# PTPT: Physical Design Tool Parameter Tuning via Multi-Objective Bayesian Optimization

Hao Geng, Tinghuan Chen, Yuzhe Ma, *Member, IEEE*, Binwu Zhu, and Bei Yu, *Member, IEEE*

***Abstract*—Physical design flow through associated electronic design automation (EDA) tools plays an imperative role in the advanced integrated circuit design. Mostly, the parameters fed into physical design tools are mainly manually picked based on the domain knowledge of the experts. Nevertheless, owing to the ever-shrinking scaling down of technology nodes and the complexity of the design space spanned by combinations of the parameters, even coupled with the time-consuming simulation process, such manual explorations for parameter configurations of physical design tools have become extremely laborious. There exist a few works in the field of design flow parameter tuning. However, very limited prior arts explore the complex correlations among multiple quality-of-result (QoR) metrics of interest (e.g., delay, power, and area) and explicitly optimize these goals simultaneously. To overcome these weaknesses and seek effective parameter settings of physical design tools, in this article, we propose a multi-objective Bayesian optimization (BO) framework with a multi-task Gaussian model as the surrogate model. An information gain-based acquisition function is adopted to sequentially choose candidates for tool simulation to efficiently approximate the Pareto-optimal parameter configurations. The experimental results on three industrial benchmarks under the 7-nm technology node demonstrate the superiority of the proposed framework compared to the cutting-edge works.**

***Index Terms*—Multi-objective Bayesian optimization (BO), multi-task Gaussian process, physical design, tool parameter tuning.**

## I. Introduction

OVER the past decades, both academia and industry have made great efforts on studying the physical design flow, the core of modern VLSI design. As a result, several physical design tools with dozens of sophisticated algorithms incorporated are developed to improve chip productivity and design quality and reduce time-to-market. Manual configuring the input parameters of physical design tools relies on expertise and domain knowledge, which may be time intensive and unrobust. Therefore, automatic parameter tuning for physical design tools is highly desirable.

Recently, a few works [1]–[6] have focused on physical design flow parameter tuning. For example, [1] is the first self-evolving and autonomous system for tuning the input parameters of logic and physical synthesis tools, while [2] further enhances the tuning framework proposed in [1] by integrating an adaptive online learning method. Ustun *et al.* [3] accelerated FPGA design closure by utilizing XGBoost [7] regressor-based machine learning tuning framework with design-specific features extracted from early stages of the design flow as guidance. Kwon *et al.* [4] used a tensor decomposition-based recommender system to tune parameters for macros. Agnesina *et al.* [5] optimized the placement parameters via a deep reinforcement learning framework fed with a mixture of handcrafted features covering the graph topology theory along and graph embeddings generated using graph neural networks. In [6], the active learning-based approach leveraging the feature importance sampling and XGBoost regressor is proposed for automatic parameter tuning. Although these pioneered parameter tuning frameworks have started the research explorations in this field and achieved notable progress, they have their limitations. Ustun *et al.* [3] only considered the single quality-of-result (QoR) metric, while the deep reinforcement learning framework utilized in [5] is hard to train and requires a large amount of training data. Neither Kwon *et al.* [4] nor Xie *et al.* [6] explicitly explored the correlations among QoR metrics to be optimized, which may lead to performance degradation.

It is worth mentioning that the parameter tuning issue often involves optimizing multiple QoR metrics simultaneously. However, these QoR metrics (e.g., delay, power, and area) are usually in conflict or coupled. The tradeoffs naturally need to be considered. What is more, "white-box" and "black-box" optimization problems emerge. For "white-box" optimization problems, the objective functions have explicit formulations, which can be directly addressed by numerous methods, such as evolutionary strategies [8] and gradient-based optimization methods [9]. Nevertheless, there are no explicit functions for optimized objectives in black-box problems. Parameter tuning of physical design tools is required to combat the black-box optimization. To a certain extent, the physical design tool parameter tuning issue, which can be casted as parameter space exploration, is analogous to the design space exploration (DSE) problem [10]. Some arts like [11] exploit the state-of-the-art single-objective Bayesian optimization (BO) framework with a scalarization trick (i.e., the trick of a weighted-sum cost function which is also

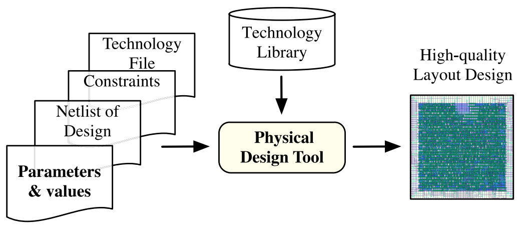
Manuscript received 5 July 2021; revised 17 October 2021; accepted 2 December 2021. Date of publication 15 April 2022; date of current version 22 December 2022. This work was supported in part by the Hisilicon Technologies Company, AI Chip Center for Emerging Smart Systems (ACCESS), Hong Kong, and in part by the Research Grants Council of Hong Kong under Project CUHK14209420. This article was recommended by Associate Editor L. Behjat. *(Corresponding author: Bei Yu.)*

Hao Geng is with the School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China.

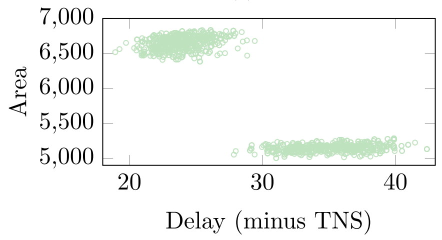
Tinghuan Chen, Binwu Zhu, and Bei Yu are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: byu@cse.cuhk.edu.hk).

Yuzhe Ma is with the Microelectronics Thrust, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou 511453, China.

Digital Object Identifier 10.1109/TCAD.2022.3167858



(a)



(b)

Fig. 1. Visualizations of the working flow of (a) physical design tool and (b) QoR metric space (area versus minus total negative slack) generated by invoking physical design tool 1440 times in a manual parameter tuning manner. In the QoR metric space, each light green dot represents an output QoR metric value tuple of a physical design tool, which is associated with a specific parameter configuration. The units for area and delay are *ns* and um$^2$, respectively.

exploited in [1], [2], and [6]) to handle multiple objectives. In these works, scalar weights of multi-objective functions are either sampled from a uniform distribution or decided based on the user's preference to construct a single-objective-like function. However, they are not really designed to seek near Pareto-optimal solutions to real QoR metrics (e.g., area versus delay, power versus delay, and area versus delay versus power). Pareto-driven works, such as [12], try to classify the input points based on the learned models into three classes: 1) Pareto optimal; 2) non-Pareto optimal; and 3) unclassified. In each iteration, they select the candidate input for evaluation toward the goal of minimizing the size of the uncertain set. Although the flow provides theoretical guarantees, it is only applicable to input space with a finite dataset of discrete points. Finally, by virtue of a different search process for optimum solutions, the multi-objective BO framework [13] is intuitively compatible with the black-box, multi-objective problem setting of physical design tool parameter tuning.

Given a design (e.g., a multiply accumulate (MAC) design under the advanced 7-nm technology node in our scenario) and PDK files [seen in Fig. 1(a)], how to automatically and efficiently acquire high-quality (near Pareto optimal) parameter configurations of the physical design tool is a critical issue. We try to visualize the challenges existing in parameter tuning in Fig. 1(b). The first challenge is that the final performance after physical synthesis could vary significantly under delicate changes of sensitive parameters. We can see in Fig. 1(b), the design quality variance in QoR metric space with changing seven physical synthesis tool parameters based on human intervention. Especially, manually tuning the parameter noted as "maxDensity" leads to the two clusters in QoR metric space. Even worse, QoR metrics are realistically correlated, which affects the tuning process. In Fig. 1(b), most of the dots in QoR metrics show either unsatisfied area quality with low delay or a small area value with a high delay. Additionally, the parameter design space is huge [thousands of dots in the QoR metric space in Fig. 1(b)] and will expand exponentially with more constraints or parameters are taken into account, which makes an efficient flow with as few evaluations as possible in high demand. To address the aforementioned issues, we develop a multi-task Gaussian process (MGP)-based multi-objective BO flow for efficient physical design tool parameter tuning. MGP regressor is leveraged as the surrogate model in the proposed BO framework, which captures the correlations among the tool parameters and QoR metrics. Compatible with the multi-objective optimization problem, the information gain-based acquisition function is developed to guide the exploration process to find the superior parameter settings. Our main contributions are summarized as follows.

1) An MGP is built to learn interobjective dependencies.
2) A multi-objective BO framework with an MGP as its surrogate model is investigated to attempt to tune physical design tool parameters.
3) The proposed optimization framework finds better Pareto frontiers on three benchmarks under the advanced 7-nm technology node with less expense on physical design tool runs.

The remainder of this article is organized as follows. Section II introduces some prior knowledge about multi-objective optimization and then gives the problem formulation. Section III describes the proposed multi-objective BO framework, while Section IV discusses the developed parameter tuning flow. Section V shows the experimental results. Section VI presents an analysis of the differences between SOTA works and ours, followed by a conclusion and future work discussion in Section VII.

## II. Preliminaries

In this section, the background of the multi-objective optimization is depicted, and then with descriptions of two evaluation metrics, we give the problem formulation.

### A. Multi-Objective Optimization

Assume a multi-objective optimization problem has a set of feasible solutions $\mathcal{X} \in \mathbb{R}^P$. There are $N$ objective functions, $f_1(\cdot), \ldots, f_N(\cdot)$, which map an input $\boldsymbol{x}$ to corresponding results $f_1(\boldsymbol{x}), \ldots, f_N(\boldsymbol{x})$ forming an $N$-dimensional result vector $\boldsymbol{f}(\boldsymbol{x})$. A result vector $\boldsymbol{f}(\boldsymbol{u})$ is said to dominate another result vector $\boldsymbol{f}(\boldsymbol{v})$ if $\boldsymbol{f}(\boldsymbol{u})$ is at least as good as $\boldsymbol{f}(\boldsymbol{v})$ in all the objectives, namely, $f_i(\boldsymbol{u}) \leq f_i(\boldsymbol{v})$ $\forall i \in [1, N]$ if all the objectives are to be minimized. Hence, we say that a solution $\boldsymbol{x}$ is Pareto optimal if it is not dominated by other solutions in the feasible solution set $\mathcal{X}$. A sketch of multi-objective optimization is exemplified in Fig. 2.

In our context of physical design tool parameter tuning, a feasible solution vector $\boldsymbol{x}$ is the feature representation of encoded, normalized, and concatenated physical tool parameters, which satisfies the predetermined constraints, while a Pareto-optimal design is where none of the QoR metrics, such as the area, power, and delay, can be optimized without

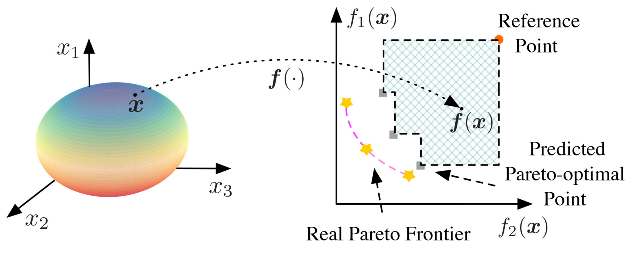


Fig. 2. Example of a bi-objective (two correlated QoR metrics) optimization problem with a tridimensional parameter configuration space. The left subfigure is the tridimensional parameter space, while the right subfigure displays the associated bi-objective space. The middle curve connecting the two parts refers to a parameter-to-QoR mapping $f(\cdot)$. Suppose that the parameter configuration features are the physical design parameters, such as frequency, max fanout, and max density, while QoR metrics are area and delay. Each data point in the parameter space is a simply encoded, normalized and concatenated feature vector of the three design parameters. The Pareto frontier, golden stars with dash carnation line in the right sketch, features optimized delay and area.

worsening at least one of the rest. The Pareto set contains all the Pareto-optimal solutions.

## B. Problem Formulation

*Definition 1 (Hypervolume):* This metric reflects the volume fenced by the Pareto frontier and a reference point in the objective space. It measures how well distributed the points are on the Pareto frontier approximation.

In the right subfigure of Fig. 2, the area filled with grids is an example of the hypervolume of a predicted Pareto-optimal set in a bi-objective space. The hypervolume error for a Pareto set approximation $\hat{\mathcal{P}}$ is defined

$$e = \frac{H(\mathcal{P}) - H\left(\hat{\mathcal{P}}\right)}{H(\mathcal{P})} \tag{1}$$

where $\mathcal{P}$ is the golden Pareto-optimal set, and $H(\mathcal{P})$ is the ground truth of hypervolume. If a solution set $\mathcal{P}'$ is better than another set $\mathcal{P}''$, $H(\mathcal{P}')$ is greater than $H(\mathcal{P}'')$ in our scenario.

*Definition 2 [Average Distance From Reference Set (ADRS)]:* Given a reference Pareto-optimal set $\mathcal{A} = \{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots | \boldsymbol{a} = ({m_1}^a, {m_2}^a, \ldots, {m_N}^a)\}$ and an approximated Pareto-optimal set $\mathcal{P} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, | \boldsymbol{p} = ({m_1}^p, {m_2}^p, \ldots, {m_N}^p)\}$ in the $N$-objective optimization problem

$$\text{ADRS}(\mathcal{A}, \mathcal{P}) = \frac{1}{(|\mathcal{A}|)} \sum_{\boldsymbol{a} \in \mathcal{A}} \min_{\boldsymbol{p} \in \mathcal{P}} \delta(\boldsymbol{a}, \boldsymbol{p}) \tag{2}$$

where $\delta(\boldsymbol{a}, \boldsymbol{p}) = \max\{|({m_1}^p - {m_1}^a)/{m_1}^a|, \ldots, |({m_N}^p - {m_N}^a)/{m_N}^a|\}$. Assume in a QoR metric space (e.g., area versus delay), there are several points in the reference Pareto-optimal set and we assign index numbers to them. Hence, we can write down the mathematical definition of the reference set as $\mathcal{A} = \{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots\}$. Then, for instance, the meaning of $m_1^a$ is the first coordinate (i.e., the area value) of the $\boldsymbol{a}_1$ in the reference Pareto-optimal set.

ADRS is used to quantify how close a set of nondominated points is from the Pareto frontier in the objective space. The smaller ADRS value, the closer the approximate set $\mathcal{P}$ is to the reference set $\mathcal{A}$.

With the above knowledge, our problem can be formulated as follows.

*Problem 1 (Automatic Parameter Tuning for Physical Design Tool):* Given the boundary of parameters of a physical design tool, the objective of physical design tool parameter tuning is to automatically search the Pareto-optimal parameter configurations, which bring about the high design quality concerning multiple QoR metrics, such as delay versus power/area and delay versus power versus area.

## III. Multi-Objective Bayesian Optimization Method

As aforementioned, the ultimate goal of physical design parameter tuning is to simultaneously achieve timing closure and smallest reachable area and lowest acceptable power. Quite often the QoR metrics are coupled or conflicted, which is not ideal for traditional tuning flows. For example, reducing the supply voltage can effectively cut down the dynamic power consumption. However, it leads to an increase in gate delays. To tackle such negatively correlated issue, we propose a tuning flow which is based on an MGP surrogated multi-objective BO method. The optimization method explores the correlations among QoR metrics and attempts to explore the best tradeoffs among them. The two main components of the proposed optimization method, i.e., the MGP surrogate model and information gain-based acquisition function, are well-customized for our case. In the following, we will first introduce the two vital keys and then the whole optimization method.

## A. Surrogate Model: Multi-Task Gaussian Process

MGP models are prevalently harnessed to couple related objectives or functions for a joint regression [14], [15]. This coupling is achieved by designing a structured covariance function, yielding a prior on objectives to be regressed. More importantly, MGP tries to learn a kernel involving inter-task dependencies based solely on the task identities and the observed data for each task. This kind of property is naturally compatible with our optimization methodology. We can utilize MGP as the surrogate model to jointly predict the multiple QoR metric [e.g., power–performance–area (PPA)] values with respect to tool parameters as inputs. Note that the "task" means regression on one QoR metric value in our case.

In the same way as single-task GPs, multi-task GPs are mainly specified by their kernels which are usually assumed to be zero-mean functions. For developing valid covariance functions, we adopt a linear model of coregionalization (LMC) where the outputs are expressed as linear combinations of independent random processes. As aforementioned, MGP attempts to build a multi-output function $\boldsymbol{f}(\boldsymbol{x}) : \mathbb{R}^p \rightarrow \mathbb{R}^N$ with a feature vector $\boldsymbol{x} \in \mathbb{R}^p$ as input and $N$ the number of QoR metrics. Note that in our work, with simple encoding and normalization tricks, the physical design tool parameters are simply encoded and concatenated as $\boldsymbol{x}$. In the LMC assumption, the $d$th element of output vector (in our case, the output vector contains estimated QoR metric values, such as area, delay, and power given the input tool parameter configuration

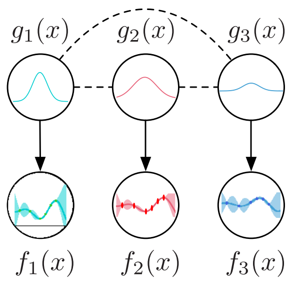


Fig. 3. Visualization for a tritask Gaussian process model with a scalar input and $Q = 1$. The dashed lines indicate the correlations. We can imagine this model is built for an area versus power versus delay joint regression problem with a parameter, `max_transition`, as input $x$.

$\boldsymbol{x}$), i.e., $f_d(\boldsymbol{x})$, can be represented as

$$f_d(\boldsymbol{x}) = \sum_{q=1}^{Q} a_{d,q} g_q(\boldsymbol{x}) \tag{3}$$

where each latent function $g_q(\boldsymbol{x})$ is considered to follow an independent Gaussian prior with zero mean and covariance $\mathrm{cov}\,[g_q(\boldsymbol{x}), g_{q'}(\boldsymbol{x}')] = k_q(\boldsymbol{x}, \boldsymbol{x}')$ if $q = q'$, and $a_{d,q}$ is a scalar number. Specifically, when we set $Q$ equal to 1 and $\sum a_{d,q} = 1$, (3) is degraded to $f_d(\boldsymbol{x}) = g_d(\boldsymbol{x})$ with a little abuse of notations. For a better understanding, a visualization for MGP is exemplified in Fig. 3.

Obviously, the processes $\{g_q(\boldsymbol{x})\}_{q=1}^{Q}$ are independent if $q \neq q'$. We can derive the cross covariance between any two functions $f_d(\boldsymbol{x})$ and $f_{d'}(\boldsymbol{x})$ like

$$\begin{aligned}\mathrm{cov}\big[f_d(\boldsymbol{x}), f_{d'}(\boldsymbol{x}')\big] &= \sum_{q=1}^{Q} a_{d,q} a_{d',q} k_q(\boldsymbol{x}, \boldsymbol{x}') \\ &= \sum_{q=1}^{Q} b_{d,d'}^{q} k_q(\boldsymbol{x}, \boldsymbol{x}').\end{aligned} \tag{4}$$

$(\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}'))_{d,d'}$ is exploited to denote $\mathrm{cov}\,[f_d(\boldsymbol{x}), f_{d'}(\boldsymbol{x}')]$, which indicates the similarity or covariance across $d$- and $d'$th tasks at $\boldsymbol{x}$ and $\boldsymbol{x}'$, respectively. According to the expression in (4), the kernel for MGP is shown as follows:

$$\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{q=1}^{Q} \boldsymbol{B}_q(d, d') k_q(\boldsymbol{x}, \boldsymbol{x}') \tag{5}$$

where $\boldsymbol{B}_q \in \mathbb{R}^{N \times N}$ is a so-called coregionalization matrix, and the coefficient $b_{d,d'}^{q}$ is the element of $\boldsymbol{B}_q$. $\boldsymbol{B}_q$ is positive semidefinite matrix which specifies the intertask similarities. The proof is supplemented as follows. Assuming $a_{d,q} a_{d',q} = b_{d,d'}^{q}$ and column vector $\boldsymbol{A}_q = (a_{1,q}, \ldots, a_{d,q}, \ldots, a_{N,q}) \in \mathbb{R}^{N \times 1}$, we can write down $\boldsymbol{B}_q = \boldsymbol{A}_q \boldsymbol{A}_q^{\top}$. According to the definition of the positive semidefinite matrix, $\boldsymbol{B}_q$ is positive semidefinite if every nonzero real column vector $z$ satisfies $z^{\top} \boldsymbol{B}_q z \geq 0$. Because $z^{\top} \boldsymbol{B}_q z = z^{\top} \boldsymbol{A}_q \boldsymbol{A}_q^{\top} z = (\boldsymbol{A}_q^{\top} z)^{\top} (\boldsymbol{A}_q^{\top} z)$, which is the square of $(\boldsymbol{A}_q^{\top} z)$, it cannot be less than 0. By definition, the coregionalization matrix $\boldsymbol{B}_q \in \mathbb{R}^{N \times N}$ fulfills the requirement of the positive semidefiniteness. The above linear expression of outputs represents the covariance function as the sum of the products of two covariance functions. More specifically, $\boldsymbol{B}_q$ models the dependence between the outputs, which is independent to the input parameter vector $\boldsymbol{x}$. $k_q(\boldsymbol{x}, \boldsymbol{x}')$ discovers the parameter dependence, independently of the ultimate output QoR metrics $\boldsymbol{f}(\boldsymbol{x})$.

For further reduction, a reasonable assumption that $b_{d,d'}^{q} = k_{d,d'}^{g} b_q$ for a suitable scalar coefficient $k_{d,d'}^{g}$ can be made. With substituting this assumption for $b_{d,d'}^{q}$, (4) can be rewritten as

$$\mathrm{cov}\big[f_d(\boldsymbol{x}), f_{d'}(\boldsymbol{x}')\big] = k_{d,d'}^{g} k^{x}(\boldsymbol{x}, \boldsymbol{x}') \tag{6}$$

where $k^{x}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{q=1}^{Q} b_q k_q(\boldsymbol{x}, \boldsymbol{x}')$. Finally, on the back of independent GP priors over the latent functions $g_q(\boldsymbol{x})$, our kernel matrix corresponding to a dataset $\boldsymbol{X}$ (stacking the tool parameter configurations as rows) takes the form in

$$\begin{aligned}\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) &= \boldsymbol{K}^{g} \otimes \boldsymbol{K}^{x}(\boldsymbol{X}, \boldsymbol{X}) \\ f_d &\sim \mathcal{N}\left(\sum_{q=1}^{Q} a_{d,q} g_q(\boldsymbol{x}), \sigma_d^2\right)\end{aligned} \tag{7}$$

where $\otimes$ refers to Kronecker product. Equation (7) decouples the correlation between task (estimating the QoR metric values) similarities and input (tool parameter configurations) similarities.

In the same spirit of standard GP for the mean and variance predictions, inference in the MGP model can be calculated. Given $M$ training points (tool parameter configurations) with concatenated golden QoR metric values $\boldsymbol{y} \in \mathbb{R}^{MN}$ and a new parameter configuration $\boldsymbol{x}_*$, The closed-form expressions of mean and uncertainty for task $d$ at $\boldsymbol{x}_*$ shown as (9) and (10) are acquired by using the first-order optimality condition on the marginal likelihood function $L$ [in (8)] of the MGP

$$\begin{aligned}L = &-\frac{M}{2}\log\big|\boldsymbol{K}^{g}\big| - \frac{N}{2}\log\big|\boldsymbol{K}^{x}\big| \\ &- \frac{1}{2}\mathrm{tr}\Big[\big(\boldsymbol{K}^{g}\big)^{-1}\boldsymbol{F}^{\top}\big(\boldsymbol{K}^{x}\big)^{-1}\boldsymbol{F}\Big] - \frac{M}{2}\sum_{l-1}^{N}\log\sigma_l^2 \\ &- \frac{1}{2}\mathrm{tr}\Big[(\boldsymbol{Y}-\boldsymbol{F})\boldsymbol{D}^{-1}(\boldsymbol{Y}-\boldsymbol{F})^{\top}\Big] - \frac{MN}{2}\log 2\pi\end{aligned} \tag{8}$$

$$\bar{\boldsymbol{f}}_d(\boldsymbol{x}_*) = \big(\boldsymbol{k}_d^{g} \otimes \boldsymbol{k}_*^{x}\big)^{\top} \boldsymbol{\Sigma}^{-1} \boldsymbol{y} \tag{9}$$

$$\bar{\mathrm{var}}\big[\bar{\boldsymbol{f}}_d(\boldsymbol{x}_*)\big] = k_{d,d}^{g} k^{x}(\boldsymbol{x}_*, \boldsymbol{x}_*) - \big(\boldsymbol{k}_d^{g} \otimes \boldsymbol{k}_*^{x}\big)^{\top} \boldsymbol{\Sigma}^{-1} \big(\boldsymbol{k}_d^{g} \otimes \boldsymbol{k}_*^{x}\big) \tag{10}$$

where $\boldsymbol{\Sigma} \in \mathbb{R}^{MN \times MN} := (\boldsymbol{K}^{g} \otimes \boldsymbol{K}^{x} + \boldsymbol{D} \otimes \boldsymbol{I})$ with $\boldsymbol{D}$ an $N \times N$ diagonal matrix in which the element in $(d, d)$ is $\sigma_d^2$. $\boldsymbol{K}^{x}$ indicates the matrix of covariances between all pairs of training points. $\boldsymbol{K}^{g}$ which is treated as a kind of hyperparameter of the MGP describes the task similarities, and it can be obtained by maximizing likelihood estimation. Furthermore, $\boldsymbol{k}_d^{g}$ denotes the $d$th column of $\boldsymbol{K}^{g}$, and $\boldsymbol{k}_*^{x}$ refers to the vector of covariances between the test point $\boldsymbol{x}_*$ and the other training points. Until now, we have built the surrogate model and obtained the predictions on QoR metrics, such as area, delay, and power for further calculations of the acquisition function.

## *B. Information Gain-Based Acquisition Function*

Given the parameter configuration space $\mathcal{X}$ (i.e., the boundaries of parameters) of a physical design tool and current training dataset $\mathcal{D}$, we attempt to maximize the information gain about the predicted Pareto frontier $\mathcal{Y}^*$, namely, expected reduction in entropy over $\mathcal{Y}^*$. Consequently, we can write down an information gain-based acquisition function $I(\boldsymbol{x})$ as showed in

$$\begin{aligned} I(\boldsymbol{x}) &= I(\{\boldsymbol{x},\boldsymbol{y}\}, \mathcal{Y}^* \mid \mathcal{D}) \\ &= \mathrm{H}(\mathcal{Y}^* \mid \mathcal{D}) - \mathbb{E}_{\boldsymbol{y}}\left[\mathrm{H}(\mathcal{Y}^* \mid \mathcal{D} \cup \{\boldsymbol{x},\boldsymbol{y}\})\right] \end{aligned} \tag{11}$$

where $\boldsymbol{x}$ is the parameter configuration to be selected and $\boldsymbol{y}$ is the corresponding predicted the QoR values. By using the symmetric property of mutual information, we can rewrite (11) as follows:

$$I(\boldsymbol{x}) = \mathrm{H}(\boldsymbol{y} \mid \mathcal{D}, \boldsymbol{x}) - \mathbb{E}_{\mathcal{Y}^*}\left[\mathrm{H}(\boldsymbol{y} \mid \mathcal{D}, \boldsymbol{x}, \mathcal{Y}^*)\right]. \tag{12}$$

Equation (12) tries to minimize the uncertainty estimation of Pareto frontier approximation $\mathcal{Y}^*$ after searching the next candidate $\boldsymbol{x}$ for design tool evaluation. The first term of (12) is straightforward to compute. In fact, it is simply the entropy of the predictive distribution $\mathrm{p}(\boldsymbol{y} \mid \mathcal{D}, \boldsymbol{x})$, which is an $N$-dimensional Gaussian distribution $\mathcal{N}(\boldsymbol{y} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$. Applying the substitution trick in a normalization fashion, we can obtain a simplified expression of the first term as

$$\mathrm{H}(\boldsymbol{y} \mid \mathcal{D}, \boldsymbol{x}) = \frac{N}{2}(\ln 2\pi + 1) + \frac{1}{2}\ln|\boldsymbol{\Sigma}|. \tag{13}$$

The main part of second term in (12) is the entropy of the predictive distribution conditioned on the Pareto frontier $\mathcal{Y}^*$. This expectation term can be approximated fast and effectively by

$$\mathbb{E}_{\mathcal{Y}^*}\left[H(\boldsymbol{y} \mid D, \boldsymbol{x}, \mathcal{Y}^*)\right] \simeq H(\boldsymbol{y} \mid D, \boldsymbol{x}, \mathcal{Y}^*). \tag{14}$$

In (14), we can observe that the computation involves the predicted Pareto frontier $\mathcal{Y}^*$, which means the Pareto frontier needs to be estimated. To calculate the Pareto frontier samples, a multi-objective optimization formulation should be first established. Analogous to the previous arts [13], [16], sample functions from the posterior MGP model via some kernel functions and then solve a multi-objective optimization over the $N$ sampled functions. This multi-objective optimization also allows us to capture the interactions between different objectives. In the process, the $N$ kernel-based linear functions are exploited in a ridge regression taste. The principle behind this execution is that the Gaussian Process is a Bayesian generalization of the ridge regression and can be explained in a weight space view [17]. The sample function is constructed as a finitely parameterized approximation which is shown in

$$\tilde{f}_i(\boldsymbol{x}) = \boldsymbol{\kappa}(\boldsymbol{x})^\top \boldsymbol{\mu} \tag{15}$$

where $\boldsymbol{\kappa}(\cdot)$ is some kind of kernel function such as Matern, radial basis function, and $\boldsymbol{\mu}$ is a random variable sampled from its corresponding posterior distribution conditioned on the dataset $\mathcal{D}$ containing all parameter configurations through tool evaluations. The reparameterization trick is exploited to compute $\boldsymbol{\mu}$.

**Algorithm 1** Computation Process for Information Gain-Based Acquisition Function in the $t$th Iteration

**Input**: the number of tasks $N$, the predictions from the multi-task GP model.

```
1: for $i \leftarrow 1$ to $N$ do
2:     Sample approximation function $\tilde{f}_i(\cdot)$ from each task
       output from the multi-task GP model;        ▷ Equation (15)
3: end for
4: Multi-objective optimizer (NSGA-II) optimizing over
   $\tilde{f}_1(\cdot), \tilde{f}_2(\cdot), \ldots, \tilde{f}_N(\cdot)$;        ▷ Equation (16)
5: Compute and maximize I($\boldsymbol{x}$);        ▷ Equation (17)
```

After formulating the multi-objective optimization over the $N$ sampled functions $\tilde{f}_1(\cdot), \tilde{f}_2(\cdot), \ldots, \tilde{f}_N(\cdot)$, a genetic algorithm-based solver, e.g., NSGA-II [8], is adopted to optimize the optimization problem

$$\mathcal{X}^* = \arg\min_{\boldsymbol{x}\in\mathcal{X}}\left(\tilde{f}_1(\boldsymbol{x}), \tilde{f}_2(\boldsymbol{x}), \ldots, \tilde{f}_N(\boldsymbol{x})\right) \tag{16}$$

where $\mathcal{X}^*$ is the associated Pareto-optimal set.

Until now, the Pareto-optimal set $\mathcal{X}_t^*$ with corresponding objective values $\mathcal{Y}^*$ is approximated. Next, an additional constraint is introduced. It can be proved that the value of each element of $\boldsymbol{y}$ in (14) is upper bounded by the maximum value of the corresponding element in sampled point on Pareto frontier $\mathcal{Y}^*$. Combining the boundedness property and the fact that each sampled objective function is modeled as a GP prior, we can model each component of $\boldsymbol{y}$ as a truncated Gaussian distribution. Ultimately, directly harnessing the closed-form solutions to moments of a truncated Gaussian distribution [18] and (13), we can get the approximation of acquisition function as shown in

$$I(\boldsymbol{x}) = \sum_{i=1}^{N}\left[\frac{\nu_i(\boldsymbol{x})\phi(\nu_i(\boldsymbol{x}))}{2F(\nu_i(\boldsymbol{x}))} - \ln F(\nu_i(\boldsymbol{x}))\right] \tag{17}$$

where $\phi$ and $F$ stand for the probability density function and the cumulative density function of a standard Gaussian distribution, respectively. $\nu_i(\boldsymbol{x})$ equals$([y_i^* - \mu_i(\boldsymbol{x})]/[\sigma_i(\boldsymbol{x})])$ with $y_i^*$ the maximum value among the sampled points on predicted Pareto frontier for the $i$th QoR metric. Then, we can maximize (17) via some Quasi-Newton optimizers like limited-memory BFGS. The computation process for information gain-based acquisition function is briefly concluded in Algorithm 1.

## *C. Multi-Objective Bayesian Optimization Method*

After details of two main components, we summarize the proposed multi-objective BO method in Algorithm 2. In the initialization stage of Algorithm 2, the surrogate model captures the priors about the unknown objective functions and offers predictions on posterior distributions (line 1). Per iteration, the acquisition function first exploits the predicted posterior distributions to search for the candidate points for the tool query (lines 3 and 4). After evaluation, the candidate points with golden values will in turn help calibrate the

**Algorithm 2** Multi-Objective BO Method

**Input**: the parameter configuration space $\mathcal{X}$ (i.e. the boundaries of parameters) of a certain physical design tool, initial dataset $\mathcal{D}_0$ and the number of maximum iterations $T$.
**Output**: the set of predicted Pareto-optimal parameter configurations $\tilde{\mathcal{P}}$ over $\mathcal{D}$.

```
1: Initialization: Initial the multi-task GP model with initial data set $\mathcal{D}_0$;
2: while not convergence or $t < T$ do
3:     $x_t \leftarrow \arg\min_{x \in \mathcal{X}} \mathrm{I}(x)$;        ▷ Algorithm 1
4:     Enquiry the physical design tool to acquire the golden values $y_t$ (area/power/delay values) of $x_t$;
5:     Update the training dataset $\mathcal{D}_{t-1}$ with $(x_t, y_t)$: $\mathcal{D}_t = \mathcal{D}_{t-1} \cup (x_t, y_t)$;
6:     Fine tune the multi-task GP model with $\mathcal{D}_t$;
7:     $t \leftarrow t + 1$;
8: end while
```

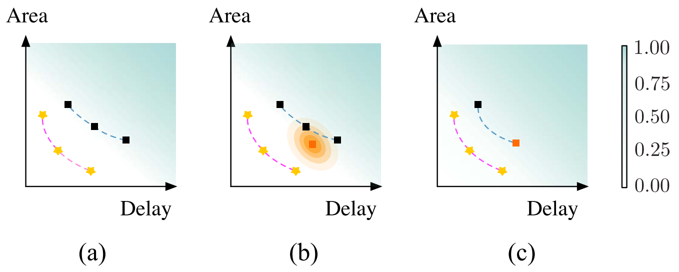


Fig. 4. Visualization for the proposed BO method optimizing multiple QoR metrics of interest (i.e., area versus delay). The golden stars with dash carnation lines refer to the golden Pareto frontier among area and delay. The black squares with blue-dashed lines denote the current predicted Pareto frontier based on observed tool parameter configurations and corresponding performance values. The cyan heat map reflects the probability of points in the bi-objective space dominated by others. With a darker color, the probability gets higher. (a) New parameter configuration for tool evaluation has not been selected yet. (b) Method chooses a new parameter configuration with associated area versus delay values near one predicted Pareto frontier point. The nested circles reflect the distribution of the output function at new data point. (c) Predicted Pareto frontier is updated with the new data point and the dark-colored area in the heat map expands, which means that the uncertainty about the Pareto frontier approximation decreases.

MGP model (lines 5 and 6). The optimization process performs in this manner iteratively until converges. Finally, the Pareto-optimal parameter configurations are obtained.

For better comprehension, we sketch some visualizations in Fig. 4 to illustrate the working principles for the proposed BO method.

## IV. Developed Physical Design Tool Parameter Tuning Flow

We visualize the global view of the proposed tuning flow in Fig. 5, where the arrows mark the dataflow. Our proposed BO framework delivers the selected parameter configurations for tool evaluation. The tool estimates the corresponding QoR metric values given selected parameter configurations with certain design netlist, technology files, and associate libraries. Then, the training dataset stores the new data point and promptly calibrates the multi-task GP model. With more accurate predictions from the multi-task GP model, the acquisition

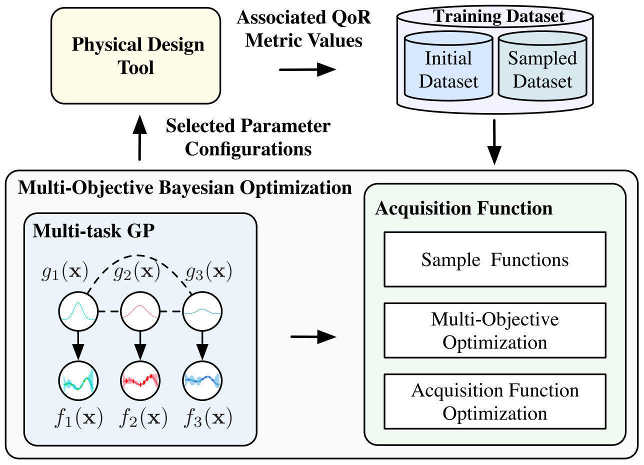


Fig. 5. Workflow of the proposed tuning flow.

function continues seeking the next parameter configuration candidate.

Practically, our approach also supports batch trials. For the part of the physical design tool, we have several software licenses so that the parallels trials are supported when invoking the physical design tool (line 4 of Algorithm 2). The generation of benchmarks is based on parallel invoking. When it turns to the computation of information gain-based acquisition function, the optimization process has already been accelerated by parallel computing. If multiple optimal solutions to (17) are found, with several licenses, the enquiring for golden QoR metric values of these parameter configurations can be simultaneously run.

## V. Experimental Results

### A. Experimental Setup and Benchmarks

The implementation of our framework is in `Python` with the `GPflow` [15] library and the certain physical design tool we used in this article is Cadence Innovus Implementation System (version 16.2) [19]. We test it on a platform with a Xeon Silver 4114 CPU processor. To evaluate the performance state of our framework, we compare it with the state-of-the-art methods [4], [6], [11], [12], [20] by exploring the parameter space of the physical design tool on three industrial benchmarks.

For a better understanding, the statistics of parameters are summarized in Table I. The data type can be either floating or integer, which depends on the specific parameters. Note that all the benchmarks consist of hundreds of different input parameter configurations of the physical design tool with associated QoR metrics values. These golden values of QoR metrics are obtained by invoking the physical design tool fed with different-sized industrial MAC designs. Under the 7-nm technology node, Benchmark1 and Benchmark2 are generated by a MAC design having 20k of cells, while data points in Benchmark3 are attained by feeding the synthesis tool with a much larger industrial MAC design (about 67k cells). In Benchmark1 and Benchmark2, 5000 data points represent parameter spaces that are built upon the distinct combinations

TABLE I
STATISTICS OF PARAMETERS OF THE PHYSICAL DESIGN TOOL ON THREE BENCHMARKS. NOTE THAT "-" IN THE TABLE MEANS THE PARAMETER IS NOT CONSIDERED IN THIS BENCHMARK. SOME DESCRIPTIONS OF PARAMETERS ARE INTRODUCED AS SUPPLEMENTARY. FOR EXAMPLE, `flowEffort` CONFIGURES THE FLOW TO GIVE TRADEOFF BETWEEN THE BEST QUALITY RESULT AND BEST TURNAROUND TIME, AND `place_global_uniform_density` ENABLES EVEN CELL DISTRIBUTION FOR DESIGNS WITH LESS THAN 70% UTILIZATION, AND `place_global_cong_effort` SPECIFIES THE EFFORT LEVEL OF RELIEVING CONGESTION, AND `place_global_max_density` CONTROLS THE MAXIMUM DENSITY OF LOCAL BINS DURING GLOBAL PLACEMENT, WHILE MAX_LENGTH BELONGS TO DRV RULE PARAMETERS INCLUDING MAX_CAPACITANCE/MAX_TRANSITION/MAX_FANOUT, AND MAX_DENSITY DEFINES THE MAXIMUM VALUE FOR DENSITY (AREA UTILIZATION)

| Parameters | Benchmark1 | | Benchmark2 | | Benchmark3 | |
|---|---|---|---|---|---|---|
| | Min | Max | Min | Max | Min | Max |
| `freq` | 950 | 1050 | 1000 | 1300 | - | - |
| `place_rcfactor` | - | - | - | - | 1.00 | 1.30 |
| `place_uncertainty` | 50 | 200 | 20 | 100 | - | - |
| `flowEffort` | standard | extreme | standard | extreme | standard | extreme |
| `place_global_timing_effort` | - | - | - | - | medium | high |
| `place_global_clock_power_driven` | - | - | - | - | FALSE | TRUE |
| `place_global_uniform_density` | FALSE | TRUE | FALSE | TRUE | - | - |
| `place_global_cong_effort` | AUTO | HIGH | AUTO | HIGH | - | - |
| `place_global_max_density` | 0.65 | 0.90 | 0.65 | 0.90 | - | - |
| `max_Length` | 160 | 310 | 160 | 300 | 250 | 350 |
| `max_Density` | 0.65 | 0.90 | 0.65 | 0.90 | 0.50 | 1. 00 |
| `max_transition` | 0.19 | 0.34 | 0.10 | 0.35 | - | - |
| `max_capacitance` | 0.08 | 0.13 | 0.08 | 0.20 | 0.05 | 0.15 |
| `max_fanout` | 25 | 50 | 25 | 50 | 25 | 39 |
| `max_AllowedDelay` | 0.00 | 0.25 | 0.00 | 0.25 | 0.00 | 0.12 |

of 12 physical design tool parameters with different parameter bounds and settings. Meanwhile, in Benchmark3, 727 data points with nine tool parameters are selected to mimic the parameter space.

To further elaborate on the generation of benchmarks, we exemplify the benchmark generated by the small MAC design with 20k cells. Based on the IC designer's experience (or preference), we select 12 vital parameters of the physical design tool to be tuned to optimize design quality after the complete synthesis and physical design flow. To collect data points (namely, configurations with their golden QoR metric values), actual physical design runs with different synthesis parameter configurations on the fed design are extensively conducted to construct the benchmark. Additionally, these parameter configurations are chosen by the latin hypercube searching (LHS) to represent the whole parameter space. Notice that the LHS resembles the uniform random in a sense that the uniform random number is drawn within each equal-space interval. To put it another way, the LHS covers the parameter space more evenly in a fashion similar to the quasirandom. Hence, the data points found by the searching method can represent the parameter space to some extent. The reason for determining the parameters and performing LHS is two-fold. One is to prune the searching space and obtain the customized searching space, and the other is to accelerate the optimization process. Until now, we have obtained the offline benchmark, and after sorting and comparing, the golden QoR metric values of the Pareto frontier are achieved. Here, to avoid ambiguous and divergent understandings, we define "the golden values" as the bests can be found in the benchmark. In the same fashion, we get the rest benchmarks and golden QoR metric values.

In the experiments, our work is compared with other five cutting-edge works [4], [6], [11], [12], [20]. For the previous works, such as [11], [12], and [20], we have obtained the source code from the authors, while [4] and [6] are reimplemented by ourselves. To offer a clear clue, we write down some important settings and several vital hyperparameter values of these methods in the area versus power versus delay case of Benchmark3. For [20], 72 data points are used for training, and 25 points are finally predicted for acquiring the golden QoR metric values. The values of $\alpha$ and $\beta$ in the source code to combine the three QoR metrics are configured ranging from 0.01 to 1000 (just as $0.1, 0.2, \ldots, 1, 2, \ldots, 10, 20, \ldots, 100, 1000$ and so forth). In [12], 70 data points are used for initialization, and two points are selected for refinement before the unclassified set is empty, and 16 configurations are finally chosen for tool evaluation. The maximum iteration number is set to be 20, while the error tolerance with 0.007 for classifying Pareto points or non-Pareto ones is applied. Regarding [4], we apply a backpropagation-based manner to implement the CP decomposition with an iteration number 100. The maximum iteration number for the used single-layer perceptron is 800. To achieve good performance, 100 points in the benchmark are chosen for training. With respect to the reimplementation of [6], the budget is set to 70, and 50 data points are used for modelless sampling, and the cluster refinement threshold is set to ten iterations. Following the setting in [6], the tree depth of the XGBoost regressor is set to 3 and 10 for the initial and final stages, respectively. Considering [11], 50 data points are used for initialization, and 20 points (one per iteration) are selected during iterations. The Gaussian process upper confidence bound is harnessed as the acquisition function. It should be noticed that in [6] and [11], weight coefficients of QoR metrics are set equally during the comparison. When it turns to our work, the sizes for initialization and iterations are the same as in [11] (i.e., 50 and 20, respectively).

TABLE II
QUANTITATIVE COMPARISON OF THE PARETO FRONTIERS ON BENCHMARK1

| Multi-objective | ISLPED'17 [20] | | | TCAD'19 [12] | | | MLCAD'19 [11] | | | DAC'19 [4] | | | ASPDAC'20 [6] | | | Ours | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HV | ADRS | Runs | HV | ADRS | Runs | HV | ADRS | Runs | HV | ADRS | Runs | HV | ADRS | Runs | HV | ADRS | Runs |
| Area-Delay | 0.163 | 0.095 | 427 | 0.209 | 0.088 | 533 | 0.175 | 0.063 | 400 | 0.155 | 0.086 | 599 | 0.218 | 0.099 | 400 | 0.104 | 0.047 | 400 |
| Power-Delay | 0.212 | 0.073 | 423 | 0.232 | 0.065 | 512 | 0.188 | 0.064 | 400 | 0.245 | 0.080 | 587 | 0.182 | 0.059 | 400 | 0.140 | 0.039 | 400 |
| Area-Power-Delay | 0.299 | 0.098 | 451 | 0.218 | 0.063 | 475 | 0.181 | 0.068 | 400 | 0.232 | 0.079 | 566 | 0.202 | 0.067 | 400 | 0.118 | 0.045 | 400 |
| Average | 0.225 | 0.089 | 433.667 | 0.220 | 0.072 | 506.667 | 0.181 | 0.065 | **400** | 0.211 | 0.082 | 584 | 0.201 | 0.075 | **400** | **0.121** | **0.044** | **400** |
| Ratio | 1.862 | 2.031 | 1.084 | 1.820 | 1.649 | 1.267 | 1.503 | 1.489 | **1.000** | 1.741 | 1.856 | 1.460 | 1.658 | 1.705 | **1.000** | **1.000** | **1.000** | **1.000** |

TABLE III
QUANTITATIVE COMPARISON OF THE PARETO FRONTIERS ON BENCHMARK2

| Multi-objective | ISLPED'17 [20] | | | TCAD'19 [12] | | | MLCAD'19 [11] | | | DAC'19 [4] | | | ASPDAC'20 [6] | | | Ours | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HV | ADRS | Runs | HV | ADRS | Runs | HV | ADRS | Runs | HV | ADRS | Runs | HV | ADRS | Runs | HV | ADRS | Runs |
| Area-Delay | 0.164 | 0.121 | 566 | 0.142 | 0.068 | 509 | 0.122 | 0.061 | 400 | 0.129 | 0.075 | 627 | 0.133 | 0.063 | 400 | 0.053 | 0.032 | 400 |
| Power-Delay | 0.239 | 0.695 | 560 | 0.237 | 0.235 | 512 | 0.199 | 0.256 | 400 | 0.243 | 0.266 | 596 | 0.190 | 0.178 | 400 | 0.099 | 0.100 | 400 |
| Area-Power-Delay | 0.185 | 0.134 | 479 | 0.185 | 0.064 | 503 | 0.160 | 0.058 | 400 | 0.214 | 0.100 | 577 | 0.195 | 0.086 | 400 | 0.098 | 0.055 | 400 |
| Average | 0.196 | 0.317 | 535 | 0.188 | 0.122 | 508 | 0.160 | 0.125 | **400** | 0.195 | 0.147 | 600 | 0.173 | 0.109 | **400** | **0.083** | **0.062** | **400** |
| Ratio | 2.352 | 5.080 | 1.338 | 2.256 | 1.963 | 1.270 | 1.924 | 2.005 | **1.000** | 2.353 | 2.371 | 1.500 | 2.080 | 1.758 | **1.000** | **1.000** | **1.000** | **1.000** |

TABLE IV
QUANTITATIVE COMPARISON OF THE PARETO FRONTIERS ON BENCHMARK3

| Multi-objective | ISLPED'17 [20] | | | TCAD'19 [12] | | | MLCAD'19 [11] | | | DAC'19 [4] | | | ASPDAC'20 [6] | | | Ours | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HV | ADRS | Runs | HV | ADRS | Runs | HV | ADRS | Runs | HV | ADRS | Runs | HV | ADRS | Runs | HV | ADRS | Runs |
| Area-Delay | 0.145 | 0.123 | 96 | 0.103 | 0.099 | 95 | 0.140 | 0.104 | 70 | 0.133 | 0.092 | 132 | 0.114 | 0.116 | 70 | 0.072 | 0.060 | 70 |
| Power-Delay | 0.139 | 0.110 | 94 | 0.113 | 0.092 | 93 | 0.121 | 0.094 | 70 | 0.127 | 0.077 | 125 | 0.122 | 0.119 | 70 | 0.090 | 0.073 | 70 |
| Area-Power-Delay | 0.170 | 0.210 | 97 | 0.107 | 0.084 | 88 | 0.099 | 0.074 | 70 | 0.105 | 0.103 | 137 | 0.140 | 0.086 | 70 | 0.071 | 0.065 | 70 |
| Average | 0.151 | 0.148 | 95.667 | 0.108 | 0.092 | 92 | 0.120 | 0.091 | **70** | 0.122 | 0.091 | 131.333 | 0.125 | 0.107 | **70** | **0.078** | **0.066** | **70** |
| Ratio | 1.948 | 2.237 | 1.367 | 1.386 | 1.389 | 1.314 | 1.545 | 1.374 | **1.000** | 1.567 | 1.374 | 1.876 | 1.614 | 1.621 | **1.000** | **1.000** | **1.000** | **1.000** |

## *B. Comparisons Against SOTA Works*

For a comprehensive comparison, the experiments are performed qualitatively and quantitatively. First, we estimate the performance of these methods in terms of the hypervolume error, the ADRS, and the number of tool runs, and then visually compare the quality of the Pareto frontiers found by those methodologies.

Tables II–IV depict the qualities of Pareto frontiers in quantification. Note that for the smaller MAC design, it costs about 3 h per tool run. For the larger MAC design utilized in Benchmark3 generation, each data point needs almost two days to go through the physical design flow. In contrast, the initialization and follow-up updating of all methods require far less time (e.g., usually a few minutes) than tool runs. On the back of that, we use the number of tool runs to calculate the optimization (or runtime) cost in lieu of the conventional running time for modeling (training and testing). For a better understanding, we also provide an example for exact runtime comparisons between modeling and tool runs by the end of this section.

Column "Multi-objective" lists three objective spaces to be explored: 1) area versus delay; 2) power versus delay; and 3) area versus power versus delay, whilst columns "HV," "ADRS," and "Runs" are the evaluation metrics referring to hypervolume error, the ADRS and the number of tool runs. Columns "ISLPED'17," "TCAD'19," "MLCAD'19," "DAC'19," "ASPDAC'20" and "Ours" represent the results acquired by a simple regression method (i.e., support vector machine) [20], a Pareto-driven active learning-based optimization framework [12], a single-objective BO framework [11] with the weighted-sum trick extending to multi-objective cases, a tensor decomposition and recommender system-based tuning framework [4], an active learning-based tuning approach leveraging the feature importance sampling and XGBoost regressor [6], and our proposed MGP-based multi-objective BO frameworks, respectively. For Benchmark1, it can be seen that, with less optimization expense, our algorithm evenly outperforms [20] with a 46.2% decrease on hypervolume error and a 51.0% drop on ADRS value, and contracts 45.0% hypervolume error and 63.6% ADRS value compared with [12]. On the other hand, our method is superior to [11] on this benchmark with a drop of 33.1% hypervolume error and almost half ADRS value, and reduces beyond 40.0% on both evaluation metrics compared with [4] and [6]. Considering Benchmark2, with fewer physical design tool runs, our algorithm averagely surpasses [20] with the average hypervolume error and the ADRS value of 0.083 and 0.062, while it has less 55.9% hypervolume error and fewer 49.2% ADRS value compared with [12]. Besides, our method behaves better than [11] by decreasing 48.1% hypervolume error and shrinking half ADRS value. With about 50.0% average reductions on hypervolume error and ADRS, our method shows a better performance against [4], [6]. When it turns to the performance of these methods on Benchmark3, our algorithm exceeds [20] by averagely decreasing 48.3% hypervolume error and 55.4% ADRS, and also excels [12] on both performance indicators, i.e., 27.8% less hypervolume error and 28.3% fewer ADRS value. Compared to [11], 35.0% hypervolume error and 27.5% ADRS are declined. Despite the three works, by at least 36.1% less hypervolume error and 27.5% fewer ADRS, the proposed methodology is better than [4], [6] The table data suggest that our method has the

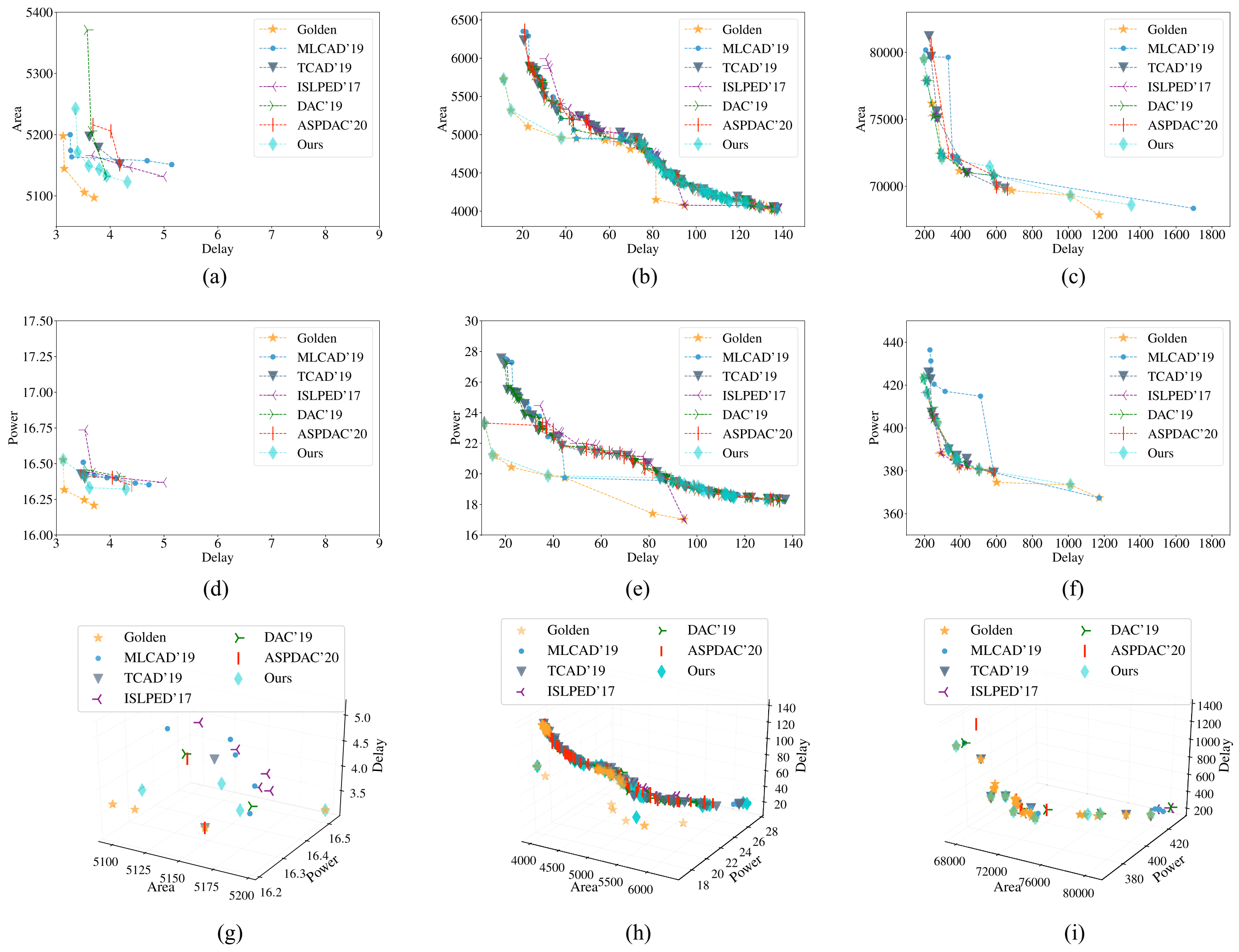


Fig. 6. Visualizations of Pareto frontiers on three industrial Benchmarks (best viewed in color and zoomed in): (a) Pareto frontiers: area versus delay on Benchmark1; (b) Pareto frontiers: area versus delay on Benchmark2; (c) Pareto frontiers: area versus delay on Benchmark3; (d) Pareto frontiers: power versus delay on Benchmark1; (e) Pareto frontiers: power versus delay on Benchmark2; (f) Pareto frontiers: power versus delay on Benchmark3; (g) Pareto frontiers: area versus power versus delay on Benchmark1; (h) Pareto frontiers: area versus power versus delay on Benchmark2; and (i) Pareto frontiers: area versus power versus delay on Benchmark3. The units for area, power, and delay are ns, mW, and um$^2$, respectively.

least optimization expense (or runtime overhead) as well. In a nutshell, the quantitative results in Tables II–IV illustrate the superiority of our method.

In addition, the visualizations of the Pareto frontiers in area versus delay space, power versus delay space, and area versus power versus delay space predicted by the methods on three benchmarks are displayed in Fig. 6. In Fig. 6, golden star dots represent the points on golden Pareto frontiers, and thin diamond dots in cyan refer to the predictions by our method, while other different markers with distinct colors stand for the compared frameworks. A glance at the graphs reveals that Pareto frontiers searched by our approach are much closer to the golden frontiers than the frontiers explored by other methods. Even some predictions by ours exactly match the golden results [e.g., see in Fig. 6(b) and (e)].

It is worth noticing that the performance of algorithms directly influences the searched Pareto frontiers, which results in the associated moving of Pareto frontier lines. As introduced in Section II, the applied two solution quality indicators (i.e., hypervolume error and ADRS value) capture the closeness of the solutions to the optimal set and the spread of the solutions across the QoR metric space. With smaller values of them, the searched Pareto frontier should be spread wider and also closer to the golden Pareto frontier. As is presented in previous result tables, our method behaves better than the other methods in terms of the two quality indicators. Accordingly, the corresponding Pareto frontiers are closer to the golden ones and spread widely across the QoR metric space. Briefly, Fig. 6 and Tables II–IV in combination demonstrate that our method predicts the better Pareto frontiers.

For a clear demonstration of time overhead, we exploit the area versus power versus delay case of Benchmark3 as an example. Note that the total runtime is composed of modeling time and the time cost on the physical design tool. It takes our method only 89.97 s to do modeling, while invoking of physical design tool costs approximately 14 days with 10 licenses. We sketch the pie chart in Fig. 7 for illustration. Compared to the huge time expenditure on the physical design

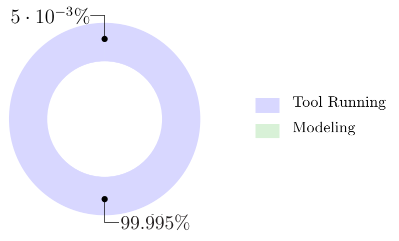


Fig. 7. Runtime overhead.

tool and the better performance on hypervolume error and ADRS, modeling time seems to be acceptable.

## VI. Related Work

In this section, we will conduct a comprehensive analysis on differences between SOTA works and ours which affects the performance (e.g., moving of Pareto lines). The main difference between ISLPED'17 [20] and our work is the searching paradigm. Our work uses a multi-objective BO framework that can search the Pareto solutions within a small number of runs of CAD tool, while [20] exploits the regression model integrated Pareto frontier exploration. In our work, the BO framework builds a surrogate model for the objective and quantifies the uncertainty in that surrogate using a Bayesian machine learning technique, Gaussian process regression, and then uses an acquisition function defined from this surrogate to decide where to sample iteratively. After several iterations, it makes final recommendations. In [20], the regression model will be calibrated to predict different best solutions which form the Pareto frontier. In the parameter tuning application, the BO framework is more suitable. Because this kind of method is very data efficient and particularly useful in situations where evaluations are costly [21].

Despite the differences in the searching flow, the correlation assumptions among multiple objectives also differ. For the ISLPED'17 [20] work, to explore the Pareto frontier in objective space, the authors exploit a linear summation idea that combines the multiple objectives with tradeoff coefficients. More specifically, they propose a joint output function as the regression model's output rather than using any single output. For example, when considering the delay versus area space, the objective function for the regression model (e.g., support vector regressor) used in [20] is $AD = \alpha \cdot \text{Area} + \text{Delay}$. However, the linear weighted-sum technique transforms the optimization problem into a single objective, which is not fully equivalent to the original multi-objective problem since the extra weighting coefficients depends on user's experience or preference [22]. The final solutions rely on these coefficients which cannot be easily and optimally chosen. Besides, oceans of methods (e.g., linear or quadratic way) can be applied to construct the weighted sum function, however, there is no explicit guideline to choose which form is the best for a given problem. Such a linear assumption is straightforward and describes the correlations superficially. What is worse, it works only for convex problem formulations, but synthesis-derived Pareto sets can be nonconvex [10]. This technique is also used in [6] and [11], which may contribute to their performance degradations. Regarding our work, we discover the correlations among multiple objectives without assuming linearity. Instead, we couple related objectives for a joint regression (i.e., via designing a structured covariance function in the multi-output Gaussian regressor) to model the correlations. The multi-output Gaussian regressor is not limited to sketching the linear correlations.

When it turns to the MLCAD'19 work [11], the state-of-the-art BO technique with the weighted and summed cost function is exploited to handle the multiple objective problems in the CAD tool (the Synopsis IC Compiler tool) parameter tuning. Multiple QoR metrics are scaled and then summed as a single metric utilized in the acquisition function (e.g., Gaussian process upper confidence bound). Although the single-objective BO framework is modified by the weighted and summed cost function, the defect that the weight-sum trick is not fully appropriate for handling the multi-objective problem still exists. Consequently, with the same size of the initial dataset and iteration number (the combination of these two is the total runs of the physical synthesis tool), our multi-objective Bayesian framework behaves much better than [11]. It would be a good proof of the superiority of the multi-objective approach.

The prior art, ASPDAC'20 [6], exploits feature impotence to guide the sampling and utilize ensemble boosting tree-based regressor as a learning model. Consequently, a good tradeoff between exploitation and exploration is achieved. Nevertheless, the weight-sum trick is also employed in this method. This kind of method is classical and commonly used, and yet the limitations emerge either. As mentioned, not like our method trying to model complex correlations among multiple QoR metrics, it works only on the simple and explicit relationships.

DAC'19 [4] work is inspired by the solution to the matrix completion issue in recommender systems and spans the problem to the high-dimension space via using tensor decomposition. A neural network is constructed to mimic the correlations among features and QoR metrics. This is a double-edged sword. Due to the overparameterized regime, neural networks are famous for their high accuracy but rely on more data than the Gaussian process. Besides, there is no iterative refinement but a one-time effort training procedure. On the contrary, our work harnesses the MGP and an iterative refinement framework.

Considering TCAD'19 [12], a simple Pareto-driven framework is utilized. The iterative refinement framework with Pareto-optimal classification rules is designed to combat the DSE issue. Unfortunately, the theoretical guarantees about optimality are related to the size of the design space.

## VII. Conclusion and Future Work

In this article, for the first time, we have proposed an effective parameter tuning flow for a certain physical design tool, which is built upon an information gain-based multi-objective BO framework surrogated with an MGP model. The optimization framework discovers the dependencies among multiple QoR metrics and explores the high-quality parameter configurations. The experimental results on three industrial benchmarks under advanced 7-nm technology node have demonstrated the efficacy and effectiveness of the proposed framework.

Finally, we would like to extend our discussion to future directions. One possible prospect is incorporating the customer's preference on QoR metrics. In practice, not all metrics have equal importance due to the different customers' specifications. Currently, there is no explicit weight mechanism incorporated in our proposed method but a pretty naive idea. Since the Pareto set acquired by the current multi-objective approach usually consists of several Pareto-optimal configurations, based on the associated QoR metric values, the customer can select the design to suit their appetite for objectives to some extent.

Some very recent works like [23] and [24] may enlighten our future studies. The main idea behind [23] and [24] is formulating an acquisition function to ensure that the subset of the Pareto front satisfies the preference-order constraints. However, some accurate prior knowledge about objective space and the preferred region (generally, a hyperbox) for Pareto-front solutions may be needed. Reducing the prior information and formulating the preference-order constraints into the acquisition function appear to be a promising way.

## REFERENCES

[1] M. M. Ziegler, H.-Y. Liu, G. Gristede, B. Owens, R. Nigaglioni, and L. P. Carloni, "A synthesis-parameter tuning system for autonomous design-space exploration," in *Proc. IEEE/ACM Design Autom. Test Eurpoe (DATE)*, 2016, pp. 1148–1151.

[2] M. M. Ziegler, H.-Y. Liu, and L. P. Carloni, "Scalable auto-tuning of synthesis parameters for optimizing high-performance processors," in *Proc. IEEE Int. Symp. Low Power Electron. Design (ISLPED)*, 2016, pp. 180–185.

[3] E. Ustun, S. Xiang, J. Gui, C. Yu, and Z. Zhang, "LAMDA: Learning-assisted multi-stage autotuning for FPGA design closure," in *Proc. IEEE Int. Symp. Field-Programmable Custom Comput. Mach. (FCCM)*, 2019, pp. 74–77.

[4] J. Kwon, M. M. Ziegler, and L. P. Carloni, "A learning-based recommender system for autotuning design flows of industrial high-performance processors," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, 2019, pp. 1–6.

[5] A. Agnesina, K. Chang, and S. K. Lim, "VLSI placement parameter optimization using deep reinforcement learning," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, 2020, pp. 1–9.

[6] Z. Xie *et al.*, "FIST: A feature-importance sampling and tree-based method for automatic design flow parameter tuning," in *Proc. IEEE/ACM Asia South Pacific Design Autom. Conf. (ASPDAC)*, 2020, pp. 19–25.

[7] C. G. Tianqi Chen, "XGBoost: A scalable tree boosting system," in *Proc. ACM Int. Conf. Knowl. Disc. Data Min. (KDD)*, 2016, pp. 785–794.

[8] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[9] J.-A. Désidéri, "Multiple-gradient descent algorithm (MGDA) for multiobjective optimizationAlgorithme de descente à gradients multiples pour l'optimisation multiobjectif," *Comptes Rendus Mathematique*, vol. 350, nos. 5–6, pp. 313–318, 2012.

[10] H.-Y. Liu, I. Diakonikolas, M. Petracca, and L. Carloni, "Supervised design space exploration by compositional approximation of Pareto sets," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, 2011, pp. 399–404.

[11] Y. Ma, Z. Yu, and B. Yu, "CAD tool design space exploration via Bayesian optimization," in *Proc. ACM/IEEE Workshop Mach. Learn. CAD (MLCAD)*, 2019, pp. 1–6.

[12] Y. Ma, S. Roy, J. Miao, J. Chen, and B. Yu, "Cross-layer optimization for high speed adders: A Pareto driven machine learning approach," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 12, pp. 2298–2311, Dec. 2019.

[13] D. Hernández-Lobato, J. Hernandez-Lobato, A. Shah, and R. Adams, "Predictive entropy search for multiobjective Bayesian optimization," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2016, pp. 1492–1501.

[14] E. V. Bonilla, K. Chai, and C. Williams, "Multi-task Gaussian process prediction," in *Proc. Annu. Conf. Neural Inf. Process. Syst. (NeurIPS)*, vol. 20, 2007, pp. 153–160.

[15] M. Van der Wilk, V. Dutordoir, S. John, A. Artemev, V. Adam, and J. Hensman, "A framework for interdomain and multioutput Gaussian processes," 2020, *arXiv:2003.01115*.

[16] J. M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani, "Predictive entropy search for efficient global optimization of black-box functions," in *Proc. Annu. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2014, pp. 918–926.

[17] C. K. Williams and C. E. Rasmussen, *Gaussian Processes for Machine Learning*, vol. 2. Cambridge, MA, USA: MIT Press, 2006.

[18] J. Burkardt, *The Truncated Normal Distribution*, Florida State Univ., 2014, pp. 1–35.

[19] "Cadence Innovus Implementation System." [Online]. Available: http://cadence.com (Accessed: Oct. 10, 2021).

[20] S. Roy, Y. Ma, J. Miao, and B. Yu, "A learning bridge from architectural synthesis to physical design for exploring power efficient high-performance adders," in *Proc. IEEE Int. Symp. Low Power Electron. Design (ISLPED)*, 2017, pp. 1–6.

[21] 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," *Proc. IEEE*, vol. 104, no. 1, pp. 148–175, Jan. 2016.

[22] X.-S. Yang, *Nature-Inspired Optimization Algorithms*. Waltham, MA, USA: Elsevier Sci. Publ., 2014.

[23] B. Paria, K. Kandasamy, and B. Póczos, "A flexible framework for multiobjective Bayesian optimization using random scalarizations," in *Proc. Conf. Uncertainty Artif. Intell. (UAI)*, vol. 115, 2020, pp. 766–776.

[24] M. Abdolshah, A. Shilton, S. Rana, S. Gupta, and S. Venkatesh, "Multi-objective Bayesian optimisation with preferences over objectives," in *Proc. Annu. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2019, pp. 12235–12245.

**Hao Geng** received the Ph.D. degree from the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, in 2021.

He is currently an Assistant Professor with the School of Information Science and Technology, ShanghaiTech University, Shanghai, China. His research interests include design space exploration, machine learning, deep learning, and optimization methods with applications in VLSI CAD.

Dr. Geng received one Best Paper Award Nomination from ASPDAC'2019.

**Tinghuan Chen** received the Ph.D. degree from the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, in 2021.

He is a Postdoctoral Fellow with the Department of Computer Science and Engineering, The Chinese University of Hong Kong. His research interests include machine learning in analog/mixed-signal VLSI design-for-reliability and cyber–physical systems.

**Yuzhe Ma** (Member, IEEE) received the B.E. degree from the Department of Microelectronics, Sun Yat-sen University, Guangzhou, China, in 2016, and the Ph.D. degree from the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong, in 2020.

He is currently an Assistant Professor with Microelectronics Thrust, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China. His research interests include agile VLSI design methodologies, machine learning-aided VLSI design, and hardware-friendly machine learning.

Dr. Ma received the Best Paper Awards from ICCAD 2021, ASPDAC 2021, and ICTAI 2019 and the Best Paper Award Nomination from ASPDAC 2019.

**Binwu Zhu** received the B.E. degree in information engineering from Zhejiang University, Hangzhou, China, in 2020. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

His research interest is machine learning in EDA.

**Bei Yu** (Member, IEEE) received the Ph.D. degree from The University of Texas at Austin, Austin, TX, USA, in 2014.

He is currently an Associate Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

Dr. Yu received nine Best Paper Awards from DATE 2022, ICCAD 2021 and 2013, ASPDAC 2021 and 2012, ICTAI 2019, *Integration*, the VLSI Journal in 2018, ISPD 2017, and SPIE Advanced Lithography Conference 2016, and six ICCAD/ISPD Contest Awards. He has served as a TPC Chair for the ACM/IEEE Workshop on Machine Learning for CAD, and many journal editorial boards and conference committees. He is the Editor of *IEEE Technical Committee on Cyber-Physical Systems Newsletter*.