RankTuner: When Design Tool Parameter Tuning Meets Preference Bayesian Optimization

Peng Xu[1], Su Zheng[1], Yuyang Ye[1], Chen Bai[1], Siyuan Xu[2], Hao Geng[3], Tsung-Yi Ho[1], Bei Yu[1]
1 The Chinese University of Hong Kong, 2 Huawei Noah's Ark Lab, 3 ShanghaiTech University

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.

1 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].

These complex algorithms and optimizations are a mixing 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.

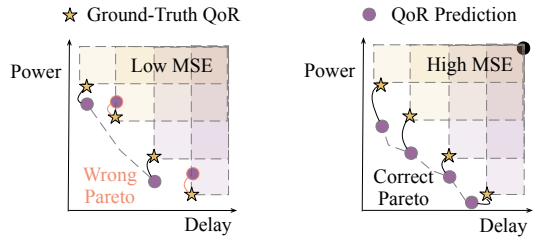


Figure 1: Regression-based method with low MSE loss and wrong Pareto dominance (left) vs. Ranking-based method with high MSE loss and correct Pareto dominance (right).

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. Recommander [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 Figure 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. 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 hypervolume.

2 PRELIMINARIES

2.1 Bayesian Optimization

Bayesian optimization (BO) is an effective technique used for solving global optimization, particularly effective when the evaluation of the objective function is computationally expensive. The core idea of Bayesian optimization is to employ a surrogate mode, typically a Gaussian process (GP), as a prior over the objective function and then use Bayesian inference to guide the search process. Then, data is collected by selecting points in the design space and evaluating the objective function. The parameters of the GP model are updated using the collected data.

A GP model is a non-parametric Bayesian method for function modeling. It is defined over a continuous input space, where any finite collection of joint distributions is Gaussian. 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:

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

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.

An acquisition function will be employed to determine which point to evaluate in the next step. The Expected Improvement (EI) is commonly used to select the next point, which measures the expected value of improvement over the current best value at point $\mathbf{x}$. However, when it comes to parameter comparisons, designing an appropriate acquisition function becomes more challenging. This is because it requires considering the modeling of both the success rate and uncertainty of the dominance relationship.

2.2 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].

2.3 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 $\{1, ..., M\}$, the inequality of objective vectors $f_m(\mathbf{x}_1) \leqslant f_m(\mathbf{x}_2)$ 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:

$$\mathrm{HV}_{\boldsymbol{v}_{\mathrm{ref}}}(\mathcal{P}(\mathcal{Y})) = \int_{\mathcal{Y}} \mathbb{1}\left[\boldsymbol{y} \succcurlyeq \boldsymbol{v}_{\mathrm{ref}}\right] \left[1 - \prod_{\boldsymbol{y}_* \in \mathcal{P}(\mathcal{Y})} \mathbb{1}\left[\boldsymbol{y}_* \not\succeq \boldsymbol{y}\right]\right] \mathrm{d}\boldsymbol{y}, \tag{2}$$

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

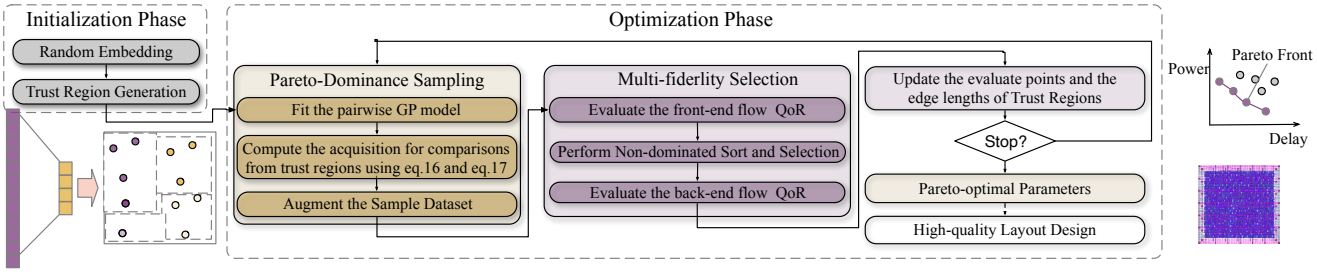


Figure 2: The overall flow of our RankTuner framework.

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 x, and the corresponding QoR y form the objective space $\mathcal{Y}$. For each x, the corresponding QoR metrics $y \in \mathcal{Y}$, 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.

3 RANKTUNER

3.1 Overview

This section introduces our proposed RankTuner framework and provides a detailed explanation of the algorithm. Figure 2 depicts the overall flow of our framework. Our algorithm differs from regression-based parameter tuning methods, which focus on predicting absolute QoR values. Instead, we utilize a pairwise Gaussian process model to predict the probability and uncertainty of the dominance relationship between architectures and leverage it to select the Pareto front.

To accelerate high-dimensional optimization problem exploration, we also employ the techniques of Random Embedding and Trust Region similar to [1]. Due to the time-consuming evaluation process, we also employ front-end QoR results to filter the unpromising parameters.

3.2 The Pairwise Gaussian Process

To model the dominance relationships between parameter configuration 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 2.3, we utilize a pairwise likelihood function defined as:

$$p_{\text{ideal}}(\mathbf{x}_v \succeq \mathbf{x}_u | f(\mathbf{x}_v), f(\mathbf{x}_u)) = \begin{cases} 1 & \text{if } f(\mathbf{x}_v) \geq f(\mathbf{x}_u) \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

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 Figure 3(a).

Then, the pairwise likelihood function could be formulated as:

$$\begin{aligned} \Phi(z_k) =& p\left(\mathbf{x}_v \succeq \mathbf{x}_u \mid f\left(\mathbf{x}_v\right), f\left(\mathbf{x}_u\right)\right), \\ =& \iint p_{\text{ideal}}\ \left(\mathbf{x}_v \succeq \mathbf{x}_k \mid f\left(\mathbf{x}_v\right)+\delta_v, f\left(\mathbf{x}_u\right)+\delta_u\right) \\ & \mathcal{N}\left(\delta_v; 0, \sigma^2\right) \mathcal{N}\left(\delta_u; 0, \sigma^2\right) \mathrm{d}\delta_v \mathrm{d}\delta_u, \end{aligned} \tag{4}$$

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

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 $\boldsymbol{f}_t = [f(\mathbf{x}_r), f(\mathbf{x}_s)]^\top$ have correlations with the n zero-mean random variables of training samples $\{f\left(x_i\right)\}_{i=1}^n$. We compute the prior joint multivariate Gaussian distribution as follows:

$$\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{f}_t \end{bmatrix} \sim \mathcal{N}\left[\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \Sigma & \boldsymbol{k}_t \\ \boldsymbol{k}_t^\top & \Sigma_t \end{pmatrix}\right] \tag{5}$$

where

$$\boldsymbol{k}_t = \begin{bmatrix} \mathcal{K}\left(\mathbf{x}_r, x_1\right), \mathcal{K}\left(\mathbf{x}_r, x_2\right), \ldots, \mathcal{K}\left(\mathbf{x}_r, x_n\right) \\ \mathcal{K}\left(\mathbf{x}_s, x_1\right), \mathcal{K}\left(\mathbf{x}_s, x_2\right), \ldots, \mathcal{K}\left(\mathbf{x}_s, x_n\right) \end{bmatrix}^\top$$

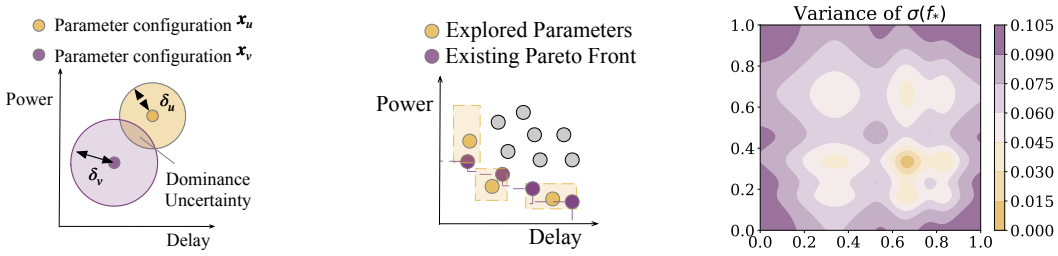
and

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

The predictive distribution of $p\left(f_t \mid \mathcal{D}\right)$ can be computed as an integral over f-space, which can be written as:

$$p\left(\boldsymbol{f}_t \mid \mathcal{D}\right) = \int p\left(\boldsymbol{f}_t \mid \boldsymbol{f}\right) p(\boldsymbol{f} \mid \mathcal{D}) \mathrm{d}\boldsymbol{f}, \tag{6}$$

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 $p(\boldsymbol{f} \mid \mathcal{D})$ as a Gaussian distribution centered on $\boldsymbol{f}_{\text{MAP}}$ with the covariance matrix $\left(\Sigma^{-1} + \Lambda_{\text{MAP}}\right)^{-1}$ using Laplace approximation according to [20].



(a) (b) (c)

Figure 3: The visualization of concepts in RankTuner frameworks: (a) The dominating uncertainty region between two parameter configurations $\mathbf{x}_u$ and $\mathbf{x}_v$; (b) Exploit the existing Pareto front (purple points) vs. Explore the unknown parameters with maximum dominating uncertainty region (golden points); (c) The variance of $\sigma(f_\star)$ decreases in areas with available data, making it suitable for the exploration of comparisons.

The predictive distribution in Equation (6) can then be simplified as a Gaussian distribution $\mathcal{N}\left(f^\star; \mu^\star, \Sigma^\star\right)$ with mean $\mu^\star$ and variance $\Sigma^\star$, where:

$$\mu^\star = \left[\mu_r^\star, \mu_s^\star\right]^\top = \boldsymbol{k}_t^\top \Sigma^{-1} \boldsymbol{f}_{\text{MAP}}, \tag{7}$$

and

$$\Sigma^\star = \begin{bmatrix} \Sigma_{rr}^\star & \Sigma_{rs}^\star \\ \Sigma_{sr}^\star & \Sigma_{rr}^\star \end{bmatrix} = \Sigma_t - \boldsymbol{k}_t^\top \left(\Sigma + \Lambda_{\text{MAP}}^{-1}\right)^{-1} \boldsymbol{k}_t. \tag{8}$$

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

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

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

3.3 Acquisition Function for Pareto-dominance Comparison

After establishing the pairwise Gaussian process model for Pareto dominance comparison pairs, another important question 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.

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 D, we can make inferences about the latent function f and its wrapped version $\pi_{f,\theta}$ using the pairwise Gaussian process in Section 3.2 for Pareto dominance classification:

$$\begin{aligned} \pi_f\left(\left[\mathbf{x}_\star, \mathbf{x}_\star'\right]; \mathcal{D}, \theta\right) &= p\left(y_\star = 1 \mid \mathcal{D}, \left[\mathbf{x}, \mathbf{x}'\right], \theta\right), \\ &= \int \sigma\left(f_\star\right) p\left(f_\star \mid \mathcal{D}, \left[\mathbf{x}_\star, \mathbf{x}_\star'\right], \theta\right) \mathrm{d}f_\star, \end{aligned} \tag{9}$$

where θ represents the parameters of the Pairwise GP.

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 (GP), measured by the variance of $\sigma\left(f_\star\right)$, which is visualized in Figure 3(c). The variance of $\sigma\left(f_\star\right)$ could be mathematically defined as:

$$\begin{aligned} \mathbb{V}\left[\sigma\left(f_\star\right)\right] &= \int \left(\sigma\left(f_\star\right) - \mathbb{E}\left[\sigma\left(f_\star\right)\right]\right)^2 p\left(f_\star \mid \mathcal{D}, \left[\mathbf{x}, \mathbf{x}'\right]\right) \mathrm{d}f_\star \\ &= \int \sigma\left(f_\star\right)^2 p\left(f_\star \mid \mathcal{D}, \left[\mathbf{x}, \mathbf{x}'\right]\right) \mathrm{d}f_\star - \mathbb{E}\left[\sigma\left(f_\star\right)\right]^2, \end{aligned} \tag{10}$$

which explicitly takes into account the uncertainty over 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.

Exploitation. 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 parameters 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:

$$\mathrm{EI}(\mathbf{x}) = \int_{\mathcal{X}} \pi_f\left(\left[\mathbf{x}, \mathbf{x}'\right]\right) \mathrm{d}\mathbf{x}', \tag{11}$$

which is equivalent to randomly sampling parameters from the current Pareto front. The objective is to compute the probability that $\mathbf{x}$ is the dominator in all current comparisons.

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

Table 1: 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
	`wireclength 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, advanced respectively.)

(1) Selecting x: First, generate a sample $\tilde{f}$ from the model using continuous Thompson sampling and compute the associated Expected Improvement score using Equation (11). The first element of the new comparison, $\mathbf{x}_{\text{next}}$, is selected as:

$$\mathbf{x}_{\text{next}} = \arg\max_{\mathbf{x}\in\mathcal{X}} \int_{\mathcal{X}} \pi_{\tilde{f}}\left(\left[\mathbf{x}, \mathbf{x}'\right]\right) \mathrm{d}\mathbf{x}'. \tag{12}$$

As more evaluations are collected, the selection becomes more greedy toward the dominant parameter configuration 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 $\mathbf{x}_{\text{next}}$ as the first element of the selected comparison, the second element is selected as the parameter configuration that maximizes the variance of $\sigma\left(f_\star\right)$ in the direction of $\mathbf{x}_{\text{next}}$. Formally, $\mathbf{x}'_{\text{next}}$ is selected as:

$$\mathbf{x}'_{\text{next}} = \arg\max_{\mathbf{x}'_\star\in\mathcal{X}} \mathbb{V}\left[\sigma\left(f_\star\right) \mid \left[\mathbf{x}_\star, \mathbf{x}'_\star\right], \mathbf{x}_\star = \mathbf{x}_{\text{next}}\right]. \tag{13}$$

This second step is purely explorative in the direction of x_{next}, which aims to find informative comparisons to run with parameters from the current Pareto front.

4 EXPERIMENTS

4.1 Experimental Setting

Parameter Space. Our parameter configuration space is designed based on PTPT and REMOTune, which have demonstrated impressive performance in both academic and industrial settings [1, 11], as shown in Table 1. Each parameter in the search space corresponds to an essential command option in Genus or Innovus, including the synthesis, floorplanning, and routing stage.

Benchmarks. We primarily utilize our proposed RankTuner for parameter tuning in RISC-V processors (`RISCV32I` [23] and `Rocket` [24]). These benchmarks are implemented on TSMC's 65nm technology node. We aim to showcase the effectiveness of our proposed framework across benchmarks of varying characteristics. The `RISCV32I` and `Rocket` benchmarks consist of 7.6k and 14.2k cells respectively, where `RISCV32I` is implemented by designers using Verilog and Rocket is generated from Chisel code. This results in significant differences between the design of `RISCV32I` , leading to different optimization options required. These differences are particularly evident during the synthesis phase [1].

Compared Methods. The baseline methods that will be compared in our study are as follows: 1) FIST [9]: An ensemble tree model (XGBoost) with importance sampling to adjust parameters in the EDA design process. 2) DAC'19 [7]: A tensor decomposition and regression-based collaborative prediction model to reduce parameter tuning effort. 3) MLCAD'19 [2]: A Bayesian approach to explore the parameter space of EDA tools. 4) ICCAD'21 [10]: An open-source platform incorporating various optimization algorithms like evolutionary and tree-structured Parzen estimator. 5) PTPT [11]: A multi-objective Bayesian optimization method that uses a multi-task Gaussian process model to capture correlations among multiple objectives. 6) TODEAS'23 [1]: A state-of-the-art guided design flow parameter tuning approach using random embedding and multi-objective trust-region Bayesian optimization. 7) DATE'24 [22]: An EDA tool parameter explorer based on an attention mechanism and a hybrid space Gaussian process model to capture complex interactions between continuous and discrete parameters.

Evaluation Metrics. The proposed approaches aim to boost the ability to obtain parameter configurations with better QoR. The QoR-related metrics are used to compare the parameter tuning methods as in [1]: 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). The default reference point for hypervolume computation is set at $[150.0, 150.0, 150.0]$. For fairness, we used the same environment and implementation of compared methods as in [1].

4.2 Experiemtnal Results

The comparison of parameter tuning methods on benchmarks is shown in Table 2 and Table 3 respectively. In comparison to existing approaches, RankTuner consistently outperforms them across all benchmarks up to 40.34% improvement of hypervolume, demonstrating its ability to discover better Pareto-optimal sets. RankTuner acquires 4.89% and 3.59% higher hypervolumes than the best baseline method, REMOTuner [1], on `RISCV32I` and `Rocket` benchmarks. We further visualize the Pareto-optimal sets obtained from the baseline methods and RankTuner in `RISCV32I` in Figure 4(a). The results indicate that Rank-DSE outperforms all other baseline methods, showing a significant superiority that leads to a noticeable performance improvement compared to the baselines. RankTuner also outperforms other methods in terms of $\mathrm{HV}_{0,1}$, $\mathrm{HV}_{0,2}$, and obtains competitive $\mathrm{HV}_{1,2}$ with REMOTuner [1], demonstrating its superior ability to effectively explore different parameters with optimal QoR objectives combinations. These results suggest that our ranking-based method can provide more reliable prediction guidance, leading to enhancement in solution quality.

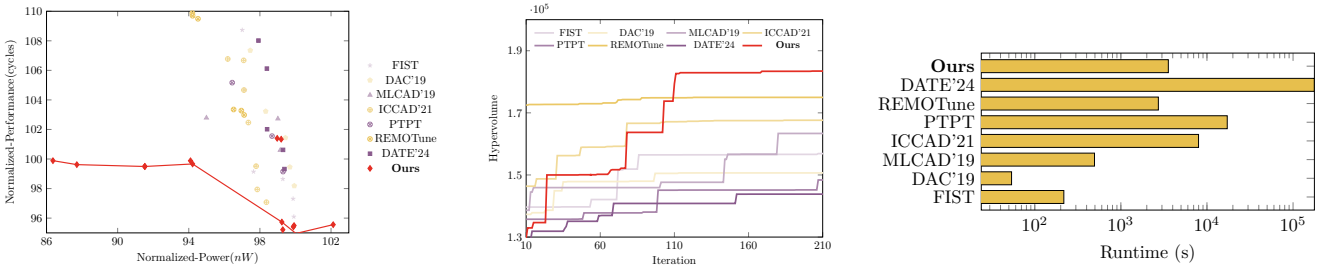
The RankTuner framework also offers a notable advantage in constantly improving the explored Pareto front. This advantage can be verified by referring to Figure 4(b). The figure showcases the obtained hypervolume at different iterations on the `RISCV32I` benchmark. Although having better initial values through initialization, REMOTune does not experience noticeable HV improvement in subsequent exploration. This suggests that regression-based methods often produce inaccurate estimates of the Pareto boundaries, resulting in much exploration wasted.

Table 2: Comparison of Parameter Tuning Methods on `RISCV32I` Benchmark.

Method	FIST [9]	DAC'19 [7]	MLCAD'19 [2]	ICCAD'21 [10]	PTPT [11]	REMOTune [1]	DATE'24 [22]	**Ours**
HV (10^5)	1.57	1.55	1.63	1.68	1.48	1.75	1.44	**1.84**
$HV_{0,1}$ (10^3)	2.85	2.72	3.00	2.95	2.70	3.05	2.63	**3.44**
$HV_{0,2}$ (10^3)	2.94	2.99	3.00	3.07	2.95	3.12	2.84	**3.43**
$HV_{1,2}$ (10^3)	2.97	2.97	3.00	3.14	2.79	**3.23**	2.77	3.00
MPI1(%)	3.16	2.54	5.00	3.81	3.56	4.38	2.08	**13.64**
MPI2(%)	3.90	2.12	5.12	5.23	0.85	**6.27**	0.68	5.04
MAI(%)	5.47	7.18	4.64	7.10	5.15	**7.45**	4.74	5.12
MPPI(%)	6.94	4.51	9.88	8.83	4.37	10.38	1.30	**13.73**
MPAI (%)	8.46	9.53	9.41	10.63	8.52	11.53	5.43	**12.26**

Table 3: 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 [22]	**Ours**
HV (10^5)	1.47	1.19	1.35	1.50	1.31	1.61	1.36	**1.67**
$HV_{0,1}$ (10^3)	3.03	2.79	2.93	3.16	2.85	3.35	2.97	**3.45**
$HV_{0,2}$ (10^3)	3.02	2.75	2.94	3.16	2.84	3.18	2.63	**3.06**
$HV_{1,2}$ (10^3)	2.42	1.85	2.19	2.23	2.20	2.51	2.40	**2.69**
MPI1 (%)	12.38	14.50	13.44	**16.72**	11.97	16.11	7.34	13.00
MPI2 (%)	-0.51	-6.70	-2.99	-2.42	-2.83	**1.57**	2.16	**5.09**
MAI (%)	-1.01	-7.25	-3.32	-2.55	-3.39	-1.31	-3.79	**-0.96**
MPPI (%)	11.93	8.77	10.85	14.70	9.48	**17.43**	5.46	13.11
MPAI (%)	11.50	8.30	10.67	14.60	8.99	**15.01**	1.02	6.68



(a) (b) (c)

Figure 4: (a) The learned Pareto optimal set (normalized performance *v.s.* normalized power consumption); (b) The attained Hypervolume *v.s.* Iteration; (c) The Runtime comparison of the tested algorithms on `RISCV32I`.

Figure 4(c) plots the runtime breakdown of our proposed method and comparisons with other methods on `RISCV32I` benchmark respectively. 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× slower than REMOTune [1], it is nearly 4.83× 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.

5 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 Posteriori 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.

ACKNOWLEDGMENT

This work is partially supported by The Research Grants Council of Hong Kong SAR (No. CUHK14210723 and No. CUHK14211824), and AI Chip Center for Emerging Smart Systems (ACCESS), Hong Kong.

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," 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 *Proc. 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 *Proc. 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 *Proc. 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 *Proc. 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 *Proc. 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 *Proc. DAC*, 2019.

[8] A. Agnesina, K. Chang, and S. K. Lim, "VLSI placement parameter optimization using deep reinforcement learning," in *Proc. 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 *Proc. 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 *Proc. 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 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 *Proc. ASPDAC*, 2024, pp. 213–218.

[15] J. González, Z. Dai, A. Damianou, and N. D. Lawrence, "Preferential bayesian optimization," in *Proc. 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," 2022, pp. 4235–4258.

[17] R. Astudillo and P. Frazier, "Multi-attribute bayesian optimization with interactive preference learning," 2020, pp. 4496–4507.

[18] P. Mikkola, M. Todorović, J. Järvi, P. Rinke, and S. Kaski, "Projective preferential bayesian optimization," in *Proc. ICML*, 2020, pp. 6884–6892.

[19] S. Takeno, M. Nomura, and M. Karasuyama, "Towards practical preferential bayesian optimization with skew gaussian processes," in *Proc. 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 *Proc. ICML*, 2017, pp. 1282–1291.

[22] L. Donger, S. Qi, X. Qi, C. Tinghuan, and G. Hao, "Attention-based eda tool parameter explorer: From hybrid parameters to multi-qor metrics," 2024.

[23] 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

[24] 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.