance advantage of the proposed method in high-dimensional problems, especially against DE which finds feasible solutions only once.

In terms of runtime, execution and results processing for a batch of 100, 15 and a single simulation take 17.2, 4.9 and 1.4 seconds respectively. Therefore, batched parallel simulation favours both the proposed approach and DE against WEIBO again. GPU acceleration provides remarkable optimization speedup once more; compared to the cpu implementation, it is $\times 2.6$ faster and provides a total runtime speedup of $\times 42$ compared to WEIBO, for the same number of simulations.

In this experiment, employing multiple trust regions leads to better results. This can be explained as follows. Multiple trust regions traverse the variable space finding multiple paths into feasible sub-regions. By exploring different parts of the feasible variable space, the search for global optimum becomes more efficient. This however, comes with a cost in runtime, since more GP models need to be trained and sampled to provide with query points.

V. CONCLUSION

An approach for the automated sizing of analog circuits using local-based BO was presented. The proposed approach can handle high-dimensional problems by focusing on promising sub-regions of the design space. By using kernel function approximations and GPU accelerated GP inference, the proposed approach scales efficiently to large sample budgets. Experiments on two real-world circuits verify its effectiveness.

ACKNOWLEDGMENT

This research is co-financed by Greece and the European Union (European Social Fund- ESF) through the Operational Programme "Human Resources Development, Education and Lifelong Learning" in the context of the project "Strengthening Human Resources Research Potential via Doctorate Research" (MIS-5000432), implemented by the State Scholarships Foundation (IKY).

REFERENCES

[1] Y. Wang, M. Orshansky, and C. Caramanis, "Enabling efficient analog synthesis by coupling sparse regression and polynomial optimization," in *Proceedings of the 51st Annual Design Automation Conference*, 2014, pp. 1–6.

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

[3] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[4] S. Zhang, W. Lyu, F. Yang, C. Yan, D. Zhou, X. Zeng, and X. Hu, "An efficient multi-fidelity bayesian optimization approach for analog circuit synthesis," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*, 2019, pp. 1–6.

[5] W. Lyu, F. Yang, C. Yan, D. Zhou, and X. Zeng, "Multi-objective bayesian optimization for analog/rf circuit synthesis," in *Proceedings of the 55th Annual Design Automation Conference*, ser. DAC '18. New York, NY, USA: Association for Computing Machinery, 2018.

[6] B. He, S. Zhang, F. Yang, C. Yan, D. Zhou, and X. Zeng, "An efficient bayesian optimization approach for analog circuit synthesis via sparse gaussian process modeling," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 67–72.

[7] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2, no. 3.

[8] J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson, "Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration," in *Advances in Neural Information Processing Systems*, 2018, pp. 7576–7586.

[9] D. Eriksson, M. Pearce, J. Gardner, R. D. Turner, and M. Poloczek, "Scalable global optimization via local bayesian optimization," in *Advances in Neural Information Processing Systems*, 2019, pp. 5496–5507.

[10] P. C. Roy, R. Hussein, J. Blank, and K. Deb, *Trust-Region Based Multi-objective Optimization for Low Budget Scenarios: 10th International Conference, EMO 2019, East Lansing, MI, USA, March 10-13, 2019, Proceedings*, 01 2019, pp. 373–385.

[11] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer methods in applied mechanics and engineering*, vol. 186, no. 2-4, pp. 311–338, 2000.

[12] H. Liu, Y.-S. Ong, X. Shen, and J. Cai, "When gaussian process meets big data: A review of scalable gps," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[13] A. Wilson and H. Nickisch, "Kernel interpolation for scalable structured gaussian processes (kiss-gp)," in *International Conference on Machine Learning*, 2015, pp. 1775–1784.

[14] G. Pleiss, J. Gardner, K. Weinberger, and A. G. Wilson, "Constant-time predictive distributions for Gaussian processes," ser. Proceedings of Machine Learning Research, vol. 80. PMLR, 2018, pp. 4114–4123.

[15] B. Liu, F. V. Fernandez, and G. G. E. Gielen, "Efficient and accurate statistical analog yield optimization and variation-aware circuit sizing based on computational intelligence techniques," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 6, pp. 793–805, 2011.

[16] Z. Yan, P. Mak, M. Law, and R. P. Martins, "A 0.016-mm^2 144-μ w three-stage amplifier capable of driving 1-to-15 nf capacitive load with $>$0.95-mhz gbw," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 2, pp. 527–540, 2013.

[17] B.-K. Kim, D. Im, J. Choi, and K. Lee, "A highly linear 1 ghz 1.3 db nf cmos low-noise amplifier with complementary transconductance linearization," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 6, pp. 1286–1302, 2014.