Attention-Based EDA Tool Parameter Explorer: From Hybrid Parameters to Multi-QoR Metrics

Donger Luo, Qi Sun, Peng Xu, Su Zheng, Qi Xu, Tinghuan Chen, Bei Yu, Hao Geng

***Abstract*—Improving the outcomes of very-large-scale integration design without altering the underlying design enablement, such as process, device, interconnect, and IPs, is critical for integrated circuit (IC) designers. Parameter tuning for electronic design automation (EDA) tools is an emerging technology for improving the final design Quality-of-Result (QoR). It can be observed that many complex heuristics have been accreted upon previous complex heuristics integrated into tools, resulting in a vast number of tunable parameters. Even worse, these parameters include both continuous and discrete ones, making the parameter tuning process laborious and challenging. In this paper, we propose an attention-based EDA tool parameter explorer. A self-attention mechanism is developed to navigate the parameter importance. A hybrid space Gaussian process model is leveraged to optimize continuous and discrete parameters jointly, capturing their complex interactions. Considering multiple QoR metrics and the large amount of time required to invoke EDA tools, a customized acquisition function based on expected hypervolume improvement (EHVI) is proposed to enable multi-objective optimization and parallel evaluation. In addition, a self-adjusting additive kernel is proposed to optimize the hybrid space Bayesian Optimization flow and increase its explainability. Experimental results on a set of IWLS2005 benchmarks demonstrate the effectiveness and efficiency of our method.**

I. Introduction

THE ever-increasing design complexity plus market pressures for optimal Quality-of-Result (QoR), such as timing, power, area, etc., poses an intractable challenge to modern IC design flows. Physical design and logic synthesis, which involve several processes as seen in Fig. 1(a), are not an exception. Therefore, electronic design automation (EDA) tools for logic synthesis and physical design have to be incessantly evolving to cope with the ever-growing design demands. Parameter tuning for EDA tools is an emerging technology to achieve good QoR metrics without too much human intervention.

However, modern EDA tools incorporate numerous optimization heuristics that have been built upon previous complex heuristics, leading to inherently unexpected outcomes. Additionally, the designer has access to countless parameter options and parameter combinations owing to the unstopping upgrading and incorporation of algorithms of tools. Thus, even for sophisticated designers, finding the optimal or near-optimal parameter combination within this huge tool parameter space is challenging. Unfortunately, using EDA tools takes a lot of time, making it impossible for designers to exhaustively enumerate all possible parameter combinations and find the optimal or near-optimal one. Automatically tuning parameter settings is therefore crucial.

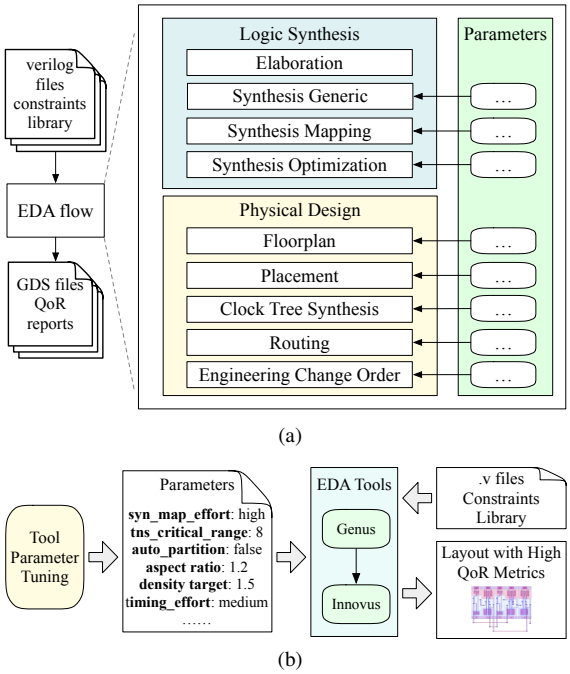


Fig. 1 The visualization of a typical synthesis flow and the working flow of the tool parameter tuner.

In recent years, several works have been proposed that enable automatic parameter tuning utilizing heuristics or machine learning techniques. FlowTuner [1] incorporates the ant colony optimization (ACO) algorithm into a co-evolutionary framework. The AutoTuner platform [2] integrates several optimization algorithms, such as evolutionary algorithms and tree-structured Parzen estimators. LAMBDA [3] employs a machine learning adjustment framework based on the XGBoost [4] regressor, coupled with an early stopping mechanism to speed up FPGA design. [5] harnesses a tensor decomposition-based recommender algorithm to tune

This work is partially supported by The Research Grants Council of Hong Kong SAR (No. CUHK14211824 and CUHK14210723). (*Corresponding authors: Hao Geng*)

Donger Luo and Hao Geng are with the School of Information Science and Technology, ShanghaiTech University. H. Geng is also with the Shanghai Engineering Research Center of Energy Efficient and Custom AI IC.

Qi Sun is with Zhejiang University.

Su Zheng, Peng Xu, and Bei Yu are with Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong SAR.

Qi Xu is with School of Microelectronics, University of Science and Technology of China, Hefei, China.

Tinghuan Chen is with Shandong Yunhai Guochuang Cloud Computing Equipment Industry Innovation Co., Ltd, Jinan 250101, China. He is also with the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, Guangdong 518172, China.

some synthesis and physical design tool parameters. Agnesina *et al*. [6] optimize the placement parameters via a deep reinforcement learning framework fed with a mixture of handcrafted features and graph embeddings extracted via Graph Neural Networks. In [7], the active learning-based approach exploiting the feature importance sampling and XGBoost regressor is developed for tool parameter tuning. By incorporating the transfer Gaussian process model, PPAtuner [8] can learn the transfer knowledge from the existing tool parameter combinations autonomously.

However, most of the aforementioned tuning algorithms require rich and representative data to train a high-quality QoR predictor or regressor. In the tuning context, the total number of samples available in a given dataset is limited by a lack of relevant test cases. Worse, long synthesis and physical design tool runtimes can be a bottleneck in collecting enough samples. Bayesian optimization, which performs efficient global searches by balancing exploration and exploitation, is a promising tuning framework. It has achieved some success on some EDA problems. For example, Zhang *et al*. [9] propose a Bayesian optimization approach for analog circuit synthesis using neural network, and they further present a multi-fidelity Bayesian optimization approach by fusing the simple but potentially inaccurate low-fidelity model and a few accurate but expensive high-fidelity data [10]. In the context of EDA tool parameter tuning, [11] solves the parameter selection problem of EDA tools through Bayesian optimization, while PTPT [12] establishes the tuning framework based on multi-objective Bayesian optimization and multi-task Gaussian models.

It is worth noticing that the EDA tool parameter space is often composed of discrete parameters, while parameters of EDA tools consist of both discrete parameters and continuous parameters, as is shown in Fig. 1(b). Prior arts often simplify the issue by transforming continuous parameters into discrete parameters. Although this makes optimization convenient, it prevents continuous parameters from unleashing their potential to achieve optimal design QoR metrics. In our previous work, we propose the principal method of constructing diffusion kernels in a hybrid space using the formula of additive kernel functions, which enables Bayesian optimization to solve the parameter tuning problem in a discrete-continuous hybrid space. This approach shows promise, but the potential of the kernel's additive structure is not fully realized. Thus, in this paper, the constructed additive kernel is orthogonalized to increase the model's certainty and generate analytical Sobol indices, enabling the model to self-adjust and improve explainability. We also apply the self-attention mechanism to process the input of Bayesian optimization, so that the important relationship between parameters can be better captured. Considering multiple QoR metrics and the large amount of time required to invoke EDA tools, a customized acquisition function based on expected hypervolume improvement (EHVI) and Sobol indices is proposed to enable multi-objective optimization and parallel evaluation. Our main contributions are as follows:

- We have constructed a hybrid space multi-objective Bayesian optimization process that is more closely aligned with real IC design scenarios.
- An additive kernel is built as the core part of the Bayesian optimization to optimize continuous and discrete parameters in the hybrid space.
- The additive kernel is enhanced to be orthogonal for more accurate modeling of the hybrid design space. At the same time, the analytical Sobol indices make the additive Gaussian process model flexible and explainable.
- Through the self-attention mechanism, the weights of important parameters are navigated to better obtain the optimal solution.
- A customized acquisition function based on EHVI and Sobol indices is used, and parallel computing is used to evaluate multiple points simultaneously to accelerate computation.

The rest of the paper is organized as follows. Section II introduces some prior knowledge of Bayesian optimization and provides a problem formulation. Section III sketches the whole optimization flow. Section IV describes the details of attention-based hybrid space Bayesian optimization. Section V introduces the techniques of the enhanced additive kernel for hybrid space Bayesian optimization. Section VI presents the experimental results followed by the conclusion and future work in Section VIII.

II. Preliminaries

A. Additive kernel for Gaussian Process

[13] considers building a Gaussian process (GP) model with the additive structure by breaking down $f(x)$ into simpler component functions:

$$f(\boldsymbol{x}) = f_1(x_1) + f_2(x_2) + \cdots + f_{12}(x_1, x_2) + \cdots + f_{12\ldots D}(x_1, x_2, \ldots x_D), \quad (1)$$

where D is the number of $\boldsymbol{x}$'s dimensions, and $f_{ij}(x_i, x_j)$ is the component function whose input only includes the i-th and j-th dimension of $\boldsymbol{x}$. The kernel's decomposition can be constructed in the following manner, which enforces the additive structure of the function decomposition in a GP model: Assign a one-dimensional base kernel $k_i(x_i, x_i')$ for each dimension $i \in 1...D$; then define the d^{th} order additive kernel k_{add_d} as:

$$k_{add_d}(x, x') = \sigma_d^2 \sum_{1 \le i_1 \le i_2 < \cdots < i_d < D} \left[\prod_{l=1}^{d} k_{i_l}(x_{i_l}, x'_{i_l}) \right]. \quad (2)$$

The additive kernel for the GP model is constructed by summing all of the orders up to the dimensionality of the data. The parameters σ_d^2 control the relative importance of functions with different dimensions in the sum.

B. Bayesian Optimization

Bayesian optimization (BO) is a sequential design strategy for the global optimization of noisy black-box functions. It builds surrogate models to mimic the unknown black-box objective functions, *i.e.*, the complicated EDA flow in our context. The GP model is widely used as the surrogate model, which provides a posterior distribution of functions given prior and observed data.

A GP model is defined as a collection of random variables, any finite number of which have a joint Gaussian distribution. A GP is completely specified by its mean function $m(\boldsymbol{x})$ and covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$:

$$f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')), \tag{3}$$

where $\boldsymbol{x}$ represents the input variable vector. The objective in BO is to find the global optimal $\boldsymbol{x}^*$ of an unknown objective function f:

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x} \in \mathbb{R}^d} f(\boldsymbol{x}). \tag{4}$$

To guide the search for the optimal $\boldsymbol{x}$, an acquisition function $Ac(\mathcal{GP}, \boldsymbol{x})$ is built on the already-sampled variable vectors $\boldsymbol{x}$ and the surrogate model $\mathcal{GP}$, and utilizes the prior knowledge to evaluate the unsampled variable vectors.

C. Multi-objective Bayesian Optimization

Real-world optimization problems often involve multiple conflicting objectives. Multi-objective Bayesian optimization (MOBO) extends the standard BO to tackle such multi-objective problems. Let functions $\{f_i(\boldsymbol{x})\}_{i=1}^m$ denote the m-dimension minimization objectives, $\boldsymbol{X}$ denotes the parameter space. A parameter vector $\boldsymbol{x}^* \in \boldsymbol{X}$ is said to dominate $\boldsymbol{x}' \in \boldsymbol{X}$, denoted as $\boldsymbol{x}^* \succeq \boldsymbol{x}'$, if the following two conditions are met in this minimization problem:

1) $f_i(\boldsymbol{x}') \geq f_i(\boldsymbol{x}^*), \forall i \in \{1, ..., m\}$,
2) There exists at least one j such that $f_j(x') > f_j(\boldsymbol{x}^*)$.

The set of parameter vectors that are not dominated by other vectors is called the Pareto-optimal set, denoted as $\boldsymbol{X}_{\text{Pareto}}$, which is the target of MOBO:

$$\boldsymbol{X}_{\text{Pareto}} = \{\boldsymbol{x}^* \in \boldsymbol{X} | \nexists \boldsymbol{x}' \in \boldsymbol{X}, \boldsymbol{x}' \succeq \boldsymbol{x}^*\}. \tag{5}$$

Sometimes we would use $\boldsymbol{y}^* \succeq \boldsymbol{y}'$, which also means that $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ dominate $(\boldsymbol{x}', \boldsymbol{y}')$. For the problem with m optimization objectives, Bayesian optimization builds m surrogate models for these objectives, denoted as $\boldsymbol{M}$, to mimic the unknown black-box objective functions. Further, an acquisition function $Ac(\boldsymbol{M}, \boldsymbol{x})$ is built based on the surrogate models and estimates the quality of a parameter vector $\boldsymbol{x}$ with respect to finding the Pareto set, with no need to run the time-consuming EDA flow.

D. Problem Formulation

In the context of parameter tuning, the parameter space is of hybrid types, both continuous and discrete. Thus, a parameter vector $\boldsymbol{x} = (\boldsymbol{x}^c, \boldsymbol{x}^d)$ contains $\boldsymbol{x}^c \in \mathbb{R}^c$ representing continuous variables and $\boldsymbol{x}^d \in \mathbb{Z}^d$ representing discrete variables, while the 3 metrics, performance, power, and area, should be optimized simultaneously.

Problem 1 (Parameter Tuning for EDA synthesis Tool)**.** *Given the boundary of hybrid parameters of EDA tools and the QoR metrics to be optimized, the objective of EDA tool parameter tuning is to automatically search the Pareto-optimal parameter configurations, which bring about the high design quality concerning multiple QoR metrics like delay versus power/area and delay versus power versus area.*

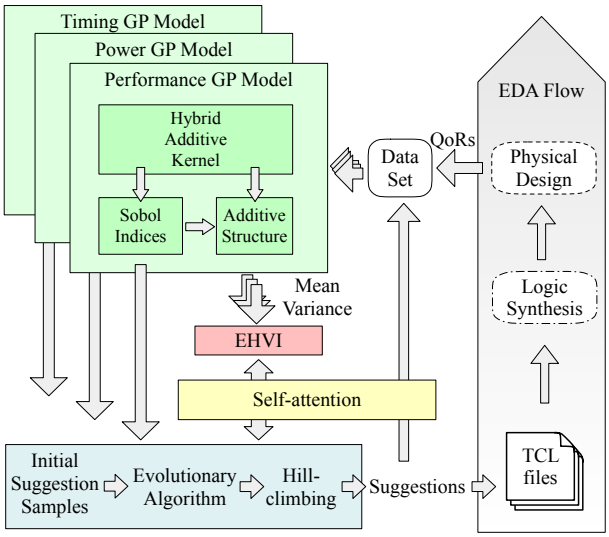


Fig. 2 The architecture of our Explorer.

III. Hybrid Space Bayesian Optimization Flow

In this section, we introduce a novel multi-objective hybrid Bayesian optimization approach, capable of efficiently handling hybrid (discrete and continuous) parameter spaces. The primary goal of this approach is to simultaneously optimize multiple conflicting objectives, thereby providing a set of Pareto-optimal solutions.

The overall architecture of our approach is shown in Fig. 2. The EDA flow takes in the suggested set of parameters as TCL (tool command language) files and then operates the logic synthesis and physical design. After that, the QoR information of the design under current parameters can be extracted from the report files. This QoR information is appended to the data set as $\boldsymbol{Y}$ along with the set of parameters as $\boldsymbol{X}$, which was transformed using the attention function. A hybrid Gaussian process regression model is built using the data set for each metric of QoR. These models offer mean and variance information for EHVI acquisition to select the next set of suggestions. The next set of suggestions to evaluate is selected from random samples, they are then transformed using the attention function, and each gets an EHVI score. The K samples with top K scores will be optimized with evolutionary algorithm and hill-climbing, and become the next suggested set of parameters.

The proposed approach is encapsulated in Algorithm 1, which outlines the steps taken in each iteration of the optimization process. In line 1, the algorithm begins by randomly sampling the initial parameter vectors $\boldsymbol{X}_0$ from the parameter space. The parameter vectors are then fed into the EDA tools to run designs to get ground-truth performance values $\boldsymbol{Y}_0$. For simplicity, we denote the EDA tools as $\mathbb{EDA}$. The $\boldsymbol{X}_0$ is then transformed using the attention function to learn better inputs to the Bayesian optimization, denoted as $atten(\boldsymbol{X}_0)$. Note that the attention function is non-parametric and easy to be computed given $\boldsymbol{x}$, as discussed in Section IV-A. The initial data set is $\boldsymbol{D}_0 = \{\boldsymbol{X}_0, atten(\boldsymbol{X}_0), \boldsymbol{Y}_0\}$. In each iteration (lines 3 to 20), we calibrate a set of additive GP models $\boldsymbol{M}$,

Algorithm 1 Multi-objective Hybrid BO Approach

Input: Hybrid input domain $\boldsymbol{X}$, acquisition function $Ac(\cdot,\cdot)$, tool flow $\mathbb{EDA}$, attention function $atten(\boldsymbol{x})$, optimization budget t_{max}, number of samples K in each step.
Output: The set of Pareto-optimal solutions $\boldsymbol{X}_{\text{Pareto}}$.

1: **Initialization**: sample initial input $\boldsymbol{X}_0$ with $\boldsymbol{X}_0 \subset \boldsymbol{X}$, $\boldsymbol{Y}_0 = \mathbb{EDA}(\boldsymbol{X}_0)$, $\boldsymbol{D}_0 = \{\boldsymbol{X}_0, atten(\boldsymbol{X}_0), \boldsymbol{Y}_0\}$;
2: **for** $t \leftarrow 1$ to t_{max} **do**
3: $\quad \mathcal{X}_{t-1} = atten(\boldsymbol{X}_{t-1})$;
4: $\quad$ **for** each of the m metrics **do**
5: $\quad\quad$ **for** each dimension in $\mathcal{X}_{t-1}$ **do**
6: $\quad\quad\quad$ Discrete dimension $\leftarrow$ diffusion kernel;
7: $\quad\quad\quad$ Continuous dimension $\leftarrow$ RBF kernel;
8: $\quad\quad$ **end for**
9: $\quad\quad$ Fit GP model set $\boldsymbol{M}_t$ with the enhanced additive kernel; ▷ Section V-A
10: $\quad\quad$ Update the Sobol indices SI_m for each parameter combinations; ▷ Equation (25)
11: $\quad\quad$ Construct posterior distribution over the additive function identified by SI_m; ▷ Equation (24)
12: $\quad$ **end for**
13: $\quad$ Select the next candidate set to evaluate: $\boldsymbol{X}_t \leftarrow$ Top K $Ac(\boldsymbol{M}_{t;1,\ldots,m}, atten(\boldsymbol{x}))$, $\forall \boldsymbol{x} \in \boldsymbol{X}$:
14: $\quad\quad$ Initialize suggestion samples randomly;
15: $\quad\quad$ Fixing discrete parameters, optimize continuous parameters $\boldsymbol{x}_c$ through evolutionary algorithm;
16: $\quad\quad$ Fixing continuous parameter, optimize discrete parameters $\boldsymbol{x}_d$ through hill-climbing;
17: $\quad$ Evaluate next candidate set $\boldsymbol{Y}_t = \mathbb{EDA}(X_t)$;
18: $\quad$ Update $\boldsymbol{D}_t \leftarrow \boldsymbol{D}_t \cup \{\boldsymbol{X}_t, atten(\boldsymbol{X}_t), \boldsymbol{Y}_t\}$;
19: $\quad$ Update $\boldsymbol{X} \leftarrow \boldsymbol{X} \backslash \boldsymbol{X}_t$;
20: **end for**
21: Select Pareto vectors $\boldsymbol{X}_{\text{Pareto}}$ according to $\boldsymbol{D}_{t_{max}}$;
22: **return** $\boldsymbol{X}_{\text{Pareto}}$.

one for each objective (line 4). A base kernel is assigned to each input dimension: a diffusion kernel for a discrete dimension, or an RBF kernel for a continuous dimension (lines 5 to 7). The posterior distribution is constructed using the additive function associated with Sobol indices for each parameter combination (lines 9 to 11). The next candidate set for evaluation is composed of parameter vectors with the top-K $Ac(\boldsymbol{M}, atten(\boldsymbol{X}_0))$ values over the current GP models. The selected candidates are then fed into the EDA tools to evaluate the performance, and the results are added to the sample set for the next iteration (lines 17 to 18). This process is repeated until the maximum number of iterations is reached. Finally, the algorithm returns the set of all Pareto-optimal solutions from the sampled set. Note that given the sampled set $\boldsymbol{D}_{t_{max}}$, it is easy to select its Pareto-optimal vectors according to Equation (5) (line 21). In other words, our final solutions are obtained from the sampled parameter vectors $\boldsymbol{X}_{t_{max}}$.

This proposed method presents a systematic approach to multi-objective optimization in hybrid tool parameter spaces, efficiently handling the discrete and continuous variables. By leveraging the Gaussian process models for each objective, it allows for uncertainty-aware exploration of the parameter space, and enables the identification of the Pareto-optimal solutions, thereby providing a comprehensive set of optimal trade-off parameters.

IV. Attention-based Hybrid Space Bayesian Optimization

In this section, we detail the methodology we employed to carry out attention-based Bayesian optimization (BO) for optimizing QoR metrics in a hybrid space, which includes both continuous and discrete parameters. This is achieved using the concept of additive hybrid diffusion kernels.

A. Attention Mechanism in Bayesian Optimization

The attention mechanism, originating from the field of deep learning, is an effective mechanism for encoding the dependencies between different parts of the input data. In the context of EDA tool parameter tuning, we adapt this concept to model the relationships between different parameters within our hybrid input parameter space. Our approach is inspired by self-attention, where interactions between different elements (in our case, EDA tool parameters) are assessed. However, to maintain computational simplicity and avoid introducing new hyperparameters, we employ a degenerate form of self-attention. Note that the proposed attention algorithm operates separately on discrete and continuous parameter spaces since their GP kernels are formed separately. Then, the final weighted parameters, discrete and continuous parameters, are sent into different GP kernels.

The attention mechanisms we have designed for discrete and continuous parameters are the same. For simplicity, we exploit $\boldsymbol{x}$ to represent a parameter configuration vector for either discrete or continuous parameters. x_i represents the i-th element (i.e., tool parameter). The attention module computes an importance score for each parameter. In a typical self-attention mechanism, each "token" (here, each parameter x_i) would be projected into query, key, and value vectors, and pairwise dot products of queries and keys would measure relevance. To keep our procedure computationally negligible and to avoid new hyperparameters, we choose a degenerate setting where the query, key, and value projections are all the identity. Consequently, the relevance between parameters x_i and x_j collapses to the simple product $x_i x_j$. Summing these scalar products over all $j \neq i$ yields the importance of parameter x_i:

$$\text{Importance}(x_i) = \sum_{j=1, j\neq i}^{n} x_i \cdot x_j. \tag{6}$$

Although Equation (6) appears straightforward, it effectively embeds every second-order cross-interaction once, akin to how self-attention would, but without any learned weights. This absence of learned weights is a deliberate design choice integral to our Bayesian Optimization (BO) framework. In BO, the Gaussian Process (GP) model itself is the primary learning component, using its kernel and associated hyperparameters (like length scales) to model the objective function

landscape based on observed data. Our attention-inspired mechanism is intended as a computationally lightweight, deterministic input transformation that acts as a heuristic to guide the GP by pre-emphasizing potential parameter interactions. Introducing learnable weights at this stage would necessitate an additional optimization process for these weights, potentially increasing the demand for scarce evaluation data (EDA tool runs) and adding complexity that could obscure the GP's own learning process.

This importance score, based on aggregated pairwise relationships, is then used for input warping, which is an effective method to optimize the modeling capability of the Gaussian process [14], [15]. Further, the updated parameter value to the GP model is a warped version of the original input parameter, obtained by weighting it with its importance score:

$$\text{atten}(x_i) = x_i \cdot \text{Importance}(x_i). \tag{7}$$

Through Equation (7), the input values in more important dimensions are magnified. Because dimensions with larger importance are magnified before the GP kernel is evaluated, they effectively acquire shorter length-scales under the same hyper-prior, encouraging the GP to explore them more thoroughly [16]. No extra hyper-parameters are introduced by this warping, and it can be trivially inverted when the GP proposes a new point, ensuring all suggested configurations remain valid for the EDA tool.

This attention mechanism allows our Bayesian optimization process to consider complex dependencies between different dimensions in the input space, while it is easy to compute with no further burden. These values are easy to rescale back to practical parameters as inputs to the EDA tools since the ranges of these parameters are already known. Therefore, using attention mechanisms can simultaneously facilitate Gaussian process models and ensure the efficient use of EDA tools.

B. Hybrid Space Gaussian Process

Kernels of Continuous Space Gaussian Processes. GPs are a powerful tool for probabilistic non-parametric inference, allowing us to mimic unknown functions in continuous space to make predictions given some observed data. In Gaussian processes (GPs), kernels play a crucial role in defining the covariance structure of the unknown function that we aim to model.

The Gaussian distribution, characterized by its mean and covariance, can be specified using a mean function and a covariance function (kernel) in the context of GPs. The mean function typically represents prior knowledge about the function, while the kernel captures the smoothness, periodicity, and other properties of the function. Specifically, the kernel function quantifies the similarity or correlation between pairs of points in the input space.

Given a set of single-objective observations, the kernel $k(\boldsymbol{x}, \boldsymbol{x}')$ defines the covariance between the function values

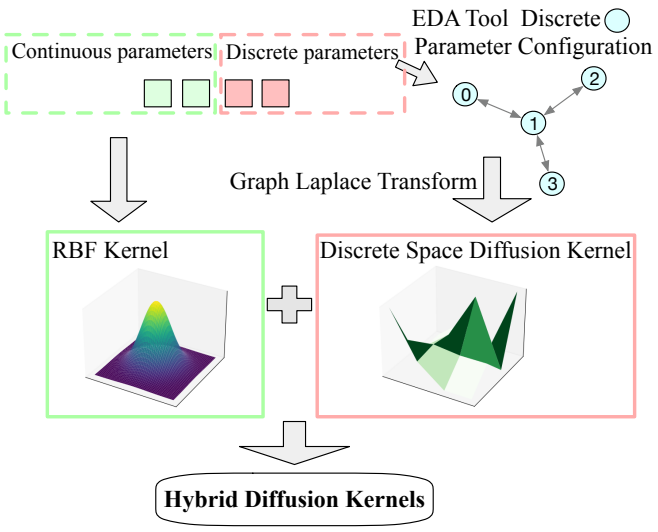


Fig. 3 Construction of Hybrid Diffusion Kernels. RBF kernel is used for continuous parameters, while the discrete kernel is based on a graph representation of the discrete parameter space. For the RBF kernel, X and Y axis in the picture represent the value of the two continuous inputs, respectively, while the Z axis represents the value of Equation (9) with $\boldsymbol{x'}$ fixed to $(0, 0)$. For the discrete space diffusion kernel, the two discrete parameters are first transferred into a graph where each node of the graph represents a combination of discrete parameters. In the picture, X and Y axis represent the graph inputs, while the Z axis represents the value of Equation (11).

at two input points $\boldsymbol{x}$ and $\boldsymbol{x}'$. In GPs, the prior over functions can be written as:

$$f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')), \tag{8}$$

where $m(\boldsymbol{x})$ is the mean function and $k(\boldsymbol{x}, \boldsymbol{x}')$ is the kernel function. A common choice of kernel in continuous space is the radial basis function (RBF) kernel, also known as the squared exponential kernel:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{x}'||^2}{2s^2}\right), \tag{9}$$

where σ^2 is the signal variance and s is the length scale. The kernel function and its associated hyperparameters have a significant impact on the GP's predictive performance, making the choice of an appropriate kernel essential in GP modeling.

Diffusion Kernels for Discrete Spaces. The diffusion kernel is a powerful method to characterize the discrete parameter space by constructing a graph for the discrete parameter vectors and simulating the diffusion or random walk process over the graph. Denote the graph as $\boldsymbol{G}$, in which each node is a parameter vector in our sampled parameter vector set. Two nodes are connected if their parameter vectors have only one different element value. In other words, the two connected vectors can reach each other by changing only one parameter option. Denote the node set as $\boldsymbol{V}$ (*i.e.*, the set of discrete parameter vectors), the adjacency matrix as $\boldsymbol{A}$, and the degree

matrix as $\boldsymbol{D}$, respectively. Then the normalized form of the Laplacian matrix is defined as:

$$\boldsymbol{L}(\boldsymbol{G}) = \mathbf{I} - \boldsymbol{D}^{(-1/2)}\boldsymbol{A}\boldsymbol{D}^{(-1/2)}, \tag{10}$$

where $\mathbf{I}$ is the identity matrix. The normalized Laplacian scales the influence of each node by its degree. This is particularly useful for our case with an increasing number of sampled parameter vectors as the Bayesian optimization iterates. The diffusion kernel over our graph node set $\boldsymbol{V}$ is defined in Equation (11):

$$\boldsymbol{K}(\boldsymbol{V}, \boldsymbol{V}) = \boldsymbol{\Phi}\exp(-\gamma\boldsymbol{\Pi})\boldsymbol{\Phi}^T, \tag{11}$$

where γ is a hyper-parameter, $\boldsymbol{\Phi} = [\phi_1, \cdots, \phi_{|\boldsymbol{V}|}]$ and $\boldsymbol{\Pi} = [\pi_1, \cdots, \pi_{|\boldsymbol{V}|}]$ are the eigenvector matrix and eigenvalue matrix of $\boldsymbol{L}$, respectively. The diffusion kernel provides a powerful way of understanding the structure of discrete spaces and graphs, taking into account not just the immediate connections between nodes, but also longer pathways through the graph. It has a solid mathematical foundation based on the Laplacian operator and the heat diffusion process, and it can be computed efficiently in closed forms for large graphs [17], [18].

Hybrid Diffusion Kernels. Hybrid diffusion kernels are designed to handle input spaces that consist of both continuous and discrete parameters. These kernels are constructed by combining the continuous and discrete diffusion kernels in a way that allows for seamless integration of both types of input spaces.

To construct hybrid diffusion kernels, we can use the general formulation of additive Gaussian process kernels, which defines an additive hybrid diffusion kernel over hybrid spaces. The key idea is to assign a base kernel for each input dimension $i \in \{1, 2, \cdots, n_c + n_d\}$, where n_c and n_d represent the number of discrete and continuous parameters in the hybrid space $\boldsymbol{X}$. The RBF kernel and the discrete diffusion kernel act as base kernels for continuous and discrete input dimensions, respectively, as shown in Fig. 3. The overall kernel is constructed by summing all possible orders of interactions between these base kernels.

The overall additive hybrid diffusion kernel $\mathcal{K}_{HBS}(\boldsymbol{x}, \boldsymbol{x}')$ over hybrid spaces is defined as:

$$\mathcal{K}_{HBS} = \sum_{p=1}^{n_c+n_d} \left(\theta_p^2 \sum_{i_1,\cdots,i_p} \prod_{d=1}^{p} k_{i_d}\left(\boldsymbol{x}_{i_d}, \boldsymbol{x}'_{i_d}\right) \right), \tag{12}$$

where θ_p is a hyper-parameter associated with each additive kernel, $1 \le i_1 < i_2 < \cdots < i_p \le n_c + n_d$ denotes the possible selections of kernels, k_{i_d} is the base kernel for the input dimension i_d. By using hybrid diffusion kernels, we can effectively perform Bayesian optimization over input spaces with mixed continuous and discrete parameters, which is particularly useful for applications like EDA tool tuning.

C. Parallel Evaluation based on EHVI

The acquisition function is a crucial component in Bayesian optimization, determining where to sample next. Among many alternatives, the expected hypervolume improvement (EHVI) has shown superior performance in handling multi-objective problems [19].

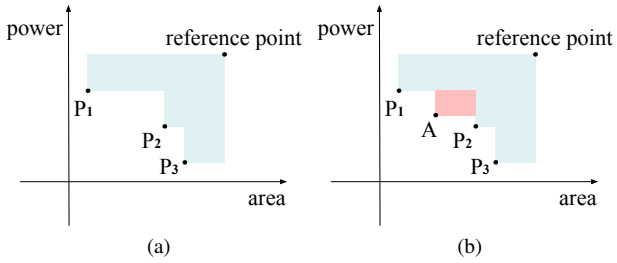


Fig. 4 Example of hypervolume improvement in 2-d space.

Hypervolume is a measure of the space covered by a set of solutions in the objective space. Specifically, it corresponds to the volume of the region that is dominated by at least one solution in the set. Hypervolume is commonly utilized as a performance metric in evolutionary multi-objective optimization, where it offers a comprehensive evaluation of the quality of the solution set, taking into account not only the convergence to the true Pareto front but also the diversity of solutions.

Given a set of Pareto-optimal parameter vectors $\boldsymbol{X}_{\text{Pareto}} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\}$, the corresponding performance set is $\boldsymbol{Y}_{\text{Pareto}} = \{\boldsymbol{y}_1, \cdots, \boldsymbol{y}_n\} \subseteq \mathbb{R}^m$, where m denotes the number of objectives. A newly sampled parameter vector $\boldsymbol{x}$ is regarded as Pareto-optimal if its performance vector $\boldsymbol{y}$ is not dominated by the current $\boldsymbol{Y}_{\text{Pareto}}$. Correspondingly, the improvement of the hypervolume resulting from sampling the Pareto-optimal $(\boldsymbol{x}, \boldsymbol{y})$ is defined in Equation (13).

$$I(\boldsymbol{y}) = \int_{\mathbb{R}^m} [\boldsymbol{y} \succeq \boldsymbol{y}' \wedge \forall i \in 1, \ldots, n : \boldsymbol{y}_i \not\succeq \boldsymbol{y}'] \mathrm{d}\boldsymbol{y}', \tag{13}$$

which computes the hypervolume composed of the solutions in the space $\mathbb{R}^m$ which are dominated by $\boldsymbol{y}$ but not dominated by previously found $\boldsymbol{Y}_{\text{Pareto}}$.

Fig. 4 is an example of hypervolume improvement in the 2-d space. Given a reference point and Pareto-optimal P_1, P_2, and P_3, the area covered in blue is the hypervolume of the Pareto sets. Point A, which is not dominated by the Pareto-optimal points, can bring an improvement to the hypervolume, so the size of the area in red is the hypervolume improvement regarding point A. Considering that the performance vector $\boldsymbol{y}$ is predicted via GP models, the expected hypervolume improvement EHVI is then defined as the expectation of $I(\boldsymbol{y})$ with respect to the posterior predictive distribution of the GP:

$$\text{EHVI}(\boldsymbol{y}) = \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{D})}[I(\boldsymbol{y})], \tag{14}$$

where $\boldsymbol{D}$ denotes the observed data. We then select the next sample point to be the one that maximizes the EHVI:

$$\boldsymbol{y}^* = \arg\max_{\boldsymbol{y}} \ \text{EHVI}(\boldsymbol{y}). \tag{15}$$

The reference point in the performance space is a virtual point, a hyperparameter set by ourselves, which does not have physical meanings and is only for ease of computation. It does not affect the performance of the proposed algorithm since what we compute is the "relative" improvements of the

hypervolume. Generally, the reference point is set as a very bad performance value point, such as one with a very bad power and a large area cost.

The EHVI acquisition function balances exploration and exploitation effectively, allowing for efficient optimization in the hybrid discrete-continuous space.

In our algorithm, the EHVI acquisition function is used to assign scores to a large number of points randomly selected from the hybrid space. Instead of selecting a single optimal point per iteration, we adopt a batch evaluation strategy to propose multiple promising suggestions simultaneously. So the most time-consuming part in the flow, EDA tool evaluation, can make full use of compute resources to evaluate multiple suggestions simultaneously. The K points with top K scores are selected for further optimization. Specifically, the discrete part of these K points undergoes optimization through a hill-climbing algorithm, while the continuous part is optimized using an evolutionary algorithm. This procedure ensures that the suggestions not only achieve high EHVI scores but also remain well-spread in the region with high EHVI scores. By integrating parallel evaluation, our algorithm is able to quickly and efficiently explore and exploit the hybrid space, making it highly effective for the problem of Bayesian optimization in hybrid spaces.

V. Self-adjusting and Explainable Additive Gaussian Process for Hybrid Space Bayesian Optimization

A. Enhanced Additive Kernel

In Section IV-B, the hybrid diffusion kernel is designed using the formulation of additive Gaussian process kernels. However, a key challenge with additive kernels is that different combinations of component functions can produce the same prediction, leading to ambiguity in interpretation [20]. To illustrate, consider a simple two-dimensional function:

$$f(x_1, x_2) = [g_1(x_1) + c] + [g_2(x_2) - c], \tag{16}$$

the additive decomposition works for any constant c, which causes the non-uniqueness of $f(x1, x2)$'s additive structure.

This non-uniqueness of the additive GP model's components makes it challenging to attribute effects to specific input parameters or their interactions reliably. To address this, we propose an enhanced additive kernel constraining the component functions to be orthogonal to each other. Specifically, we require:

$$\int g_i(x_i)p(x_i)dx_i = 0, \tag{17}$$

for each component function g_i, where $p(x_i)$ is the marginal density of input x_i.

When we condition the orthogonality condition given in Equation (17), we can demonstrate that the resulting conditional process remains a Gaussian process. Specifically, for each i, the original base kernel k_i is replaced by an enhanced kernel $\tilde{k}_i$:

$$\int g_i\left(x_i\right) p_i\left(x_i\right) dx_i = 0 \sim \mathcal{GP}\left(0, \tilde{k}_i\right). \tag{18}$$

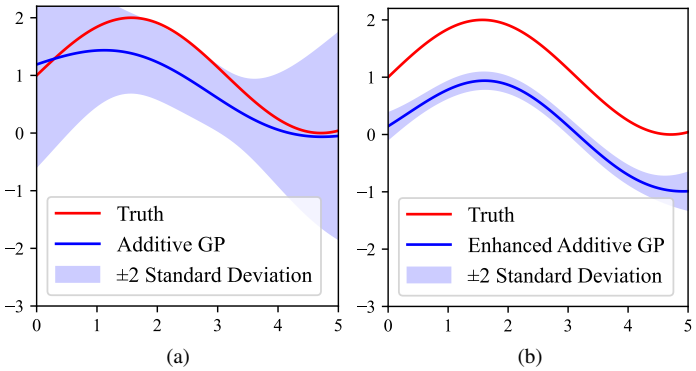


Fig. 5 Illustration of non-uniqueness of additive GP model's components. The orthogonality property in the enhanced additive GP brings smaller uncertainties, while the difference between truth and enhanced additive GP will be captured with the constant kernel.

We can incorporate this orthogonality constraint into a Gaussian process framework. The key idea is to construct a new constrained kernel for each component that enforces the desired orthogonality property while adding an additional GP g_0 with a constant kernel.

In other words, the constrained kernel is constructed to ensure that the corresponding component of the additive model captures a unique aspect of the function. So each component function $g_i(x_i)$ is separated into two parts: the part that depends on x_i, and the constant part that does not. One constrained kernel is assigned to each component function while one constant kernel is assigned to model the sum of all component functions' constant parts. Starting with a base kernel k_i for each input dimension, we define the constrained kernel by subtracting a correction term from the original kernel $k_i(x_i, x_i')$:

$$\tilde{k}_i(x_i, x_i') = k_i(x_i, x_i') - \mathbb{E}[T_i g_i(x_i)]\mathbb{E}[T_i^2]^{-1}\mathbb{E}[T_i g_i(x_i')], \tag{19}$$

where $T_i = \int g_i(x_i)p(x_i)dx_i$ and represents the average effect of the component function $g_i(x_i)$ when integrated over the probability distribution $p(x_i)$ of the input x_i. It quantifies how much $g_i(x_i)$ contributes on average. The correction term is composed of three parts: $\mathbb{E}[T_i g_i(x_i)]$ and $\mathbb{E}[T_i g_i(x_i')]$ is the expected contribution of $g_i(x_i)$ and $g_i(x_i')$ when projected onto its average T_i. It measures how closely $g_i(x_i)$ and $g_i(x_i')$ aligns with its average behavior. $\mathbb{E}[T_i^2]^{-1}$ is a normalization term based on the variance of the average contribution T_i. It ensures that the correction term is properly scaled and does not over-correct or under-correct the kernel.

This formulation guarantees that samples from the resulting GP will satisfy the orthogonality constraints while still allowing flexible modeling of additive interactions up to the desired order. By enforcing orthogonality, we can obtain a unique decomposition of the PPA function into components containing different parameters. Fig. 5 shows the comparison of the base kernel in normal additive GP and enhanced additive GP modeling a single-parameter component.

B. Explainable and Self-adjusting GP Model Based on Analytical Sobol Indices

When tuning EDA parameters for PPA metrics, understanding the influence of individual input parameters on the PPA metrics is crucial.

Despite the potential presence of numerous parameters, often only a small subset of parameters or interactions between them significantly impact the prediction of PPA metrics. To quantify this impact, we draw upon methods from global sensitivity analysis [21], particularly decomposition techniques based on Analysis of Variance [22] (ANOVA).

Functional ANOVA [23] (FANOVA) provides a systematic approach to decompose a multivariate function into main effects and interaction effects. For a function $\hat{y}(\boldsymbol{x})$, the FANOVA decomposition can be represented as:

$$\hat{y}(\boldsymbol{x}) = \sum_{a \subseteq N} \hat{y}_a(\boldsymbol{x}_a), \tag{20}$$

where $\hat{y}_a$ depends only on the parameters in $\boldsymbol{x}_a$ and is defined as:

$$\begin{aligned} \hat{y}_a(\boldsymbol{x}) &= \int_{\boldsymbol{X}_{-a}} \left(\hat{y}(\boldsymbol{x}) - \sum_{v \subset a} \hat{y}_v(\boldsymbol{x}_v) \right) dP(\boldsymbol{x}_{-a}) \\ &= \int_{\boldsymbol{X}_{-a}} \left(\hat{y_a}(\boldsymbol{x}) + \sum_{v \not\subseteq a} \hat{y}_v(\boldsymbol{x}_v) \right) dP(\boldsymbol{x}_{-a}) \\ &= \hat{y}_a\left(\mathrm{x}_a\right) + \sum_{v \not\subseteq a} \int_{\mathrm{X}_{-a}} \hat{y}_v\left(\mathrm{x}_v\right) dP\left(\mathrm{x}_{-a}\right), \end{aligned} \tag{21}$$

where $\boldsymbol{x}_{-a}$ denotes all parameters except x_a and $P(x)$ is x's posterior distribution. If the input parameters are independent of each other, Equation (21) holds when the integral part equals 0, which is the requirement of the kernel's orthogonality property in Equation (17). Notably, our additive kernel construction is closely tied to the FANOVA decomposition. Under the assumption of independent input features, the function components in the additive kernel correspond exactly to the components of the FANOVA decomposition.

The FANOVA decomposition naturally leads to the concept of Sobol indices [24], a measure quantifying the contribution of individual inputs or input combinations to the output variance. For a component $\hat{y}_a$, its Sobol indice is defined as:

$$SI := V_x[\hat{y}(\boldsymbol{x})] = \sum_{a \subseteq N} SI_a, \tag{22}$$

where parameter set a's Sobol index, SI_a, represents its contribution to the model's output variance. Essentially, SI_a quantifies the influence of parameter set a on the model's variability. This relationship is known as the ANOVA identity.

In the enhanced additive kernel model, we can compute the Sobol indices for each component analytically. Specifically, for the posterior mean function μ, the normalized Sobol indice is:

$$\tilde{SI}_a = \frac{V_{\boldsymbol{x}}[\mu_a(\boldsymbol{x})]}{V_{\boldsymbol{x}}[\mu(\boldsymbol{x})]}, \tag{23}$$

where

$$\mu_a(x) = \sigma^2_{|a|} \left(\odot_{i \in a} \tilde{k}_i(x_i, \boldsymbol{X}_i) \right) K(\boldsymbol{X}, \boldsymbol{X})^{-1} \boldsymbol{y}, \tag{24}$$

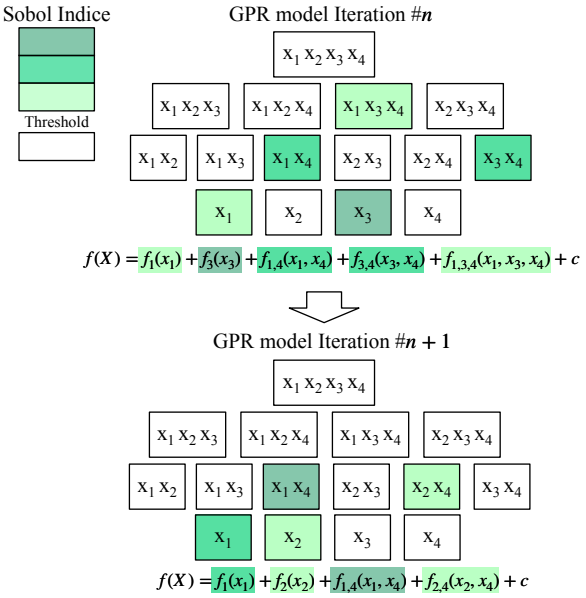


Fig. 6 The enhanced additive Gaussian process only considers parameters whose interactions' Sobol indices are above the threshold.

$K(\boldsymbol{X}, \boldsymbol{X})$ represents the covariance of all input parameters, the i-th column of X is denoted by $\boldsymbol{X}_i$ and outputs' observation is denoted by y, $\sigma^2_{|a|}$ is the variance for the parameters' $|a|$-th order interaction and the element-wise multiplication is indicated by $\odot$. Therefore, the Sobol indice of the input set a is:

$$\begin{aligned} V_x[\mu_a(x)] &= V_{\boldsymbol{x}}[\sigma^2_{|a|} \left(\odot_{i \in a} \tilde{k}_i(x_i, \boldsymbol{X}_i) \right) K(\boldsymbol{X}, \boldsymbol{X})^{-1} \boldsymbol{y}] \\ &= \sigma^4_{|a|} y^\top K(\boldsymbol{X}, \boldsymbol{X})^{-1} \odot_{i \in a} \mathcal{I}i K(\boldsymbol{X}, \boldsymbol{X})^{-1} \boldsymbol{y}, \end{aligned} \tag{25}$$

where $\mathcal{I}i = \int \tilde{k}i(x_i, \boldsymbol{X}i) \otimes \tilde{k}i(x_i, \boldsymbol{X}i) \mathrm{d}pi(x_i)$, and $\otimes$ indicates outer product.

In practice, we use Sobol indices to identify the most important single parameters or higher-order interactions between multiple parameters in the model. By ranking these interactions according to their Sobol indices, we can determine the minimal interactions between parameters needed to achieve the desired predictive performance. In each iteration of the optimization flow, the GPR model adjusts itself to modeling PPA metrics with parameters and their interactions with Sobol indices higher than the threshold value, as is shown in Fig. 6. This approach not only enhances model interpretability but often leads to less complicated additive representations.

C. Sobol-based Sampling Optimization for Acquisition Function

In a Bayesian Optimization flow, the acquisition function calculates a score for each sample from the design space and chooses the sample with the highest score as the suggestion. However, it is impossible to calculate the score for every point in the design space. Section IV-C introduces a flow to optimize the suggestion using an evolutionary algorithm and hill-climbing algorithm for continuous and discrete parameters,

TABLE I Benchmark Designs

Name	#Cells	Clock Period (ps)
8bit_risc_cpu	976	500
des3_area	3097	700
b15	12014	1000
b19	48398	1200
vga	73898	400

TABLE II Examples of Parameters

Parameters	Stage	Range
`syn_generic_effort`	synthesis	high/medium/low
`syn_map_effort`		high/medium/low
`tns_critical_range`		0-15
`auto_partition`		true/false
`aspect ratio`	floorplan	0.5-2.0
`target density`		0.5-1.0
`place_detail_wire_length_opt_effort`	placement	high/medium/none
`place_global_cong_effort`		auto/high/medium/low
`place_global_timing_effort`		high/medium
`drouteUseMultiCutViaEffort`	routing	high/medium/low
`routeWithLithoDriven`		true/false

respectively. The Sobol indices can benefit the evolutionary algorithm and hill-climbing algorithm in different ways to accelerate optimization.

With a CMA-ES [25] algorithm used as the evolutionary algorithm, the Sobol indices help initialize the covariance matrix by increasing the value of the diagonal element corresponding to the parameters with higher Sobol indices, enabling the CMA-ES algorithm to explore a broader range of values for such parameters. In the hill-climbing algorithm, 'neighboring' is redefined as one-step changes in one or multiple parameters if the combination of the changed parameters has high Sobol indices, enabling the co-exploration of multiple parameters whose interactions have high Sobol indices.

VI. Experimental Results

A. Experimental Setup and Benchmarks

We test our Explorer with a VLSI design flow consisting of Cadence Genus and Innovus (version 20.1), on a platform with 2 Xeon Gold 6226R CPU processors. Our experiments are conducted on 5 representative IWLS2005 benchmark designs [26] (shown in TABLE I) under a 7-nm technology node [27]. Specifically, `vga` and `b19` are the two largest designs in the benchmark which can be synthesized under this technology node. According to the experience, 27 potential design performance-relative parameters are selected and tuned in our experiments, among which 4 are continuous parameters, and 23 are discrete parameters. Minimum clock period, total power and total area are chosen as the metrics to be optimized simultaneously. Examples of the parameters are shown in TABLE II. The purpose of some parameters is explained for supplementary:

- `syn_generic_effort` controls the effort of turning an elaborated design into a netlist of generic gates by doing high-level RTL and datapath optimizations.
- `syn_map_effort` controls the effort of mapping a design from generic gates to a technology library while optimizing for the best PPA.
- `tns_critical_range` assigns a minimum slack time for paths in a path group.
- `auto_partition` enables the process of disassembling designs into more manageable block sizes.
- `aspect ratio` assigns a specified ratio between the height and width of the floorplan.
- `target density` specifies a row density value, which equals $(std\,area + block/macro\,area) \div core\,area$.
- `place_detail_wire_length_opt_effort` specifies the effort for optimizing wire length by swapping cells.
- `place_global_cong_effort` controls the numerical iterations and ways of instance bloating of placement.
- `place_global_timing_effort` specifies the level of effort for timing-driven global placer.
- `routeWithLithoDriven` avoids lithography problems during routing by avoiding certain routing patterns.
- `droutePostRouteViaPillarEffort` specifies the effort level toward increasing the ratio of double-cut vias to single-cut vias concurrently with routing.

The following metrics are used to compare each parameter tuning method on all $five$ designs: hypervolume (HV), maximum performance improvement (MPI1), maximum power improvement (MPI2), and maximum area improvement (MAI). These metrics are good at assessing the ability of each method to explore the tool parameter spaces.

To compute the hypervolume, we use $(600, 1, 500)$, $(900, 1, 800)$, $(1200, 20, 110000)$, $(1500, 5, 30000)$, $(600, 100, 150000)$ as the reference points for the $five$ designs respectively based on the range of metrics. Hypervolume is calculated as:

$$HV\left(f^{ref}, \mathrm{X}\right) = \Lambda\left(\bigcup_{\mathrm{X}_n \in \mathrm{X}} \left[f_1\left(\mathrm{X}_n\right), f_1^{ref}\right] \times \cdots \times \left[f_m\left(\mathrm{X}_n\right), f_m^{ref}\right]\right), \tag{26}$$

where f^{ref} refers to the reference point, and X refers to the evaluated PPA metrics.

The runtime of the proposed flow consists of two parts: The algorithm's runtime and the evaluation's runtime. The overall runtime is calculated as follows:

$$T_{all} = n \cdot (t_{algo} + t_{eda}), \tag{27}$$

where n is the number of iterations, t_{algo} is the algorithm's runtime in each iteration, and t_{eda} is the runtime of evaluation using EDA tools in each iteration. t_{algo} is about 2 minutes while t_{eda} takes a few hours or more, according to the size of the design. Since $t_{algo} \ll t_{eda}$, we compared the performance of different methods by fixing the number of evaluation budgets in each method. The hyperparameter K, which indicates the number of parallel evaluations, is set to 5, meaning our flow consumes 5 evaluation budgets in one iteration.

To fairly and truly evaluate the performance state of our work, we compare it to the following state-of-the-art methods by exploring the parameter space of the selected designs using 50 iterations:

- DAC'19 [5]: A recommender system that is trained offline using data from previous designs and produces

TABLE III Comparisons between Explorer and SOTA works on the 4 benchmarks, with data in parentheses representing the ratios of these values to the values produced by Explorer.

Design	Metric	DAC'19 [5]	MLCAD'19 [11]	ASPDAC'20 [7]	ICCAD'21 [1]	Explorer [28]	Enhanced Explorer
8bit_risc_cpu	MPI1(%)	7.23	3.51	0.68	9.76	12.41	**14.47**
	MPI2(%)	2.34	0.94	3.62	1.05	3.87	**3.93**
	MAI(%)	0.37	0.59	0.15	0.82	0.85	**0.94**
	$HV(10^3)$	79.1(0.80)	73.9(0.75)	70.8(0.72)	85.1(0.86)	94.3(0.95)	**98.9**(1)
des3_area	MPI1(%)	12.69	17.33	3.33	15.92	18.20	**21.3**
	MPI2(%)	2.54	2.04	0.05	3.05	4.04	**4.72**
	MAI(%)	0.28	0.28	0	0	0.67	**0.80**
	$HV(10^3)$	46.3(0.92)	45.7(0.91)	38.9(0.78)	44.7(0.89)	48.9(0.98)	**50.1**(1)
b15	MPI1(%)	1.84	2.62	4.20	4.10	**5.46**	4.79
	MPI2(%)	2.56	2.87	2.69	3.46	3.60	**3.72**
	MAI(%)	**6.82**	6.51	6.03	5.94	6.61	**6.87**
	$HV(10^3)$	718.7(0.94)	733.7(0.96)	738.1(0.96)	728.6(0.95)	759.3(0.99)	**767.3**(1)
b19	MPI1(%)	1.30	2.76	3.01	3.89	4.58	**5.25**
	MPI2(%)	1.41	2.43	0.66	1.21	**2.90**	**2.90**
	MAI(%)	0.47	0.56	1.18	0.98	1.23	**1.45**
	$HV(10^3)$	349.5(0.79)	411.2(0.93)	368.7(0.83)	428.9(0.97)	435.3(0.98)	**444.3**(1)
vga	MPI1(%)	21.10	7.51	24.56	15.89	23.99	**27.21**
	MPI2(%)	1.53	0.65	6.60	5.45	7.69	**8.21**
	MAI(%)	0.15	0.73	2.20	3.66	**4.40**	**4.40**
	$HV(10^3)$	412.2(0.82)	446.7(0.89)	457.5(0.93)	450.9(0.90)	490.3(0.98)	**502.2**(1)
Average MPI1(%)		8.83	6.75	7.16	9.91	12.93	**14.60**
Average MPI2(%)		2.08	1.77	2.72	2.84	4.42	**4.70**
Average MAI(%)		1.62	1.73	1.91	2.28	2.75	**2.89**
Average $HV(10^3)$		321.2(0.86)	342.2(0.92)	334.8(0.90)	347.6(0.93)	365.62(0.98)	**372.6**(1)

TABLE IV Parameter sets outputted by each method for design `des3_area`, along with corresponding minimum clock period(ps), total power(mW) and total area(μm^2)

	syn_generic_effort	syn_map_effort	tns_critical_range	aspect ratio	place_detail_wire _length_opt_effort	place_global_ timing_effort	Minimum clock period	Power	Area
baseline	high	high	15	1	high	high	275.7	0.203	591.8
DAC'19 [5]	high	high	15	1	medium	high	267.3	0.201	591.8
MLCAD'19 [11]	high	medium	8	1	low	high	273.4	0.199	592.2
ASPDAC'20 [7]	high	medium	8	1	high	medium	266.5	0.200	591.8
ICCAD'21 [1]	high	high	15	0.8	high	low	287.4	0.199	590.1
Explorer [28]	high	medium	14	1.13	medium	high	250.9	0.198	588.0
Enhanced Explorer	high	medium	12	0.87	high	low	248.1	0.197	586.4

recommendations for the current design. For fair comparisons, the model is trained based on the 30 high reward parameter configurations from historical design records. For 20 iterations of online recommending, the model's recommendations are employed to guide the sampling.

- MLCAD'19 [11]: A multi-objective Bayesian optimization method that treats the EDA tools' parameter tuning as an expensive black-box optimization, and uses ordinary Gaussian progress regression model as the surrogate model. In our experiment, this method is implemented with the package GPyOpt [29].
- ASPDAC'20 [7]: This method analyzes the importance of different parameters from previous designs and divides the parameter space into different clusters for further sampling and refinement using XGBoost. In our experiment, 30 high reward parameter configurations are used for importance sampling, and 20 iterations are used for refinement.
- ICCAD'21 [1]: A method that uses an ant colony engine to optimize the parameters in each stage of the EDA flow, and uses a cooperative co-evolutionary controller to harmonize the optimization in different stages.

The "Explorer" and "Enhanced Explorer" denote the methods proposed in [28] and this work, respectively.

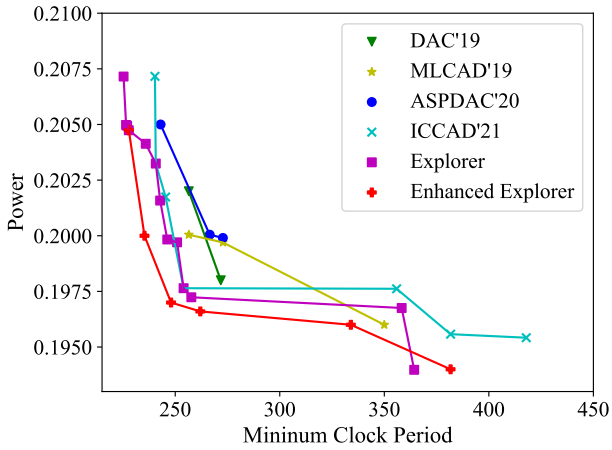
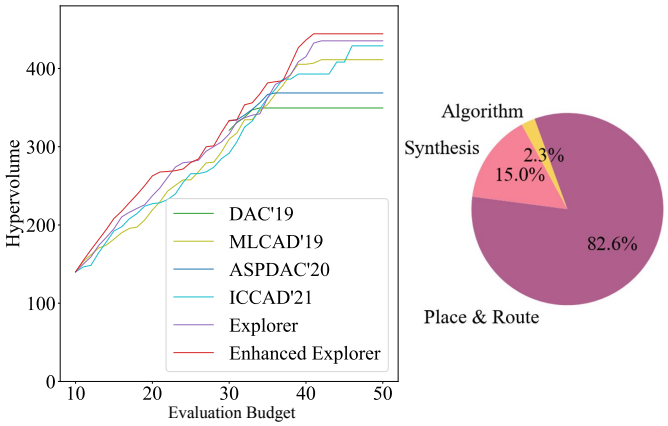


Fig. 7 The visualizations of Pareto frontiers in two QoR metric spaces of `des3_area`.

B. Comparisons Against SOTA Works

TABLE III shows the comparison results of our work with existing methods. The baseline setting is a parameter set with all parameters set to the highest effort. It can be seen



(a) (b)

Fig. 8 (a) The visualization of different methods' hypervolume changes with iterations in design $b19$. DAC'19 [5] and ASPDAC'20 [7] are given 30 high award data for initialization, while other methods share the same 10 randomly picked data to initialize. (b) The comparison of the average time of different steps in one iteration in design $b19$.

that our "Enhanced Explorer" averagely behaves the best on MPI1, MPI2, and MAI. Moreover, the "Enhanced Explorer" achieves better hypervolume than existing methods on all benchmarks, which indicates that the proposed method can find better tool parameter configurations when considering multiple objectives concurrently.

To improve the readability and reproducibility, some results from the Pareto frontiers of parameter configurations `des3_area` given by each method are listed in TABLE IV along with partially associated parameters.

The visualization of the Pareto frontiers in power versus minimum clock period predicted by the methods on benchmark `des3_area` is displayed in Fig. 7. The purple square points and red diamond points refer to the method in [28] and our enhanced method, respectively. It visually shows that the parameter configurations we find dominate those found by other algorithms. Fig. 8 shows the increase of hypervolume of each method and different steps' runtime in design $b19$. The red curve rises faster than others in hypervolume, which means our method outperforms others.

Fig. 9 shows the Sobol indices which were analytically calculated in the final iteration of the "Enhanced Explorer" flow. The importance values are derived analytically based on data from the evaluated configurations. The analysis reveals that specific individual parameters and their interactions exhibit significant influence across various designs. For example, the parameter `syn_generic_effort` consistently plays a crucial role in optimizing the area, reflecting its relevance in logic synthesis optimizations. Similarly, `leakage_power_effort` is shown to be pivotal for power reduction, further confirming its direct influence on power optimization outcomes. For performance improvements, however, the results suggest that no single parameter holds high importance universally across all designs. A notable observation is that most important parameters and their interactions related to area and power optimizations are concentrated within the logic synthesis stage. This is intuitive, as the selection of standard cells and synthesis efforts play a primary role in determining area and power consumption. In contrast, performance optimization proves more complex, as it is influenced by parameters spanning multiple stages, including logic synthesis, placement, and routing. The Sobol-based analysis further corroborates that only a few key parameters and interactions dominate the impact on PPA outcomes. This insight highlights one of the reasons behind the effectiveness of our enhanced explorer: by focusing on finding and optimizing this small set of influential parameters and interactions, the modeling of PPA is simplified, leading to faster convergence and better PPA results. Thus, this analysis validates the approach of exploring and concentrating on high-impact regions in iterative optimization to enhance overall design efficiency and quality.

TABLE V Ablation Study on GP model accuracy

Metrics	Vanilla GP	Ours w/o additive	Ours w/o attention	Ours
$\mathbf{R}^2$	0.389	0.408	0.457	0.473
MAE (10^{-3})	6.51	6.31	5.54	5.11

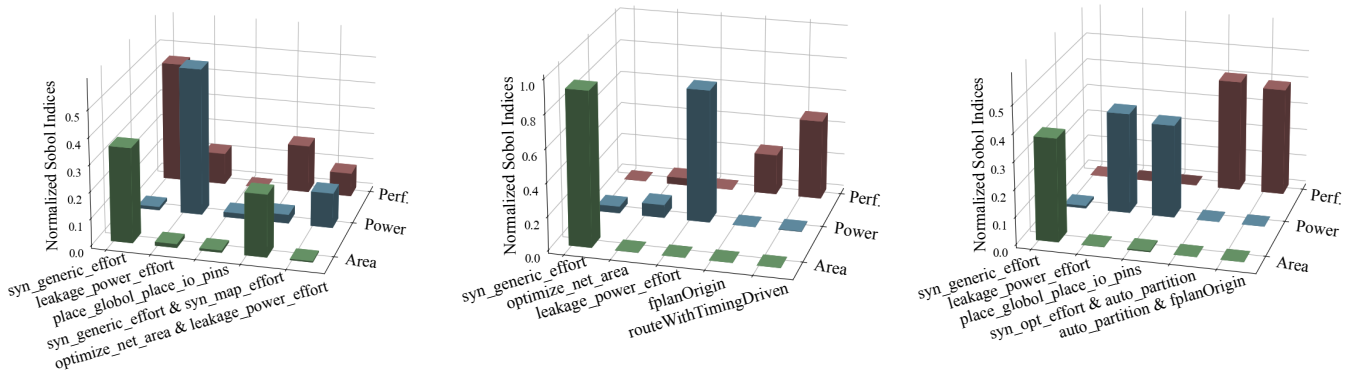
C. Ablation Study

To further analyze the effectiveness of our proposed method, we conduct an ablation study by comparing the accuracy of four different versions of the Gaussian process regression model:

- Vanilla GP: A standard Gaussian Process model, with all inputs considered continuous.
- Ours w/o additive: A standard GP model with proposed attention mechanism for data processing.
- Ours w/o attention: A non-standard GP model with enhanced additive kernel, with continuous and discrete parameters differentiated.
- Ours: GP model used in our enhanced explorer, with attention mechanism and enhanced additive kernel.

The GP model with higher accuracy will output more accurate means and variances, leading to more accurate EHVI scores and more efficient design space exploration. We conduct the experiments on the `des3_area` design's power prediction, randomly selecting the same 50 training data and 50 test data for each method. The experiment is repeated 50 times. The results are averaged over all runs. As the metrics of prediction accuracy, we use $\mathbf{R}^2$ (coefficient of determination), which measures the proportion of variance explained by the model, ranging from 0 to 1 where higher values indicate better fit, and **MAE** (Mean Absolute Error), which directly reflects the average prediction deviation.

The results in TABLE V demonstrate that our method outperforms all ablated versions, highlighting the contribution of both the additive kernel and the attention mechanism in improving performance.



(a) (b) (c)

Fig. 9 Important tool parameters or parameter interactions' Sobol indices calculated in the final iteration of "Enhanced Explorer" on design (a) `des3_area`, (b) `b15`, and (c) `vga`.

VII. Discussion

EDA tool parameter tuning is a complex and challenging task due to the large number of parameters and their interactions within the design flow. These interactions often lead to non-intuitive outcomes, making it insufficient to consider individual parameters in isolation. This combinatorial nature of parameter interactions expands the design space exponentially, making exhaustive exploration infeasible. To address this issue, our proposed framework incorporates dynamic parameter screening through the use of Sobol indices. By quantifying the influence of individual parameters and their interactions on the QoR metrics, the algorithm can prioritize high-impact parameters and interactions while progressively refining the search space. This design ensures scalability and efficiency, enabling effective exploration of larger design spaces.

Through our observations across multiple designs, we found that certain parameters within the logic synthesis stage, such as `syn_generic_effort` and `leakage_power_effort`, consistently exhibit a significant impact on the QoR metrics. For instance, these parameters demonstrated high importance across different benchmarks, highlighting their critical role in the synthesis process. However, beyond a few key parameters, the majority of parameters show varying degrees of importance depending on the specific design and context.

Although constructing a large-scale dataset covering diverse designs and parameter configurations could offer deeper insights into these interactions, it remains a costly and time-consuming endeavor. Moreover, the closed-source nature of most EDA tools further complicates the process, as it is often unclear how specific parameter settings influence the internal optimization algorithms. For instance, whether the high/medium/low settings of `syn_generic_effort` correspond to adjustments in iteration counts, changes in heuristic strategies, or entirely different internal algorithms remains largely unknown. Additionally, EDA tool updates may alter optimization mechanisms without modifying the user-visible parameter interface, making static parameter analysis impractical in the long term.

Our findings suggest that the improvement of our hybrid Bayesian optimization approach stems from its ability to model the hybrid space and focus on a small yet highly influential subset of parameters and their interactions. By simplifying the modeling of PPA trade-offs and accelerating convergence, our framework demonstrates a direction for improving design space exploration in modern IC design workflows.

VIII. Conclusion & Future Work

In this paper, for the first time, we have proposed an attention mechanism-based EDA tool parameter explorer surrogated with a hybrid space Gaussian process model. The explorer navigates the weights of important parameters through the attention mechanism and constructs a hybrid space multi-objective Bayesian optimization process that is more closely aligned with real-world scenarios. Additionally, the kernel of the model is enhanced to be more accurate and flexible, while improving explainability. Parallel computing is used to accelerate computation by evaluating multiple points simultaneously. Experimental results on 5 designs under an advanced 7nm technology node have demonstrated the effectiveness of the proposed framework.

While our proposed framework has demonstrated improvement in EDA tool parameter optimization, several directions remain open for future exploration. Different QoR metrics exhibit varying levels of sensitivity to tool parameters across different stages of the EDA flow. For instance, in the logic synthesis stage, key parameters affecting area and power primarily involve synthesis mapping effort and power optimization strategies. This suggests the need for domain knowledge-guided customized optimization strategies, where different QoR objectives are optimized with tailored parameter tuning processes at each design stage. Such metric-sensitive optimization strategies can reduce ineffective searches, accelerate convergence, and improve overall optimization efficiency.

The current optimization framework treats all QoR objectives equally, primarily relying on Pareto front exploration to balance multiple metrics. However, backend engineers often need to consider many primary and secondary metrics beyond

PPA, such as IR drop, thermal distribution, DRC/ERC errors, and so on. As the number of optimization objectives increases, simply searching for the Pareto front may become insufficient. Moreover, in practical design flows, certain metrics require strict optimization while others only need to satisfy predefined constraints, calling for a more nuanced treatment than exploring the Pareto frontier.

Additionally, most existing EDA parameter-tuning methods focus almost exclusively on PPA, neglecting critical design-rule compliance. A parameter configuration may achieve excellent PPA yet incur numerous DRC/ERC violations, rendering it infeasible for tape-out and subsequent validation. Therefore, future works should augment the evaluation framework to incorporate violation metrics, such as DRC/ERC counts, either as penalty terms within the objective function or as hard constraints. This extension will ensure that optimized parameter sets not only enhance PPA but also maintain full compliance with design rules and manufacturing requirements.

The rapid advancement of large language models (LLMs) in EDA applications suggests a potential new avenue for LLM-driven parameter pre-filtering. Recent work, such as ChatEDA [30], has demonstrated the effectiveness of LLMs in assisting backend flow scripting. Building on this concept, LLMs could be employed to suggest an initial subset of promising parameter configurations, effectively reducing the dimensionality of the search space. This could be achieved by encoding prior design experiences and expert heuristics into prompt-based interactions or fine-tuned models, guiding Bayesian optimization towards more relevant regions of the design space from the outset.

REFERENCES

[1] R. Liang, J. Jung, H. Xiang, L. Reddy, A. Lvov, J. Hu, and G.-J. Nam, "FlowTuner: A Multi-Stage EDA Flow Tuner Exploiting Parameter Knowledge Transfer," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2021, pp. 1–9.

[2] J. Jung, A. B. Kahng, S. Kim, and R. Varadarajan, "METRICS2. 1 and Flow Tuning in the IEEE CEDA Robust Design Flow and OpenROAD ICCAD Special Session Paper," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2021, pp. 1–9.

[3] E. Ustun, S. Xiang, J. Gui, C. Yu, and Z. Zhang, "LAMDA: Learning-Assisted Multi-stage Autotuning for FPGA Design Closure," in *IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2019, pp. 74–77.

[4] C. G. Tianqi Chen, "XGBoost: A scalable tree boosting system," in *ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016, pp. 785–794.

[5] J. Kwon, M. M. Ziegler, and L. P. Carloni, "A Learning-Based Recommender System for Autotuning Design Flows of Industrial High-Performance Processors," in *ACM/IEEE Design Automation Conference (DAC)*, 2019, pp. 1–6.

[6] A. Agnesina, K. Chang, and S. K. Lim, "VLSI Placement Parameter Optimization using deep Reinforcement Learning," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2020, pp. 1–9.

[7] Z. Xie, G.-Q. Fang, Y.-H. Huang, H. Ren, Y. Zhang, B. Khailany, S.-Y. Fang, J. Hu, Y. Chen, and E. C. Barboza, "FIST: A Feature-Importance Sampling and Tree-Based Method for Automatic Design Flow Parameter Tuning," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 2020, pp. 19–25.

[8] H. Geng, Q. Xu, T.-Y. Ho, and B. Yu, "PPATuner: pareto-driven tool parameter auto-tuning in physical design via gaussian process transfer learning," in *ACM/IEEE Design Automation Conference (DAC)*, 2022, pp. 1237–1242.

[9] S. Zhang, W. Lyu, F. Yang, C. Yan, D. Zhou, and X. Zeng, "Bayesian optimization approach for analog circuit synthesis using neural network," in *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*, 2019, pp. 1463–1468.

[10] 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 *ACM/IEEE Design Automation Conference (DAC)*, 2019, pp. 1–6.

[11] Y. Ma, Z. Yu, and B. Yu, "CAD tool Design Space Exploration via Bayesian Optimization," 2019, pp. 1–6.

[12] H. Geng, T. Chen, Y. Ma, B. Zhu, and B. Yu, "PTPT: Physical Design Tool Parameter Tuning via Multi-objective Bayesian Optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 42, no. 1, pp. 178–189, 2022.

[13] D. K. Duvenaud, H. Nickisch, and C. Rasmussen, "Additive gaussian processes," *Advances in neural information processing systems*, vol. 24, 2011.

[14] J. Snoek, K. Swersky, R. Zemel, and R. P. Adams, "Input warping for bayesian optimization of non-stationary functions," in *International Conference on Machine Learning (ICML)*, 2014.

[15] E. Snelson, C. E. Rasmussen, and Z. Ghahramani, "Warped gaussian processes," in *Conference on Neural Information Processing Systems (NIPS)*, 2003.

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

[17] R. I. Kondor and J. Lafferty, "Diffusion kernels on graphs and other discrete structures," in *International Conference on Machine Learning (ICML)*, vol. 2002, 2002, pp. 315–322.

[18] C. Oh, J. Tomczak, E. Gavves, and M. Welling, "Combinatorial bayesian optimization using the graph cartesian product," *Conference on Neural Information Processing Systems (NIPS)*, vol. 32, 2019.

[19] K. Yang, M. Emmerich, A. Deutz, and T. Bäck, "Multi-objective bayesian global optimization using expected hypervolume improvement gradient," *Swarm and evolutionary computation*, vol. 44, pp. 945–956, 2019.

[20] D. Ginsbourger, C. Helbert, and L. Carraro, "Discrete mixtures of kernels for kriging-based optimization," *Quality and Reliability Engineering International*, vol. 24, no. 6, pp. 681–691, 2008.

[21] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola, *Global sensitivity analysis: the primer*. John Wiley & Sons, 2008.

[22] L. St, S. Wold *et al.*, "Analysis of variance (anova)," *Chemometrics and intelligent laboratory systems*, vol. 6, no. 4, pp. 259–272, 1989.

[23] G. Hooker, "Generalized functional anova diagnostics for high-dimensional functions of dependent variables," *Journal of computational and graphical statistics*, vol. 16, no. 3, pp. 709–732, 2007.

[24] I. M. Sobol', "On sensitivity estimation for nonlinear mathematical models," *Matematicheskoe modelirovanie*, vol. 2, no. 1, pp. 112–118, 1990.

[25] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es)," *Evolutionary computation*, vol. 11, no. 1, pp. 1–18, 2003.

[26] C. Albrecht, "IWLS 2005 benchmarks," in *International Workshop for Logic Synthesis (IWLS): http://www.iwls.org*, 2005.

[27] L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy, and G. Yeric, "ASAP7: A 7-nm finFET predictive process design kit," *Microelectronics Journal*, vol. 53, pp. 105–115, 2016.

[28] D. Luo, Q. Sun, Q. Xu, T. Chen, and H. Geng, "Attention-based eda tool parameter explorer: From hybrid parameters to multi-qor metrics," in *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2024, pp. 1–6.

[29] N. Knudde, J. van der Herten, T. Dhaene, and I. Couckuyt, "GPflowOpt: A Bayesian optimization library using TensorFlow," *arXiv preprint arXiv:1711.03845*, 2017.

[30] H. Wu, Z. He, X. Zhang, X. Yao, S. Zheng, H. Zheng, and B. Yu, "Chateda: A large language model powered autonomous agent for eda," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2024.

Donger Luo is currently a Ph.D. student at the School of Information Science and Technology, ShanghaiTech University, under the supervision of Prof. Hao Geng, since Fall 2022. Previously, he received his B.S. from ShanghaiTech University. His research interests include EDA tool parameter tuning and microarchitecture design space exploration.

Qi Sun (M'23) is now a ZJU100 Young Professor at the College of Integrated Circuits, Zhejiang University. He received his Ph.D. degree from the Chinese University of Hong Kong and worked as a postdoc researcher at Cornell University. His research interests include AI for EDA, LLM for DTCO, computational lithography, etc. He has published over 50 top-tier papers, and has been recognized with two ICCAD Best Paper Awards (2021 & 2024), a Best Paper Award Nomination of DATE, etc.

Peng Xu is currently a Ph.D. student at the Department of Computer Science and Engineering, The Chinese University of Hong Kong (CUHK), under the supervision of Prof. Bei Yu, since Fall 2022. Previously, he received his B.S. from Central South University in 2019 and his M.S. from Harbin Institute of Technology(Shenzhen) in 2022. His research interests include machine learning for analog physical design and optimization in EDA problems.

Su Zheng received his B.Eng. degree from Fudan University in 2019, and M.S. degree from Fudan University in 2022. He is currently pursuing his Ph.D. degree at the Department of Computer Science and Engineering, the Chinese University of Hong Kong (CUHK), under the supervision of Prof. Bei YU and Prof. Martin D.F. Wong. His research interest is to solve critical problems in electronic design automation (EDA) with advanced artificial intelligence (AI) methods.

Qi Xu (Member, IEEE) received his Ph.D. degree in electronic science and technology from University of Science and Technology of China (USTC), Hefei, China, in 2018. He is currently an Associate Professor in the School of Microelectronics, USTC. His research interests include physical design automation, machine learning in EDA, and design for reliability for 3D integrated circuits.

Tinghuan Chen (M'21) received his B.Eng. and M.Eng. degrees in electronics engineering from Southeast University in 2014 and 2017, and Ph.D. degree in Computer Science and Engineering from The Chinese University of Hong Kong in 2021. He is currently an Assistant Professor in the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen. His research interests include machine learning for EDA and deep learning accelerators.

Bei Yu (M'15-SM'22) received the Ph.D. degree from The University of Texas at Austin in 2014. He is currently a Professor in the Department of Computer Science and Engineering, The Chinese University of Hong Kong. He has served as TPC Chair of ACM/IEEE Workshop on Machine Learning for CAD, and in many journal editorial boards and conference committees. He received eleven Best Paper Awards from ICCAD 2024 & 2021 & 2013, IEEE TSM 2022, DATE 2022, ASPDAC 2021 & 2012, ICTAI 2019, Integration, the VLSI Journal in 2018, ISPD 2017, SPIE Advanced Lithography Conference 2016, six ICCAD/ISPD contest awards, IEEE CEDA Ernest S. Kuh Early Career Award in 2021, DAC Under-40 Innovator Award in 2024, and Hong Kong RGC Research Fellowship Scheme (RFS) Award in 2024.

Hao Geng is currently an Assistant Professor with the School of Information Science and Technology, ShanghaiTech University, Shanghai, China. Prior to that, he received his Ph.D. degree from the Department of Computer Science and Engineering, The Chinese University of Hong Kong in 2021. His research interests include AI/LLM/VLM and optimization methods for DFM, computational lithography, chip design space exploration and other applications in VLSI CAD. He received several best paper award nominations from DAC'25, DAC'24, ASPDAC'2019.