

RankTuner: When Design Tool Parameter Tuning Meets Preference Bayesian Optimization

Peng Xu, Su Zheng, Yuyang Ye, Chen Bai, Siyuan Xu,
Hao Geng, Tsung-Yi Ho, Bei Yu

Abstract—Electronic design automation (EDA) tools are critical in the very large scale integration (VLSI) flow. To address the challenges posed by the extensive search space and intricate feature interactions, statistical and machine-learning methods have been employed. These methods aim to model tool parameters and treat the tuning process as a regression task. However, these regression-based methods suffer from inaccurate estimations owing to limited training samples. To address this issue, we propose a ranking-based tool parameter tuning framework, called RankTuner, which directly learns the dominant relationship between parameters. RankTuner utilizes a pairwise Gaussian process to estimate the probability and uncertainty of the dominance relationship. Our approach also integrates a Duel-Thompson sampling method to balance exploration and exploitation in parameter selections. A dimensionality reduction scheme with random embedding and trust region techniques is incorporated to enable parallel searches. Experimental results demonstrate the superiority of RankTuner compared to the cutting-edge tool parameter tuning methods.

I. INTRODUCTION

ELECTRONIC design automation (EDA) tools play an essential role in the VLSI flow. While chip designs have greatly benefited from the continuous scaling of feature sizes, the corresponding design complexity has also been ever-increasing. To meet requirements such as timing closure, reliability, and manufacturability, EDA tools continuously integrate complex algorithms and optimization techniques in both the front-end and back-end design stages to improve the quality of results (QoR). For example, Cadence's Genus is a front-end synthesis tool, while Innovus is a back-end physical design tool, which includes steps such as layout, clock tree synthesis, and routing. These tools involve numerous tunable parameters. In Genus, the "auto partition" parameter enables the use of partitioning algorithms in the design process. In Innovus, the "congestion effort" parameter balances the trade-off between runtime cost and layout quality during global placement, revealing areas in the chip layout that may pose difficulties for routing [1].

This work is partially supported by the Research Grants Council of Hong Kong SAR (No. CUHK14210723 and 14211824), and the MIND project (MINDXZ202404). (Corresponding author: Bei Yu)

Peng Xu, Su Zheng, Yuyang Ye, Tsung-Yi Ho, and Bei Yu are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, NT, Hong Kong SAR.

Chen Bai is with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, NT, Hong Kong SAR. Siyuan Xu is with the Huawei Noah's Ark Lab.

Hao Geng is with the ShanghaiTech University

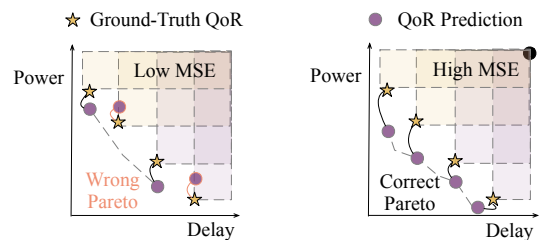


Fig. 1 Regression-based method with low MSE loss and wrong Pareto dominance (left) vs. Ranking-based method with high MSE loss and correct Pareto dominance (right).

These complex algorithms and optimizations are a mixed blessing for designers. While they provide designers with numerous adjustable parameters to enhance result quality significantly, they also make the parameter-tuning process exceptionally challenging. A typical industrial approach involves manual selection for tool parameters by computer chip designers, which requires domain expertise and substantial labor. Furthermore, the increasing complexity of tool parameter spaces presents difficulties because of their extensive magnitude (e.g., the number of combinational parameters can be more than 10^{70} according to [2]). The effectiveness of heuristic optimization methods, such as evolutionary algorithms (EA) [3], [4] and ant colony optimization (ACO) algorithm [5], is hindered by complex feature interactions and mixed type parameters.

To tackle these challenges, statistical and machine learning approaches have been introduced for efficient modeling of tool parameters [2], [6]–[11]. These approaches typically view tool parameter tuning as a regression task and utilize surrogate regression models like XGBoost [6], [9], Neural Networks [7], [8] and Gaussian process [2], [10], [11]. LAMDA [6] captures design-specific features obtained from the design process and leverages the XGBoost algorithm to model the tool parameters to achieve FPGA design closure. FIST [9] utilizes the proposed feature importance sampling to improve the performance of the XGBoost regressor. Recommender [7] incorporates the concept of tensor decomposition from recommender systems to recommend suitable parameter values using a neural network. [2] employs a Gaussian process model as the surrogate model of Bayesian optimization for tuning tool parameters. PTPT [11] further utilizes a multi-task Gaussian process with multi-objective Bayesian optimization to optimize the tool parameters.

Although earlier statistical and machine learning approaches have shown promising results, they primarily focus on predicting the **exact** QoR values of a specific tool parameter. However, using absolute prediction is subject to two limitations. First, the abundance of parameter options leads to high-dimensional inputs, making it challenging to train an accurate regression model [1]. Second, regression-based design space exploration often yields inaccurate Pareto relationship predictions [11], [12] due to a lack of modeling the inherent uncertainty, especially when dealing with expanding Pareto fronts. Consequently, model-based methods for estimating parameters suffer from inaccuracies, leading to biased solutions. As depicted in Fig. 1, when attempting to estimate Pareto boundaries, although regression-based methods might exhibit lower mean squared error (MSE) loss, they are still prone to producing inaccurate estimates (represented by purple circles). Although previous attempts have explored active learning to mitigate this issue in Design Space Exploration (DSE) [13], [14], these sampling methods have not proven effective in tool parameter tuning owing to the huge search space. Thus, we ask the following rhetorical question: *Is it necessary to learn a regression function, or can we directly learn the Pareto dominance relationship between two tool parameters?*

To address the challenges of regression-based methods, we introduce a ranking-based tool parameter tuning method that learns the probability and uncertainty of the dominance relationship from tool parameter pairs, i.e., comparisons. One of the most challenging aspects lies in the uncertainty modeling of the dominance relationship. Unlike regression-based tuning methods that benefit from the Gaussian process regression models, the uncertainty of the dominance relationship requires considering the pairwise relationship between two parameters. In this scenario, the posterior distribution of uncertainty is analytically intractable, and approximations are required, which is not straightforward in contrast to the regression case. Inspired by recent advancements in preference Bayesian optimization [15]–[17], we propose the **RankTuner** framework, which utilizes a pairwise Gaussian process to predict the dominance relationship with uncertainty estimation. To address the challenges of high-dimensional optimization, we further propose a customized dimension reduction method utilizing the EDA tool documentation information. This technique uses the TF-IDF method to represent the text information for each configuration in the EDA tool documentation to cluster configuration parameters with expert knowledge injected.

The main contributions of this paper are listed as follows:

- We introduce a ranking-based tool parameter tuning framework, which learns the probability and uncertainty of the dominance relationship from tool parameter comparisons.
- A pairwise Gaussian process is incorporated to approximate the uncertainty of the dominance relationship between parameter comparisons.
- We further utilize a Duel-Thompson sampling method to trade off the exploration and exploitation of selection with Pareto dominance comparisons. Additionally, we

implement a scheme based on random embedding and Trust Region to reduce dimension and facilitate parallel searches.

- The experimental results demonstrate a significant improvement of the proposed framework compared to the cutting-edge EDA flow parameter tuning methods, with up to 40.34% improvement of the final QoR.

The overall structure of our article is organized as follows. Section II introduces the preliminaries. Section III and Section IV describe the details of the proposed method. Section V presents experimental results that can prove the effectiveness of the proposed framework. Section VI provides a conclusion to our paper.

II. PRELIMINARIES

A. Bayesian Optimization

Bayesian optimization (BO) is a method for global optimization, particularly useful when function evaluations are costly. It uses a surrogate model, commonly a Gaussian process (GP), to approximate the objective function and guide sampling. As new data points are evaluated, the GP parameters are updated. GPs are non-parametric Bayesian models that define distributions over continuous functions, with any finite set of points following a joint Gaussian distribution. The GP model could be fully specified by a mean function and a covariance function, also known as the kernel function, which can be represented as:

$$p(y|\mathbf{x}) = \mathcal{N}(\mu(\mathbf{x}), \Sigma(\mathbf{x})), \quad (1)$$

where $\mu(\mathbf{x})$ is the mean function, typically taken as zero, representing the prior mean of the function, $\Sigma(\mathbf{x}, \mathbf{x}')$ is the covariance function, which measures the similarity between the function values at two points \mathbf{x} and \mathbf{x}' . The GP model can estimate the mean and variance of the points, enabling a trade-off between exploitation and exploration.

To decide the next evaluation point, an acquisition function is used. One commonly used acquisition function is Expected Improvement (EI), which quantifies the expectation of improvement over the current best value, y^* , at a given point \mathbf{x} . The closed-form expression for EI incorporates the posterior mean $m(\mathbf{x})$ and posterior standard deviation $s(\mathbf{x}) = \sqrt{\sigma^2(\mathbf{x})}$ as follows:

$$EI(\mathbf{x}) = (y - m(\mathbf{x})) \Phi\left(\frac{y - m(\mathbf{x})}{s(\mathbf{x})}\right) + s(\mathbf{x}) \psi\left(\frac{y^* - m(\mathbf{x})}{s(\mathbf{x})}\right), \quad (2)$$

where y^* is the current best observed value, $m(\mathbf{x})$ is the posterior mean, $s(\mathbf{x})$ is the posterior standard deviation, $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution, and $\psi(\cdot)$ is the probability density function of the standard normal distribution.

This equation highlights the two key components of EI: exploitation and exploration. The term $(y - m(\mathbf{x})) \Phi\left(\frac{y - m(\mathbf{x})}{s(\mathbf{x})}\right)$ promotes the selection of points with a low predicted mean value; (2) The term $s(\mathbf{x}) \psi\left(\frac{y^* - m(\mathbf{x})}{s(\mathbf{x})}\right)$ encourages exploration

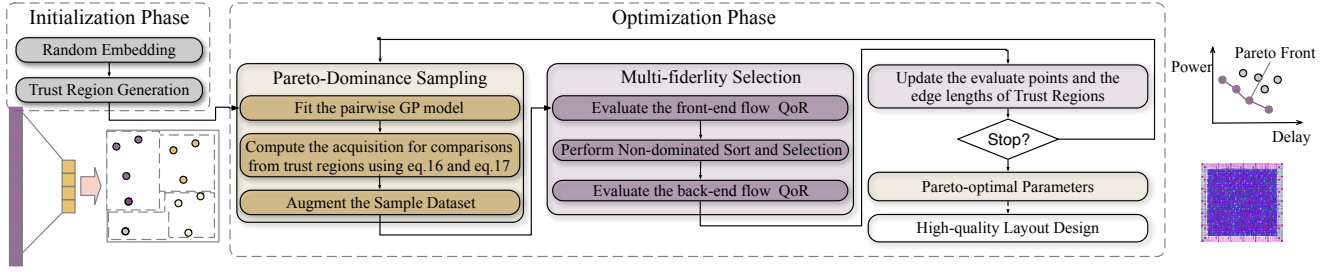


Fig. 2 The overall flow of our RankTuner framework.

of regions with high uncertainty, driven by the posterior standard deviation $s(\mathbf{x})$.

The uncertainty term plays a critical role in exploration by prioritizing regions where the model is less confident about the function's behavior. This balance between exploitation and exploration ensures a more efficient search for the global optimum. However, when dealing with parameter comparisons, designing a suitable acquisition function becomes more challenging. This requires accounting for both the probability and the uncertainty in the dominance relationship.

B. Preference Bayesian Optimization

Preference Bayesian Optimization (PBO) is a variant of Bayesian optimization used to find the optimum of a latent function of interest when the decision-maker (DM) has preferences over the outcomes but cannot express a determined trade-off over outcomes using a single real-valued score measure [15]–[17]. Instead, the DM's preferences are elicited through pairwise comparisons between outcome vectors, which are then used to guide the optimization process [18]. The goal of PBO is to efficiently find the optimal solution that aligns with the DM's preferences, using as few queries (pairwise comparisons) and experiments (function evaluations) as possible. This approach is particularly useful in situations where the outcome function is expensive to evaluate, such as in materials design, robot locomotion, or internet experiments [19].

C. Problem Formulation

In EDA flows, various metrics are used to evaluate the QoR, including performance, power, and area (PPA). Therefore, EDA flow parameters tuning involves optimizing multiple objectives. Typically, we attempt to explore the Pareto front for the Design Tool Parameters space so that an optimal balance among the QoR metrics for designs can be achieved.

Definition 1 (Pareto dominance). *For a multi-objective minimization problem with M objectives, a parameter \mathbf{x}_1 is deemed to dominate parameter \mathbf{x}_2 if, for all m belonging to the set $\{1, \dots, M\}$, the inequality of objective vectors $f_m(\mathbf{x}_1) \leq f_m(\mathbf{x}_2)$ holds true, and there exists at least one k within the same set such that $f_k(\mathbf{x}_1) < f_k(\mathbf{x}_2)$; this relationship is symbolized by $\mathbf{x}_1 \succeq \mathbf{x}_2$.*

Definition 2 (Pareto optimality). *The collection of parameters that remain non-dominated by others constitutes the Pareto-optimal set, thereby establishing Pareto optimality within the parameters space. This collection of Pareto-optimal parameters, which represents the optimal balance of competing objectives, is referred to as the **Pareto front**.*

Definition 3 (Pareto hypervolume). *Pareto hypervolume (HV) is the Lebesgue measure of the space dominated by the Pareto frontier and bounded by a reference point \mathbf{v}_{ref} as follows:*

$$\text{HV}_{\mathbf{v}_{\text{ref}}}(\mathcal{P}(\mathcal{Y})) = \int_{\mathcal{Y}} \mathbb{I}[\mathbf{y} \succ \mathbf{v}_{\text{ref}}] \left[1 - \prod_{\mathbf{y}_* \in \mathcal{P}(\mathcal{Y})} \mathbb{I}[\mathbf{y}_* \not\prec \mathbf{y}] \right] d\mathbf{y}, \quad (3)$$

where $\mathbb{I}(\cdot)$ is the indicator function, which outputs 1 if its argument is true and 0 otherwise, $\mathcal{P}(\mathcal{Y})$ is the Pareto frontier.

Problem 1 (Ranking-based tool parameter tuning). *Given a parameter search space \mathcal{X} , each tool parameter inside \mathcal{X} is regarded as a feature vector \mathbf{x} , and the corresponding QoR \mathbf{y} forms the objective space \mathcal{Y} . For each \mathbf{x} , the corresponding QoR metrics $\mathbf{y} \in \mathcal{Y}$, which can be estimated through the VLSI implementation flow f . Ranking-based tool parameter tuning is to predict the dominant relationship between parameters for exploring the Pareto optimality, intending to maximize HV while minimizing runtime.*

III. RANKTUNER FRAMEWORK

This section introduces RankTuner, a parameter tuning framework that differs from regression-based methods by using a pairwise Gaussian process to estimate dominance probabilities and guide Pareto front selection. RankTuner tackles key challenges by modeling dominance with uncertainty across multiple objectives, applying Duel-Thompson Sampling to balance exploration and exploitation, and integrating random embedding with trust-region techniques for efficient search in high-dimensional parameter spaces. The remainder of the section details the pairwise GP model (Section III-B), acquisition function (Section III-C), key steps (Section III-D), and complete algorithm (Section III-E).

A. Overview

Fig. 2 illustrates the overall flow of the proposed RankTuner framework, clarifying each major step from initialization to final output. The process begins with the Initialization

Phase, where parameter samples are generated using random embedding and trust region generation to explore the high-dimensional search space efficiently. In the subsequent Optimization Phase, RankTuner first performs Pareto-dominance sampling by fitting a pairwise Gaussian Process model to estimate the probability that one parameter configuration dominates another, incorporating uncertainty into the selection process. The acquisition function then guides the selection of informative parameter pairs for comparison, and new samples are iteratively added to the dataset. Next, during multi-fidelity selection, candidate parameter settings are evaluated at different stages of the EDA flow, with non-dominated sorting and selection used to focus on the most promising candidates. At each iteration, the framework updates both the evaluation points and the trust region boundaries, as shown by the feedback in the figure, and continues this loop until a stopping criterion is met. Finally, the framework outputs the Pareto-optimal parameters and produces a high-quality layout design, as illustrated by the Pareto front and layout images on the right, providing an efficient and effective solution for multi-objective EDA parameter optimization.

B. The Pairwise Gaussian Process

To model the dominance relationships between parameter pairs using Gaussian processes, we assume the existence of an unobserved latent function f that captures the Pareto dominance relationships between different parameters. We employ a Gaussian process prior to this latent function f and utilize a pairwise likelihood function [20] to learn the Pareto dominance relationships between different parameter pairs.

Under the assumption that the latent function maintains a consistent Pareto dominance relationship with different parameters, to capture the Pareto dominance relationship in Section II-C, we utilize a pairwise likelihood function defined as:

$$p_{\text{ideal}}(\mathbf{x}_v \succeq \mathbf{x}_u | f(\mathbf{x}_v), f(\mathbf{x}_u)) = \begin{cases} 1 & \text{if } f(\mathbf{x}_v) \geq f(\mathbf{x}_u) \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

This function reflects the degree of configuration \mathbf{x}_v dominating configuration \mathbf{x}_u , under the influence of the latent function f . If the latent function value of \mathbf{x}_v is greater than or equal to that of \mathbf{x}_u , we assign a value of 1 to indicate that \mathbf{x}_v is Pareto-dominant over \mathbf{x}_u . Conversely, if the latent function value of \mathbf{x}_v is lower than that of \mathbf{x}_u , we assign a value of 0 to indicate that \mathbf{x}_v is not dominant in the Pareto relationship.

Traditional regression-based methods often use a regression model to predict f , and then directly use f to determine the dominance relationship between two parameters [2], [11]. However, this approach overlooks the uncertainty in the derived dominance relationships caused by data scarcity and model errors. Considering the complexity and nonlinearity of the Pareto relationship, we assume that there is a Gaussian noise between the latent function f and the true Pareto dominance relationship. This Gaussian noise has a zero mean and an unknown variance σ^2 . The dominance uncertainty can

be represented as the overlapping regions of two noise sources, as shown in Fig. 3(a).

Then, the pairwise likelihood function could be formulated as:

$$\begin{aligned} \Phi(z_k) &= p(\mathbf{x}_v \succeq \mathbf{x}_u | f(\mathbf{x}_v), f(\mathbf{x}_u)), \\ &= \iint p_{\text{ideal}}(\mathbf{x}_v \succeq \mathbf{x}_k | f(\mathbf{x}_v) + \delta_v, f(\mathbf{x}_u) + \delta_u) \\ &\quad \mathcal{N}(\delta_v; 0, \sigma^2) \mathcal{N}(\delta_u; 0, \sigma^2) d\delta_v d\delta_u, \end{aligned} \quad (5)$$

where $z_k = \frac{f(\mathbf{x}_u) - f(\mathbf{x}_v)}{\sqrt{2}\sigma}$ and $\Phi(z) = \int_{-\infty}^z \mathcal{N}(\gamma; 0, 1) d\gamma$.

During the inference, given a test parameter pair $(\mathbf{x}_r, \mathbf{x}_s)$, when the Pareto-dominance relation is unknown. The zero-mean latent variables $\mathbf{f}_t = [f(\mathbf{x}_r), f(\mathbf{x}_s)]^\top$ have correlations with the n zero-mean random variables of training samples $\{f(\mathbf{x}_i)\}_{i=1}^n$. We compute the prior joint multivariate Gaussian distribution as follows:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_t \end{bmatrix} \sim \mathcal{N} \left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \Sigma & \mathbf{k}_t \\ \mathbf{k}_t^\top & \Sigma_t \end{pmatrix} \right) \quad (6)$$

where

$$\mathbf{k}_t = \begin{bmatrix} \mathcal{K}(\mathbf{x}_r, \mathbf{x}_1), \mathcal{K}(\mathbf{x}_r, \mathbf{x}_2), \dots, \mathcal{K}(\mathbf{x}_r, \mathbf{x}_n) \\ \mathcal{K}(\mathbf{x}_s, \mathbf{x}_1), \mathcal{K}(\mathbf{x}_s, \mathbf{x}_2), \dots, \mathcal{K}(\mathbf{x}_s, \mathbf{x}_n) \end{bmatrix}^\top$$

and

$$\Sigma_t = \begin{bmatrix} \mathcal{K}(\mathbf{x}_r, \mathbf{x}_r) & \mathcal{K}(\mathbf{x}_r, \mathbf{x}_s) \\ \mathcal{K}(\mathbf{x}_s, \mathbf{x}_r) & \mathcal{K}(\mathbf{x}_s, \mathbf{x}_s) \end{bmatrix}.$$

The predictive distribution of $p(\mathbf{f}_t | \mathcal{D})$ can be computed as an integral over f -space, which can be written as:

$$p(\mathbf{f}_t | \mathcal{D}) = \int p(\mathbf{f}_t | \mathbf{f}) p(\mathbf{f} | \mathcal{D}) d\mathbf{f}, \quad (7)$$

where \mathcal{D} is the observed parameters data so far. Although it is computationally challenging to optimize the integral operation, we can approximate the posterior distribution $p(\mathbf{f} | \mathcal{D})$ as a Gaussian distribution centered on \mathbf{f}_{MAP} with the covariance matrix $(\Sigma^{-1} + \Lambda_{\text{MAP}})^{-1}$ using Laplace approximation according to [20].

The predictive distribution in Equation (7) can then be simplified as a Gaussian distribution $\mathcal{N}(\mathbf{f}^*; \mu^*, \Sigma^*)$ with mean μ^* and variance Σ^* , where:

$$\mu^* = [\mu_r^*, \mu_s^*]^\top = \mathbf{k}_t^\top \Sigma^{-1} \mathbf{f}_{\text{MAP}}, \quad (8)$$

and

$$\Sigma^* = \begin{bmatrix} \Sigma_{rr}^* & \Sigma_{rs}^* \\ \Sigma_{sr}^* & \Sigma_{ss}^* \end{bmatrix} = \Sigma_t - \mathbf{k}_t^\top (\Sigma + \Lambda_{\text{MAP}}^{-1})^{-1} \mathbf{k}_t. \quad (9)$$

The predictive preference $p(\mathbf{x}_r \succeq \mathbf{x}_s | \mathcal{D})$ can be evaluated by the integral $\int p(\mathbf{x}_r \succeq \mathbf{x}_s | \mathbf{f}_t, \mathcal{D}) p(\mathbf{f}_t | \mathcal{D}) d\mathbf{f}_t$ that yields:

$$p(\mathbf{x}_r \succeq \mathbf{x}_s | \mathcal{D}) = \Phi \left(\frac{\mu_r^* - \mu_s^*}{\sigma_*} \right),$$

where $\sigma_*^2 = 2\sigma^2 + \Sigma_{rr}^* + \Sigma_{ss}^* - \Sigma_{rs}^* - \Sigma_{sr}^*$.

C. Acquisition Function for Pareto-dominance Comparison

After establishing the pairwise Gaussian process model for Pareto dominance comparison pairs, another important ques-

tion is how to determine an acquisition function that allows us to identify the Pareto-dominant solution set efficiently. The challenge here lies in balancing exploration and exploitation based on the comparison model. We will first introduce the principles of exploration and exploitation in the context of comparison-based models. Then, based on these principles, we will introduce the Duel-Thompson sampling method for the acquisition function specifically tailored for selecting Pareto dominance comparisons.

The probability of a preference outcome in a comparison is modeled using an inverse link function, π_f , which maps a latent loss function f to the probability that one input is preferred over another. It satisfies the symmetry property $\pi_f([\mathbf{x}', \mathbf{x}]) = 1 - \pi_f([\mathbf{x}, \mathbf{x}'])$. A natural choice for π_f is the sigmoid function $\phi(z) = \frac{1}{1+e^{-z}}$, which maps the latent function difference $f([\mathbf{x}, \mathbf{x}'])$ to the probability range $[0, 1]$. We assume that N comparisons have already performed, resulting in a dataset $\mathcal{D} = \{[x_i, x'_i], y_i\}_{i=1}^N$. With this dataset \mathcal{D} , we can make inferences about the latent function f and its wrapped version $\pi_{f,\theta}$ using the pairwise Gaussian process (GP) in Section III-B for Pareto dominance classification:

$$\begin{aligned} \pi_f([\mathbf{x}_*, \mathbf{x}'_*]; \mathcal{D}, \theta) &= p(y_* = 1 \mid \mathcal{D}, [\mathbf{x}, \mathbf{x}'], \theta), \\ &= \int \sigma(f_*) p(f_* \mid \mathcal{D}, [\mathbf{x}_*, \mathbf{x}'_*], \theta) df_*, \end{aligned} \quad (10)$$

where θ represents the parameters of the pairwise GP.

Potential Comparisons Exploration. For the pairwise Gaussian process model, the output variable y_c follows a Bernoulli distribution with probability given by the preference function π_f . Therefore, the variance of the output variable y_c does not necessarily decrease with sufficient observations. For example, when a non-dominance relationship exists between \mathbf{y} and \mathbf{y}' , y_c tends to approach 0.5. Hence, the exploration scheme of regression-based Bayesian optimization may not result in effective exploration for parameter comparisons.

As an alternative, the Exploration can be conducted by searching for the Pareto comparisons with the highest uncertainty in probability. This uncertainty can be quantified by modeling the probability of outcomes using pairwise Gaussian processes, measured by the variance of $\phi(f_*)$, which is visualized in Fig. 3(c). The variance of $\phi(f_*)$ could be mathematically defined as:

$$\begin{aligned} \mathbb{V}[\phi(f_*)] &= \int (\phi(f_*) - \mathbb{E}[\phi(f_*)])^2 p(f_* \mid \mathcal{D}, [\mathbf{x}, \mathbf{x}']) df_* \\ &= \int \phi(f_*)^2 p(f_* \mid \mathcal{D}, [\mathbf{x}, \mathbf{x}']) df_* - \mathbb{E}[\phi(f_*)]^2, \end{aligned} \quad (11)$$

which explicitly takes into account the uncertainty over $\phi(f)$. However, pure exploration methods do not leverage the available knowledge about the current Pareto front, which can result in a lack of effectiveness in exploring new Pareto-dominant solutions.

In traditional Bayesian optimization, exploration is computed in an expected manner relative to the marginal gain concerning the current best-observed output. However, in the

context of parameter dominance comparison, we can evaluate the quality of a single point's dominance using a conditional integration-based approach.

Specifically, we employ a soft winner scoring method, which is used in various ranking methods. It can be defined as follows:

$$C(\mathbf{x}) = \text{Vol}(\mathcal{X})^{-1} \int_{\mathcal{X}} \pi_f([\mathbf{x}, \mathbf{x}']) d\mathbf{x}', \quad (12)$$

where $\text{Vol}(\mathcal{X}) = \int_{\mathcal{X}} d\mathbf{x}'$ is a normalizing constant that bound $C(\mathbf{x})$ in the interval $[0, 1]$. The equation aims to compute the averaged probability that \mathbf{x} is the dominator in all current comparisons.

Pareto-Dominance Thompson Sampling. As previously detailed, pure exploration methods do not leverage the current Pareto front, and relying solely on exploration can lead to local optimism. We introduce an alternative acquisition function called Duel-Thompson sampling [21]. It follows a two-step strategy, which is outlined as follows:

- 1) Selecting \mathbf{x} : First, generate a sample \tilde{f} from the model using continuous Thompson sampling and compute the associated Expected Improvement score using Equation (12). The first element of the new comparison, \mathbf{x}_{next} , is selected as:

$$\mathbf{x}_{\text{next}} = \arg \max_{\mathbf{x} \in \mathcal{X}} \int_{\mathcal{X}} \pi_{\tilde{f}}([\mathbf{x}, \mathbf{x}']) d\mathbf{x}'. \quad (13)$$

As more evaluations are collected, the selection becomes more greedy toward the dominant parameter inferred by the model. Also, the policy allows exploration of other parameters based on the stochastic \tilde{f} in the initial phase.

- 2) Selecting \mathbf{x}' : Given \mathbf{x}_{next} as the first element of the selected comparison, the second element is selected as the parameter that maximizes the variance of $\phi(f_*)$ in the direction of \mathbf{x}_{next} . Formally, $\mathbf{x}'_{\text{next}}$ is selected as:

$$\mathbf{x}'_{\text{next}} = \arg \max_{\mathbf{x}'_* \in \mathcal{X}} \mathbb{V}[\sigma(f_*) \mid [\mathbf{x}_*, \mathbf{x}'_*], \mathbf{x}_* = \mathbf{x}_{\text{next}}]. \quad (14)$$

This second step is purely exploration in the direction of \mathbf{x}_{next} , which aims to find informative comparisons to run with parameters from the current Pareto front.

Extension to Batched Exploration with EUBO. To generalize our exploration strategy to batched preference exploration, we can adopt the EUBO- f method proposed by Lin et al. [16]. This approach samples multiple surrogate paths \tilde{f} of the outcome model and selects several queries in parallel:

$$(y_1^{(b)}, y_2^{(b)}) := \arg \max_{y_1, y_2 \in \tilde{f}(\mathcal{X})} \text{EUBO}(y_1, y_2), \quad b = 1, \dots, N_B, \quad (15)$$

where N_B denotes the batch size. This scheme mimics multi-duel strategies through diversified exploration within each batch, while still preserving the Bayesian optimality guarantees of the EUBO framework.

It is also worth noting that the single-query variant is closely related to the criterion of EUBO. EUBO provides a one-step

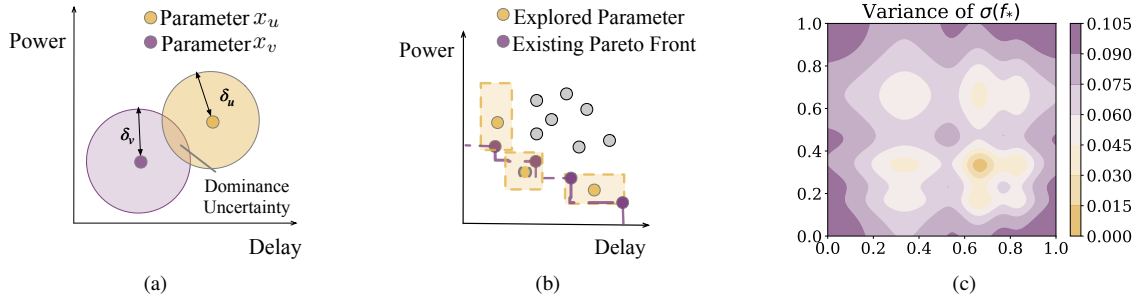


Fig. 3 The visualization of concepts in RankTuner frameworks: (a) The dominating uncertainty region between two parameters x_u and x_v ; (b) Exploit the existing Pareto front (purple points) vs. Explore the unknown parameters with maximum dominating uncertainty region (golden points); (c) The variance of $\phi(f_*)$ decreases in areas with available data, making it suitable for the exploration of comparisons.

Bayes-optimal strategy for preference elicitation, proving that even a single carefully chosen comparison suffices to approximate the optimal expected information gain. Formally, given two candidate outcomes $y_1, y_2 \in \mathbb{R}^k$, the EUBO acquisition function is defined as

$$\text{EUBO}(y_1, y_2) := \mathbb{E}_m[\max\{g(y_1), g(y_2)\}], \quad (16)$$

where $\mathbb{E}_m[\cdot]$ denotes the posterior expectation of the preference model g conditioned on observed comparisons. Lin et al. [16] demonstrate that, even under noisy preferences, maximizing $\text{EUBO}(y_1, y_2)$ produces queries that approximate the one-step optimal value function.

When computational resources are sufficient, the batched EUBO strategy can be combined with our proposed trust-region search to accelerate exploration. Under limited resources, the single-query acquisition function can be viewed as a special case of EUBO. In the subsequent experiments, we adopt EUBO as the default acquisition function.

D. The Key Steps of RankTuner

EDA Tool Parameter Tuning is a high-dimensional optimization problem, which means finding the optimal configuration among an enormous parameters. To accelerate the parameter exploration, we employ the techniques of Random Embedding and Trust Region. Then, the center of the Trust Region is initialized using a clustering algorithm, and the search space is divided into N regions. Subsequently, we utilize the Pairwise GP model to sample informative comparisons between trust regions. Due to the time-consuming evaluation process in EDA flow, we adopt a multi-fidelity evaluation strategy to further accelerate the process, which employs faster front-end QoR results to filter the unpromising parameters.

Random Embedding Generation. The random embedding generation involves generating a lower-dimensional latent variable vector x' and transforming it to the original parameter vector x using an embedding matrix A . By optimizing the

embedding matrix using simulated annealing, the transformed parameter vector is encouraged to fall within the desired range:

$$A = \arg \min_A (1 - p(Ax' \in B_D)) + \gamma \times D_{KL}(p_A \| \mathcal{N}(\mu, \sigma)), \quad (17)$$

where $p(Ax' \in B_D)$ represents the probability that Ax' falls into B_D , $D_{KL}(p\|q) = \sum p(x) \log \frac{p(x)}{q(x)}$ is the KL divergence, p_A is the probability distribution of the elements of A , $\mathcal{N}(\mu, \sigma)$ denotes the standard Gaussian distribution, and γ is a weight parameter that makes a trade-off between the two terms.

The random embedding method allows for more efficient parameter tuning as it cuts down the original problem's dimension while maintaining the effectiveness of optimization.

Diversity-Aware Trust-region Initialization. Our algorithm employs Bayesian optimization with decoupled trust regions (TRs) [22]. After generating the embedding matrix A , we first generate a diverse set of random candidate samples using a low-discrepancy Sobol sequence in the latent space [23] using the implementation of the Botorch framework [23]. These samples are mapped to the original parameter space B_D and partitioned into T subsets using k -means clustering [24], where T denotes the number of trust regions. The cluster centers $x_{center,i}$ are then used as the initial trust region centers Ω_i , and each trust region is formulated as a hypercube centered at $x_{center,i}$ with initial edge length L .

We adopt k -means because it provides representative centers with low computational overhead and scales well to large sample sizes. Thus, while k -means serves as the default for efficiency, the initialization strategy does not rely critically on this algorithm. Sobol sampling together with clustering ensures structured and diverse coverage, while the use of multiple trust regions in parallel reduces the risk of missing globally promising areas.

Informative Comparison Selection between Regions. After establishing the Trust Region framework, we obtain N exploration regions $\Omega_1, \Omega_2, \dots, \Omega_T$ and a shared dataset D_t aggregated across them. For each region, initialization data from Ω_i are combined with D_t to form Pareto comparisons, which train a pairwise GP model. The key challenge is to select

informative comparisons that guide exploration. While our ultimate goal is global Pareto optimality, exploration focuses on each trust region, enabling efficient parallel search with information sharing through D_t .

Specifically, our comparison selection proceeds in two steps: First, we choose a global candidate \mathbf{x}_{next} by maximizing the acquisition function of the pairwise GP model. The surrogate is trained on the shared dataset D_t , but the maximization itself is performed over the entire continuous design space, ensuring the method can propose new, unseen candidates rather than being restricted to previously evaluated points. The candidate is obtained using Thompson sampling as follows:

$$\mathbf{x}_{\text{next}} = \arg \max_{\mathbf{x} \in \mathcal{X}} \int_{D_t} \pi_{\tilde{f}}([\mathbf{x}, \mathbf{x}']) d\mathbf{x}'. \quad (18)$$

where \mathcal{X} denotes the full design space and $\pi_{\tilde{f}}$ is the domination probability under a sampled function \tilde{f} from the pairwise GP posterior. Then, given \mathbf{x}_{next} , we select its comparison partner locally within each trust region Ω_i by maximizing the predictive variance of the derived feature map $\phi(f_\star)$:

$$\mathbf{x}'_{\text{next}} = \arg \max_{\mathbf{x}'_\star \in \Omega_i} \mathbb{V}[\phi(f_\star) \mid [\mathbf{x}_\star, \mathbf{x}'_\star], \mathbf{x}_\star = \mathbf{x}_{\text{next}}]. \quad (19)$$

This two-stage strategy emphasizes exploration: \mathbf{x}_{next} is drawn from the continuous design space to probe promising yet unobserved regions, while $\mathbf{x}'_{\text{next}}$ leverages uncertainty within each trust region to refine the local Pareto front. At each step, this enables parallel exploration across all regions while still maintaining alignment toward global optimality.

Acquisition Maximization. In preferential Bayesian optimization [19], the acquisition function is defined over a continuous dual space, where exact maximization is generally intractable. To address this, we approximate the argmax numerically within a low-dimensional trust-region embedding. Specifically, we adopt a multi-start, gradient-based search using the L-BFGS optimizer with random restarts [25], following the default acquisition maximization strategy of BoTorch [23], implemented through the standard optimization function `optimize_acqf`. Our procedure parallels the treatment of the inner integral approximation via Monte Carlo: randomized initialization, local refinement, and subsequent projection back to the original design space.

Multi-fidelity Evaluation and Update. After sampling, we evaluate the $2b$ points using the front-end flow and get the early-stage results $\mathbf{f}_f(\mathbf{x}_i)$. We then apply a non-dominated selection process, i.e., NSGA-II [26], to select the better half of the samples. Unlike absolute thresholds, relative ranking offers two advantages: (1) it mitigates discrepancies in PPA magnitudes between the front-end and back-end stages, and (2) it avoids prematurely constraining exploration. The selection ratio used in our implementation is a tunable parameter: the ratio can be adjusted as needed by downstream tasks. This fronted information can serve as an early-stopping sign, allowing us to explore more points while keeping the overhead acceptable. The training data and the Pareto-optimal set are

Algorithm 1 The Optimization Phase of RankTuner

Input: Initialized Trust regions $\{\Omega_1, \Omega_2, \dots, \Omega_T\}$, batch size b , number of iterations T

Output: Pareto-optimal Parameter set \mathcal{P}

```

1: procedure Optimize( $\Omega_i, b$ )
2:   Fit the pairwise GP model  $\pi(\cdot)$  with the evaluated
   points in  $\Omega_i$  and the shared data  $D_t$ ;
3:    $\mathbf{X}_{\text{sampled}} \leftarrow \emptyset$ ;
4:   for  $k = 1, \dots, N_e$  do
5:      $[\mathbf{x}_{t+1}, \mathbf{x}'_{t+1}] \leftarrow \arg \max_{\mathbf{x} \in D_t, \mathbf{x}' \in \Omega_i} \alpha(\mathbf{x}, \mathbf{x}')$ 
6:     Augment  $\mathbf{X}_{\text{sampled}} \leftarrow \mathbf{X}_{\text{sampled}} \cup \{\mathbf{x}_k, \mathbf{x}'_k\} \cap \Omega_i$ 
7:   end for
8:    $\mathbf{X}_{\text{selected}}, \mathbf{Y}_{\text{selected}} \leftarrow \text{Selection}(\mathbf{X}_{\text{sampled}})$ 
9:   Update the evaluated points and the edge-length of  $\Omega_i$ 
   with  $\mathbf{X}_{\text{selected}}, \mathbf{Y}_{\text{selected}}$ ;
10:  return  $\Omega_i$ 
11: end procedure
12: procedure Selection( $\mathbf{X}_{\text{sampled}}$ )
13:  Map samples  $\mathbf{X}_{\text{sampled}}$  to the original parameter
   space, evaluate them with the front-end flow, get results
    $\mathbf{Y}_{f, \text{sampled}}$ 
14:   $\mathbf{X}_{\text{selected}} \leftarrow \emptyset$ 
15:  while  $|\mathbf{X}_{\text{selected}}| < b$  do
16:     $\mathbf{P}_{\text{sampled}} \leftarrow \text{Pareto}(\mathbf{Y}_{f, \text{sampled}})$ 
17:    Sort  $\mathbf{P}_{\text{sampled}}$  by the crowding distance
18:    Add points from  $\mathbf{P}_{\text{sampled}}$  to  $\mathbf{X}_{\text{selected}}$ 
19:    Remove the points from  $\mathbf{X}_{\text{sampled}}$ 
20:  end while
21:  Map samples  $\mathbf{X}_{\text{selected}}$  to the original parameter space,
   evaluate them with the back-end flow, and get results
    $\mathbf{Y}_{\text{selected}}$ 
22:  return  $\mathbf{X}_{\text{selected}}, \mathbf{Y}_{\text{selected}}$ 
23: end procedure
24: for iteration in  $\{1, 2, \dots, T\}$  do
25:   for  $\Omega_i$  in  $\{\Omega_1, \Omega_2, \dots, \Omega_T\}$  do
26:      $\Omega_i = \text{Optimize}(\Omega_i, b)$ ;
27:   end for
28:   Update the Pareto-optimal set and the shared data;
29: end for
```

updated after selecting the points and obtaining their QoR results from the back-end flow. We will update the training data.

The trust regions are updated according to the following rules: (1) If the best point or the Pareto-optimal set of a trust region improves, the success count of that trust region increases; otherwise, the failure count increases; (2) When the success count exceeds a threshold τ_s , the edge length is increased to $\min\{2L, L_{\max}\}$, promoting exploration in a larger space; (3) When the failure count exceeds a threshold τ_f , the edge length is set to $L/2$, encouraging more fine-grained searches; (4) If a trust region has $L < L_{\min}$, it is terminated, and a new trust region is started.

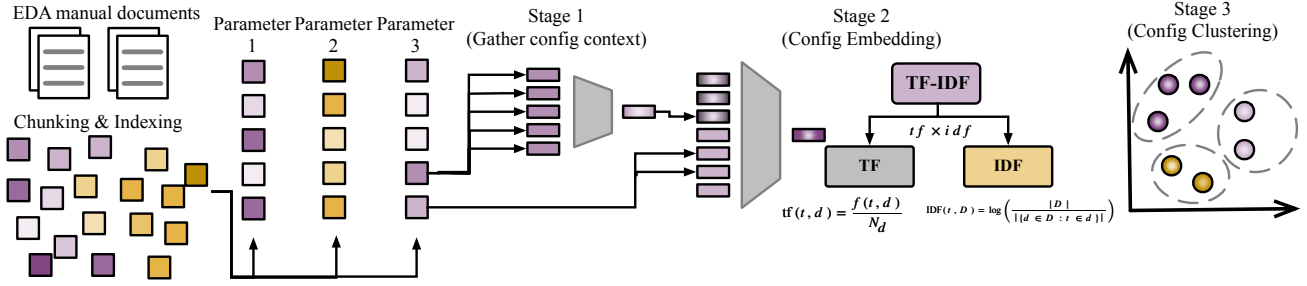


Fig. 4 The overview of the knowledge-enhanced tuning framework.

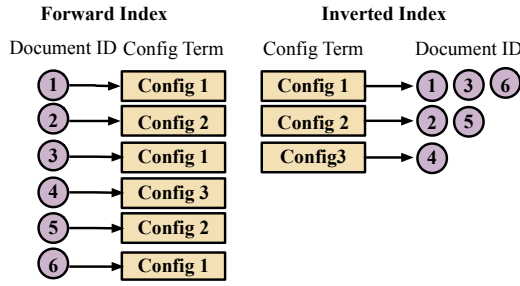


Fig. 5 The comparison between the forward Index and inverted Index.

E. The Overall Tuning Framework

The detailed algorithm of our framework is described in Algorithm 1. The RankTuner framework proceeds by concurrently invoking the Optimize algorithm (Line 8). The Optimize algorithm first utilizes the evaluated comparisons in each trust region Ω_i and the shared data D_t to fit the pairwise GP model (lines 2). Then, it iteratively performs informative comparison selection (line 3-6), multi-fidelity evaluation (line 8 and line 11-16) and update (line 9). After reaching the maximum iterations, we can obtain the final optimal tool parameters.

IV. KNOWLEDGE ENHANCED TUNING FRAMEWORK

EDA design consists of front-end and back-end phases, each governed by parameters that affect circuit depth, node count, timing, and area. While EDA manuals provide detailed parameter explanations for EDA design tools, these are often neglected in prior tuning work. To address this, we developed a data processing pipeline that gathers textual contexts from documentation, generates knowledge-enhanced embeddings, and clusters them for representation. As illustrated in Fig. 4, the pipeline includes three stages—config context gathering, embedding, and clustering. By incorporating domain knowledge through TF-IDF encoding and K-means clustering, our **KRankTuner** framework can better model the search space and improve QoR results.

A. Document Data Preparation and Preprocessing

Preparing the relevant explanation from EDA documentation for each config encompasses several essential steps. Initially, the document stream of the reference manual for EDA tools, which includes various formats such as web pages and PDFs, is standardized into a unified text format. Subsequently, documentation segmentation divides the content into retrievable units based on predefined sizes. Following this, indexable configurations are identified by formulating rules that enable a tokenizer to recognize them accurately. The removal of stop words is also utilized to filter out insignificant terms, thereby enhancing the relevance of the retrieval configuration contexts.

For the naive method of configure context gathering, a sequential search is used to find all text segments containing the corresponding parameter keywords. However, this approach would result in a query complexity of $O(n)$ for traversing the entire document set. Given the numerous optimization parameters and the length of the documents themselves, this could lead to significant time overhead. Therefore, we implemented the inverted index, a well-known data structure in information retrieval, to create a sorted array of indexable EDA parameters.

The comparison between the forward and inverted index data structures is shown in Fig. 5. The efficiency of the inverted index stems from its ability to retrieve only the relevant entries associated with the queried term, rather than scanning through all documents. This results in a time complexity of $O(k)$, where k denotes the number of distinct parameters, which also equals the number of documents that contain the term t_j , a complexity far lower than scanning the entire document collection. This stage is executed once at the beginning of each tuning process.

B. Customized Parameter Embedding with TF-IDF Method

The Tool Parameter optimization problem involves a vast search space, which makes parameter tuning highly challenging. As dimensionality increases, optimization becomes less effective and more time-consuming. Prior work addressed this by reducing the search space. For instance, REMOTune adopted a Random Embedding-based approach [1], [27]. However, its reliance on initialization can yield poor results under certain designs, and it overlooks correlations between parameters—for example, those within the same stage often induce similar effects on the outcome.

We have expanded the original dimensionality reduction approach by using a knowledge-enhanced embedding method. This involves generating corresponding text embeddings for the parameters using the TF-IDF method to interpret the documentation related to different parameters. Specifically, we first processed the explanatory documents related to the EDA parameters, obtaining specific explanatory text segments for each parameter through text processing. We then used the TF-IDF method to generate embeddings for these documents. TF-IDF is a classic text processing method that generates corresponding embeddings for text data based on its term frequency and inverse document frequency [28].

$\text{TF}(\cdot, \cdot)$ is term frequency function of a term t in a document d calculated as:

$$\text{TF}(t, d) = \frac{f(t, d)}{N_d}, \quad (20)$$

where $f(t, d)$ is the frequency of term t in document d , N_d is the total number of terms in document d . The inverse document frequency $\text{IDF}(\cdot, \cdot)$ of a term t across a corpus D is given by:

$$\text{IDF}(t, D) = \log \left(\frac{|D|}{|\{d \in D : t \in d\}|} \right), \quad (21)$$

where $|D|$ is the total number of documents in the corpus, $|\{d \in D : t \in d\}|$ is the number of documents containing the term t . The TF-IDF score for a term t in a document d within a corpus D is defined as:

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D). \quad (22)$$

The TF-IDF score reflects the importance of the term t in the context of document d relative to the entire corpus D .

Once the TF-IDF matrix is generated, we cluster these document-informed parameter embeddings using the K-means method. The clustering process starts with randomly chosen centroids, then iteratively assigns each parameter embedding to its closest centroid, updates the centroid positions based on assigned points, and repeats until convergence. This produces k clusters C_1, C_2, \dots, C_k , where k is the predefined embedding dimension. From this, we derive an assignment matrix $\mathbf{A} \in \mathbb{R}^{|T| \times k}$, in which each parameter is associated with the cluster that best reflects its contextual meaning. During optimization, the original $|T|$ -dimensional parameter vector \mathbf{x} is projected into a k -dimensional latent representation $\mathbf{z} = \mathbf{Ax}$. Each parameter value is normalized into the range $[-1, 1]$, with discrete parameters mapped to real values in the same interval. Optimization is then performed in the latent space \mathbf{z} , rather than directly on \mathbf{x} . To ensure that back-projected parameter values remain valid, \mathbf{z} is constrained within a bounding box $\mathbf{B}_e = [-b_e, b_e]^{d_e}$, where $b_e \leq 1$.

This knowledge-enhanced embedding procedure thus provides a structured way of reducing the dimensionality of parameter spaces while incorporating semantic context from documentation, better aligning parameter tuning with design insights.

TABLE I Examples of the flow parameters.

Stage (Total)	Parameter Examples	Range or Options
Synthesis (105)	auto partition	F/T
	logic optimization	B/A/N
Floorplan (7)	aspect ratio	0.5-2.0
	density target	0.5-1.0
Global Placement (10)	congestion effort	L/M/H
	timing effort	M/H
Detailed Placement (3)	IR drop aware	N/L/M/H
	wirelength optimization	N/M/H
Routing (9)	timing driven	F/T
	Si driven	F/T

(N, L, M, H represent none, low, medium, high, respectively.)
(F, T, B, A represent false, true, basic, and advanced, respectively.)

C. Preference Bayesian Optimization in Latent Space

We conduct knowledge-enhanced parameter optimization by applying preference Bayesian optimization within the latent space derived from expert-document embeddings. Each tool parameter vector \mathbf{x} is initially encoded as a TF-IDF latent vector \mathbf{z} , and these embeddings are subsequently clustered to capture semantic correlations among parameters.

To model the relationship between parameter embeddings and QoR, we employ a pairwise Gaussian process with a scaled automatic relevance determination (ARD) RBF kernel [29]. The kernel regulates variance and captures the relevance of each dimension, with an output-scale parameter σ^2 and an automatic relevance determination technique:

$$\mathcal{K}_{\text{scaled}}(\mathbf{z}, \mathbf{z}') = \sigma^2 \exp \left(- \sum_{d=1}^D \frac{(z_d - z'_d)^2}{2\ell_d^2} \right), \quad (23)$$

where σ^2 controls the global amplitude, while the length scales ℓ_d assign relative importance to each input dimension of \mathbf{z} , enabling the kernel to capture heterogeneous feature sensitivities and dynamically weight them.

The scaled RBF kernel captures the influence of each latent dimension of \mathbf{z} on the observed PPA outcomes with dynamic weighting. Subsequently, the Duel Thompson Sampling is then employed within the latent space to quantify predictive uncertainty and to adaptively select the most informative configuration pairs for evaluation in the next optimization step.

V. EXPERIMENTS

A. Experimental Setting

Parameter Space.

Our parameter space follows PTPT and REMOTune [1], [11], covering essential Genus and Innovus options across synthesis, floorplan, placement, and routing (TABLE I). In synthesis, we tune partitioning, optimization, and multi-bit instances; in floorplan, key geometric settings; in placement, timing, power, congestion, and wire length; and in routing, via,

TABLE II Comparison of Parameter Tuning Methods on RISC-V32I benchmark.

Method	FIST [9]	DAC'19 [7]	MLCAD'19 [2]	ICCAD'21 [10]	PTPT [11]	REMOTune [1]	DATE'24 [30]	ICCAD'24 [31]	Ours
HV (10^5)	1.57	1.55	1.63	1.68	1.48	1.75	1.44	1.84	1.92
HV _{0,1} (10^3)	2.85	2.72	3.00	2.95	2.70	3.05	2.63	3.44	3.55
HV _{0,2} (10^3)	2.94	2.99	3.00	3.07	2.95	3.12	2.84	3.43	3.44
HV _{1,2} (10^3)	2.97	2.97	3.00	3.14	2.79	3.23	2.77	3.00	3.15
MPI1 (%)	3.16	2.54	5.00	3.81	3.56	4.38	2.08	13.64	13.73
MPI2 (%)	3.90	2.12	5.12	5.23	0.85	6.27	0.68	5.04	7.30
MAI (%)	5.47	7.18	4.64	7.10	5.15	7.45	4.74	5.12	5.12
MPPI (%)	6.94	4.51	9.88	8.83	4.37	10.38	1.30	13.73	13.73
MPAI (%)	8.46	9.53	9.41	10.63	8.52	11.53	5.43	12.26	12.26

TABLE III Comparison of parameter tuning methods on Rocket benchmark

Method	FIST [9]	DAC'19 [7]	MLCAD'19 [2]	ICCAD'21 [10]	PTPT [11]	REMOTune [1]	DATE'24 [30]	ICCAD'24 [31]	Ours
HV (10^5)	1.47	1.19	1.35	1.50	1.31	1.61	1.36	1.67	1.72
HV _{0,1} (10^3)	3.03	2.79	2.93	3.16	2.85	3.35	2.97	3.45	3.45
HV _{0,2} (10^3)	3.02	2.75	2.94	3.16	2.84	3.18	2.63	3.06	3.15
HV _{1,2} (10^3)	2.42	1.85	2.19	2.23	2.20	2.51	2.40	2.69	2.78
MPI1 (%)	12.38	14.50	13.44	16.72	11.97	16.11	7.34	13.00	12.99
MPI2 (%)	-0.51	-6.70	-2.99	-2.42	-2.83	1.57	2.16	5.09	5.09
MAI (%)	-1.01	-7.25	-3.32	-2.55	-3.39	-1.31	-3.79	-0.96	0.91
MPPI (%)	11.93	8.77	10.85	14.70	9.48	17.43	5.46	13.11	13.56
MPAI (%)	11.50	8.30	10.67	14.60	8.99	15.01	1.02	6.68	6.68

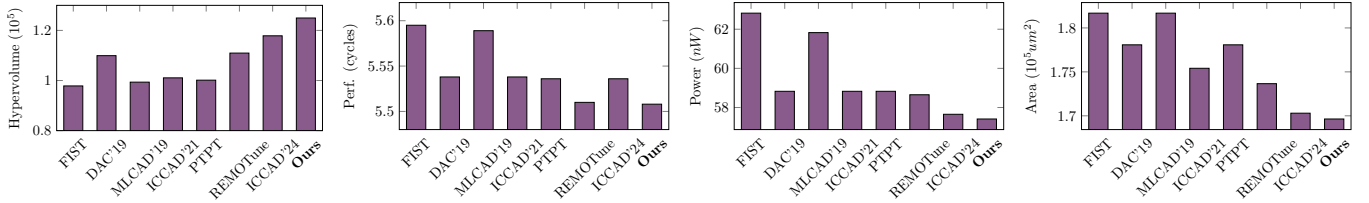


Fig. 6 Comparison of parameter tuning methods on BlackParrot benchmark.

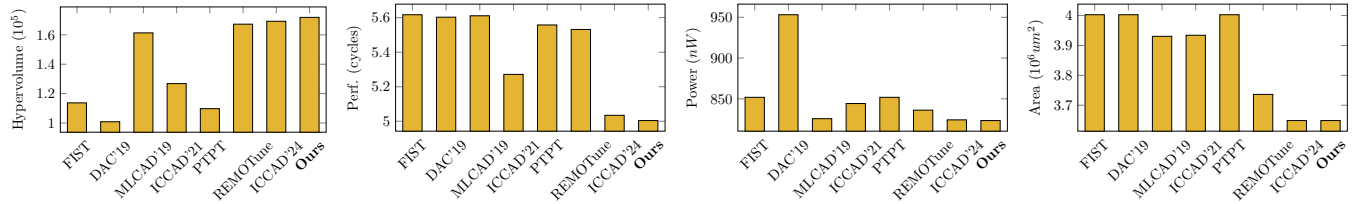


Fig. 7 Comparison of parameter tuning methods on BOOM benchmark.

wire, and timing optimizations. This space can be expanded to other tool options or stages, but higher dimensionality and less impactful parameters raise cost and noise. Our framework could alleviate this via knowledge-guided embeddings and multi-fidelity filtering, though prioritizing influential parameters remains crucial for scalability.

Benchmarks. We primarily utilize our proposed RankTuner for parameter tuning in RISC-V processors (RISC-V32I [32] and Rocket [33]). The technology node is TSMC 65nm. To evaluate the scalability of REMOTune, we also test the methods on BlackParrot (BP) [34] and Berkeley Out-of-Order Machine (BOOM) [35] processors. We aim to showcase the effectiveness of our proposed framework across benchmarks of varying characteristics. The RISC-V32I and Rocket benchmarks consist of 7.6k and 14.2k cells, respectively, where RISC-V32I is implemented by designers using Verilog

and Rocket is generated from Chisel code. This results in significant differences between the designs of RISC-V32I and Rocket, leading to different optimization options required. These differences are particularly evident during the synthesis phase [1]. The BlackParrot [34] is a multicore processor design capable of running the Linux operating system. Therefore, its scale and design complexity are larger than the previous two designs, with 43.2k cells. The BOOM processor is an open-source, synthesizable, out-of-order RISC-V processor core developed as part of the Rocket Chip generator framework, with 3.68m cells.

Compared Methods. The baseline methods compared in this study include: (1) FIST, which leverages the XGBoost ensemble model and importance sampling for parameter adjustment in EDA [9]; (2) DAC'19, which applies tensor decomposition and regression to build a collaborative prediction model for

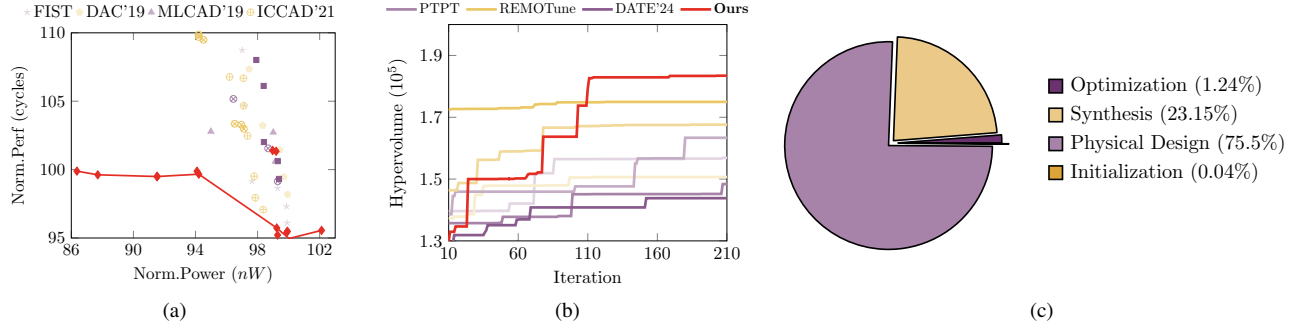


Fig. 8 (a) The learned Pareto optimal set (normalized performance v.s. normalized power consumption); (b) The attained Hypervolume v.s. Iteration; (c) Runtime breakdown for RISCv32I;

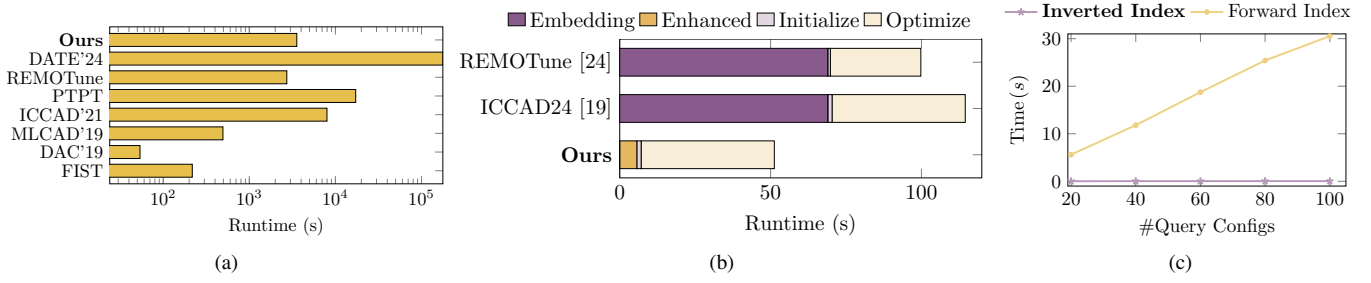


Fig. 9 (a) The Runtime comparison of the tested algorithms; (b) The breakdown of enhanced embedding time on RISCv32I benchmark; (c) The visualization of text embedding for different tool parameters.

parameter tuning [7]; (3) MLCAD'19, which employs a Bayesian framework to explore the parameter space of EDA toolset [2]; (4) ICCAD'21 AutoTuner, an open-source platform that integrates diverse optimization algorithms such as evolutionary search and tree-structured Parzen estimators [10]; (5) PTPT, which applies multi-objective Bayesian optimization to obtain Pareto-optimal design parameters and uses a multi-task Gaussian process to model correlations across objectives [11]; (6) TODEAS'23, a state-of-the-art approach that combines random embedding with multi-objective trust-region Bayesian optimization for guided parameter tuning [1]; and (7) DATE'24, which introduces a hybrid Gaussian process model with attention mechanisms to capture complex interactions between continuous and discrete EDA parameters [30].

Evaluation Metrics. We evaluate parameter tuning methods using their ability to deliver better quality of result (QoR). Similar to [31], we compare parameter tuning methods using several QoR-related metrics: hypervolume (HV), maximum performance improvement (MPI1), maximum power improvement (MPI2), maximum area improvement (MAI), maximum performance–power improvement (MPPI), and maximum performance–area improvement (MPAI). HV measures the size of the objective space dominated by the Pareto frontier relative to a reference point. Formally, it is defined as follows:

$$HV = \int_{\mathbf{y}} \mathbb{I}[\mathbf{y} \succcurlyeq \mathbf{v}_{\text{ref}}] \left[1 - \prod_{\mathbf{y}^* \in \mathcal{P}(\mathbf{y})} \mathbb{I}[\mathbf{y}^* \not\prec \mathbf{y}] \right] d\mathbf{y},$$

where $\mathbb{I}(\cdot)$ is the indicator function, $\mathcal{P}(\mathbf{y})$ is the Pareto frontier,

and the reference point is fixed to $\mathbf{v}_{\text{ref}} = [150.0, 150.0, 150.0]$ following [1].

To measure maximum gains of tuned parameters over the reference results, we use:

$$\text{MPI1} = \frac{C_{\text{ref}} - C_{\text{best}}}{C_{\text{ref}}}, \quad \text{MPI2} = \frac{P_{\text{ref}} - P_{\text{best}}}{P_{\text{ref}}}, \quad \text{MAI} = \frac{A_{\text{ref}} - A_{\text{best}}}{A_{\text{ref}}},$$

$$\text{MPPI} = \frac{(C_{\text{ref}} - C_{\text{best}})(P_{\text{ref}} - P_{\text{best}})}{C_{\text{ref}} \cdot P_{\text{ref}}}, \quad \text{MPAI} = \frac{(C_{\text{ref}} - C_{\text{best}})(A_{\text{ref}} - A_{\text{best}})}{C_{\text{ref}} \cdot A_{\text{ref}}}.$$

Here, C , P , and A denote performance, power, and area. The subscript “ref” refers to the baseline design, and “best” refers to the maximum in the tuned designs. The raw performance metrics are measured with cycles. Across all metrics, higher values indicate stronger improvements. For consistency and fairness, all experiments are conducted under the same environment and settings as [31].

B. Experimental Results

The comparison of parameter tuning methods on benchmarks is shown in TABLE II and TABLE III, respectively. RankTuner consistently achieves the best results, with hypervolume gains of up to 40.34%, including 4.89% and 3.59% improvements over REMOTune on RISCv32I and Rocket, respectively. As illustrated in Fig. 8(a), RankTuner discovers markedly better Pareto-optimal sets, outperforming baselines on HV0, 1 and HV0, 2 while remaining competitive on HV_{1,2}. Beyond higher final scores, RankTuner also shows a steady improvement of the Pareto front over iterations as visualized in Fig. 8(b). Unlike REMOTune, which benefits from strong initialization but stalls, RankTuner continues progressing and

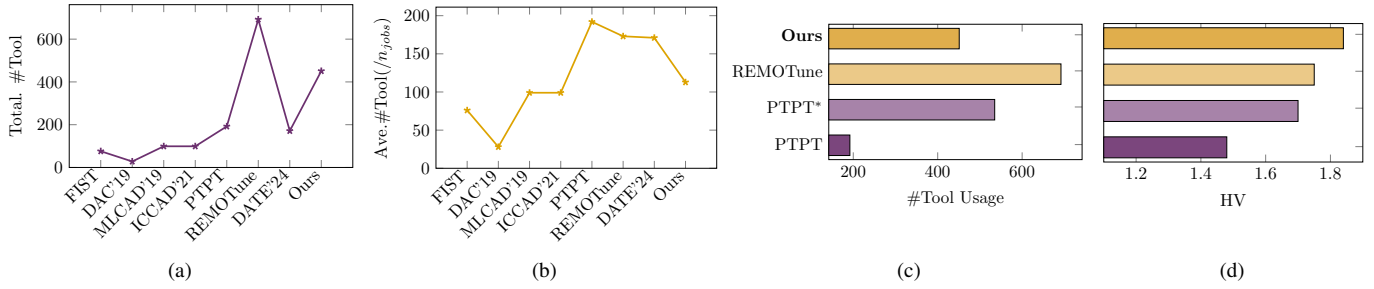


Fig. 10 (a) and (b): Tool usage statistics (total and average). Owing to the parallel exploration strategies adopted by REMOTune and our method, their average tool usage is lower than that of PTPT; (c) and (d): Parallel vs. non-parallel methods, where PTPT* uses the same tool usage yet still underperforms REMOTune and our method, confirming the advantage of parallel exploration.

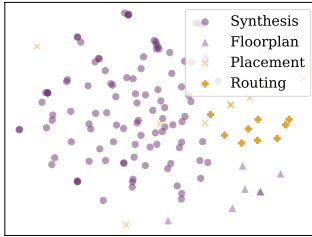


Fig. 11 t-SNE projection of text embeddings, showing design-stage parameters (synthesis, floorplan, placement, routing) clustering by semantic similarity.

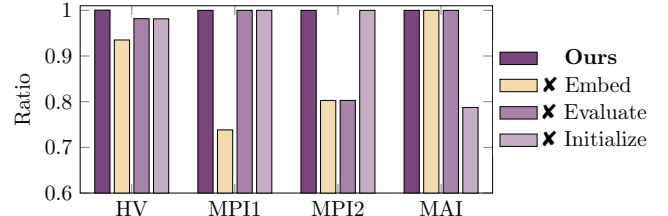


Fig. 12 Ablation study on individual modules.

ultimately surpasses all methods by 100 iterations. These results highlight the robustness of ranking-based modeling in providing reliable guidance and delivering consistently superior QoR outcomes. As shown in Fig. 6, RankTuner delivers the highest hypervolume on BlackParrot, achieving up to 20.45% improvement overall and 5.85% over REMOTune. Although REMOTune and Recommender gain from parallel exploration, their regression-based strategies are less effective under limited iterations. RankTuner's ranking-based modeling, in contrast, captures the objectives more accurately and guides exploration more efficiently, enabling it to outperform competitors on this large benchmark.

Fig. 8(c) and Fig. 9(a) plot the runtime breakdown of our proposed method and comparisons with other methods on RISC-V32I benchmark, respectively. The most consuming part is the EDA Physical Design part, which takes 75.5% of the total runtime. Furthermore, it takes 23.15% of the total time for the EDA Synthesis. The initialization and optimization time only takes about 1.25% in total. The Recommender has the shortest runtime due to its simple neural network structure and simple sampling. FIST and BO are also fast with limited iterations due to their simple surrogate models and acquisition functions. Even though the runtime of our proposed approach is $1.30\times$ slower than REMOTune [1], it is nearly $4.83\times$ faster than PTPT [11] due to the parallel exploration. While RankTuner is not the fastest, it shows significant improvement in constantly exploring the Pareto front.

C. Performance of Knowledge Enhanced Ranktuner

Experimental results (TABLE II, TABLE III, Fig. 9(b)) show that our knowledge-enhanced RankTuner consistently improves performance. On RISC-V32I, it raises overall HV by 4.35% with notable gains across sub-metrics, including up to 44.84% on MPI2. On Rocket, HV improves by 2.99%, and on the more challenging BlackParrot benchmark, by up to 6.03%. These results highlight the advantage of knowledge-enhanced embeddings in capturing parameter correlations and delivering stronger performance, while also replacing the original embedding with a more efficient design.

As shown in Fig. 9(b), KRankTuner achieves substantial runtime gains, with embedding nearly 92% faster than ICCAD'24 [31] and REMOTune [1], and overall inference 55.26% and 48.66% faster than RankTool and REMOTune, respectively. Fig. 9(c) further shows that the inverted index scales efficiently with increasing queries, unlike the linear overhead of the forward index, confirming its effectiveness.

Fig. 11 shows the t-SNE visualization of text embeddings derived from the EDA tool documentation. Each point corresponds to one tool parameter, and the markers/colors represent the design stage it belongs to (synthesis, floorplan, placement, or routing). The t-SNE algorithm projects the high-dimensional embeddings into two dimensions while preserving neighborhood relationships, so parameters with similar functions are located near each other in the plot. Routing-related parameters (yellow) form a distinct cluster, while synthesis, floorplan, and placement (purple) overlap but still group locally. This shows the embeddings capture semantic similarity, with related parameters positioned close together.

D. Ablation Studies

Performance on Larger Processor. To assess scalability, we extend our evaluation to the larger BOOM benchmark. Across all four benchmarks—RISCV32I, Rocket, BlackParrot, and BOOM—our method consistently achieves the highest hypervolume, surpassing the prior best method ICCAD’24. On the BOOM benchmark, as shown in Fig. 7, it follows the same trend as smaller designs, reaching the top HV and improving PPA trade-offs with higher performance and lower area. These results confirm the scalability of our method to complex processor benchmarks.

Tool Usage Comparison Across Methods. As shown in Fig. 10(a) and Fig. 10(b), our method and REMOTune incur more total tool calls due to parallel exploration; however, this strategy enables broader search and yields higher HV improvements. At the same time, both methods achieve significantly lower average tool usage than PTPT, highlighting their efficiency. To further contrast parallel and non-parallel approaches (Fig. 10(c) and Fig. 10(d)), we extended PTPT’s runtime—denoted as PTPT*—to match the tool budget. Even under this setting, PTPT* still lags behind REMOTune and our method with nearly $4\times$ runtime consumption, validating the effectiveness of parallel exploration.

Effectiveness of Each Module. To evaluate the contribution of each proposed module, we conduct ablation experiments by disabling them one at a time (Fig. 12). Specifically, ✕ Embed refers to removing the TF-IDF-based embedding method in the KRankTuner framework (introduced in Section IV); ✕ Evaluate refers to disabling the multi-fidelity evaluation scheme (Section III-D); ✕ Initialize refers to replacing the diversity-aware initialization method (Section III-D) with pure random sampling. Among these, knowledge embedding contributes the most, while multi-fidelity evaluation and initialization provide comparable benefits. These results confirm the importance of all components and underscore that a robust knowledge embedding approach is crucial for ranking-based high-dimensional tool parameter search.

VI. CONCLUSION

In this paper, we propose RankTuner, a novel framework for EDA tool parameter tuning based on estimating the probability and uncertainty of dominating preference between parameters. We introduce a pairwise Gaussian process to compare parameters and output the approximated posterior of the dominating probability and uncertainty. A Duel-Thompson sampling is further utilized to sample informative comparisons. To speed up the exploration process, we integrate random embedding and trust-region techniques into our framework. Experimental results show that RankTuner outperforms state-of-the-art methods in terms of search quality in competitive runtime, with up to 40.34% improvement of hypervolume.

REFERENCES

- [1] S. Zheng, H. Geng, C. Bai, B. Yu, and M. D. Wong, “Boosting vlsi design flow parameter tuning with random embedding and multi-objective trust-region bayesian optimization,” *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 28, no. 5, pp. 1–23, 2023.
- [2] Y. Ma, Z. Yu, and B. Yu, “CAD tool design space exploration via bayesian optimization,” in *ACM/IEEE Workshop on Machine Learning CAD (MLCAD)*, 2019.
- [3] 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 *IEEE/ACM Proceedings Design, Automation and Test in Europe (DATE)*, 2016, pp. 1148–1151.
- [4] M. M. Ziegler, H.-Y. Liu, and L. P. Carloni, “Scalable auto-tuning of synthesis parameters for optimizing high-performance processors,” in *IEEE International Symposium on Low Power Electronics and Design (ISLPED)*, 2016, pp. 180–185.
- [5] 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)*, 2021.
- [6] 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.
- [7] 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.
- [8] 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.
- [9] 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.
- [10] J. Jung, A. B. Kahng, S. Kim, and R. Varadarajan, “METRICS2.1 and flow tuning in the IEEE CEDA robust design flow and OpenROAD,” in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2021.
- [11] 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.
- [12] Q. Sun, T. Chen, S. Liu, J. Chen, H. Yu, and B. Yu, “Correlated multi-objective multi-fidelity optimization for hls directives design,” *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 27, no. 4, pp. 1–27, 2022.
- [13] C. Bai, Q. Sun, J. Zhai, Y. Ma, B. Yu, and M. D. Wong, “Boom-explorer: Risc-v boom microarchitecture design space exploration framework,” in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2021, pp. 1–9.
- [14] S. Huang, Y. Ye, H. Yan, and L. Shi, “Ars-flow: A design space exploration flow for accelerator-rich system based on active learning,” in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 2024, pp. 213–218.
- [15] J. González, Z. Dai, A. Damianou, and N. D. Lawrence, “Preferential bayesian optimization,” in *International Conference on Machine Learning (ICML)*, 2017, pp. 1282–1291.
- [16] Z. J. Lin, R. Astudillo, P. Frazier, and E. Bakshy, “Preference exploration for efficient bayesian optimization with multiple outcomes,” in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2022, pp. 4235–4258.
- [17] R. Astudillo and P. Frazier, “Multi-attribute bayesian optimization with interactive preference learning,” in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020, pp. 4496–4507.
- [18] P. Mikkola, M. Todorović, J. Järvi, P. Rinke, and S. Kaski, “Projective preferential bayesian optimization,” in *International Conference on Machine Learning (ICML)*, 2020, pp. 6884–6892.
- [19] S. Takeno, M. Nomura, and M. Karasuyama, “Towards practical preferential bayesian optimization with skew gaussian processes,” in *International Conference on Machine Learning (ICML)*, 2023, pp. 33 516–33 533.

- [20] W. Chu and Z. Ghahramani, "Preference learning with gaussian processes," in *Proceedings of the 22nd international conference on Machine learning*, 2005, pp. 137–144.
- [21] J. González, Z. Dai, A. Damianou, and N. D. Lawrence, "Preferential bayesian optimization," in *International Conference on Machine Learning (ICML)*, 2017, pp. 1282–1291.
- [22] D. Eriksson, M. Pearce, J. Gardner, R. D. Turner, and M. Poloczek, "Scalable global optimization via local bayesian optimization," in *Annual Conference on Neural Information Processing Systems (NIPS)*, vol. 32, 2019.
- [23] M. Balandat, B. Karrer, D. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy, "Botorch: A framework for efficient monte-carlo bayesian optimization," in *Annual Conference on Neural Information Processing Systems (NIPS)*, vol. 33, 2020, pp. 21 524–21 538.
- [24] J. Yoder and C. E. Priebe, "Semi-supervised k-means++," *Journal of Statistical Computation and Simulation*, vol. 87, no. 13, pp. 2597–2608, 2017.
- [25] A. S. Berahas, J. Nocedal, and M. Takác, "A multi-batch l-bfgs method for machine learning," *Advances in Neural Information Processing Systems*, vol. 29, 2016.
- [26] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.
- [27] Z. Wang, F. Hutter, M. Zoghi, D. Matheson, and N. De Freitas, "Bayesian optimization in a billion dimensions via random embeddings," *Journal of Machine Learning Research (JMLR)*, vol. 55, pp. 361–387, 2016.
- [28] W. Zhang, T. Yoshida, and X. Tang, "A comparative study of tf* idf, lsi and multi-words for text classification," *Journal of Expert Systems with Applications*, vol. 38, no. 3, pp. 2758–2765, 2011.
- [29] Q. Chang, Q. Chen, and X. Wang, "Scaling gaussian rbf kernel width to improve svm classification," in *2005 international conference on neural networks and brain*, vol. 1. IEEE, 2005, pp. 19–22.
- [30] L. Donger, S. Qi, X. Qi, C. Tinghuan, and G. Hao, "Attention-based eda tool parameter explorer: From hybrid parameters to multi-qor metrics," in *IEEE/ACM Proceedings Design, Automation and Test in Europe (DATE)*, 2024.
- [31] P. Xu, S. Zheng, Y. Ye, C. Bai, S. Xu, H. Geng, T.-Y. Ho, and B. Yu, "Ranktuner: When design tool parameter tuning meets preference bayesian optimization," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2024.
- [32] J. E. Stine, R. Ridley, and T.-D. Ene. (2021) Osu datapath/control rv32 single-cycle and pipelined architecture in sv. [Online]. Available: <https://github.com/stineje/osu-riscv>
- [33] K. Asanovic, R. Avizienis, J. Bachrach, S. Beamer, D. Biancolin, C. Celio, H. Cook, D. Dabbelt, J. Hauser, A. Izraelevitz *et al.*, "The rocket chip generator," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17*, vol. 4, 2016.
- [34] D. Petrisko, F. Gilani, M. Wyse, D. C. Jung, S. Davidson, P. Gao, C. Zhao, Z. Azad, S. Canakci, B. Veluri, T. Guarino, A. Joshi, M. Oskin, and M. B. Taylor, "BlackParrot: An agile open-source RISC-V multicore for accelerator SoCs," *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, vol. 40, no. 4, pp. 93–102, 2020.
- [35] J. Zhao, B. Korpan, A. Gonzalez, and K. Asanovic, "SonicBOOM: The 3rd generation Berkeley out-of-order machine," in *Fourth Workshop on Computer Architecture Research with RISC-V*, May 2020.



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 from Fall 2022. Previously, he received his B.S. from Central South University and his M.S. from Harbin Institute of Technology (Shenzhen). 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.



Yuyang Ye received his Ph.D. degree from Southeast University in 2024. He is currently a Postdoc researcher in the Department of Computer Science and Engineering at the Chinese University of Hong Kong. His current research interests include machine learning for EDA, timing analysis and optimization. He received Best Student Paper Award from ICSICT 2022.



Chen Bai received the B.E. degree in software engineering from University of Electronic Science and Technology of China, Chengdu, China, in 2020 and the Ph.D. degree in computer science and engineering from The Chinese University of Hong Kong, Hong Kong, in 2024. He is currently a Postdoctoral Fellow in the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology. His research interests include computer architecture and electronic design automation. He received the William J. McCalla Best Paper Award from ICCAD 2021 and the Best Paper Award Nomination from ISPD 2024.



Siyuan Xu received his Ph.D. from the University of Texas at Dallas. Upon graduation, he joined The MathWorks as a Senior Engineer, leading R&D projects in FPGA-based deep learning systems and HW/SW co-design. He is currently a Senior Researcher at Huawei Noah's Ark Lab, where he focuses on large language models (LLMs) and AI algorithms for the modeling and optimization of next-generation 3D chip physical design, as well as data-driven simulation acceleration. He received a Best Paper Award at DATE 2025.



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 ICCAD'25, DAC'25, DAC'24, and ASPD'2019.



Tsung-Yi Ho (F'24) received the Ph.D. degree in Electrical Engineering from National Taiwan University in 2005. He is currently a Professor in the Department of Computer Science and Engineering, at the Chinese University of Hong Kong (CUHK). He was a recipient of the Best Paper Award at IEEE TCAD in 2015. Currently, he serves as the VP Conference of IEEE CEDA, and the Executive Committee of ASP-DAC and ICCAD. He is a Distinguished Member of ACM and a Fellow of IEEE.



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, ASPD'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.