SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

December 2024, Vol. 67, Iss. 12, 222102:1–222102:17
https://doi.org/10.1007/s11432-023-3963-7

An efficient binary programming method for black-box optimization and its application in processor design

Xiaoliang LV[1], Qiaozhu ZHAI[1*], Jianchen HU[1], Yuhang ZHU[2], Jinhui LIU[1] & Xiaohong GUAN[1,2]

[1]*Faculty of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an* 710049*, China;*
[2]*Department of Automation, Tsinghua University, Beijing* 100084*, China*

Received 1 August 2023/Revised 10 November 2023/Accepted 21 February 2024/Published online 13 December 2024

Abstract Optimizing the parameter settings in a large design space for the processor with limited simulation resources is a challenging task. The current black-box optimization algorithms for processor design space exploration (DSE) problems usually require a large amount of simulation resources for high-dimensional and discrete problems. Besides, the constraints handling techniques in these algorithms need to be improved. To address the issues, we propose an efficient binary integer programming (BIP) approach for the DSE of the processor with strictly guaranteed constraints. Our approach involves adopting the separability assumption to establish a surrogate objective function that is ordinal consistent, thus avoiding the complex non-linearity of the real objective function. Moreover, the design rules can be taken simply as constraints in BIP model to further reduce the design space. Thus, the efforts spent in the infeasible exploration space can be avoided. The experimental results show that the proposed algorithm outperforms the state-of-the-art Bayesian optimization and evolutionary algorithms in terms of exploration efficiency, required simulation points and performance of the recommended points.

Keywords black-box optimization, binary programming, design space exploration, processor design, expensive simulation system

1 Introduction

Microarchitecture design space exploration (DSE) is critical to fully exploit the performance of hardware resources. The integration of multi-core processors and system-on-chip (SoC) has emerged as a prevailing trend in high-performance processor development, leading to increased complexity in processor design [1]. Architects have to explore a vast design space by running a broad range of cycle-accurate, yet time-consuming simulation to find the optimal design scheme. However, the updating frequency of product poses a significant challenge to design time. A longer time-to-market can lead to significant revenue loss for the vendor if the product fails to enter the market window — the period during which the product is likely to achieve the highest sales [2]. Therefore, finding a satisfactory design scheme within a short timeframe has become a pressing problem.

Exploring design schemes, that optimize multiple design metrics (performance, area, etc.) in a large design space composed of many microarchitectural parameters, such as cache set and way, queue size, branch predictor configuration, and pipeline depth, is a challenging task for the processor architects. First, it is difficult to express the optimization objective function in an analytic form for an expensive simulation system. In such cases, the optimization approaches have to interact with expensive cycle-accurate-simulator (CAS) in industrial application. There is typically a degree of discrepancy between simulators and physical devices. The accuracy of CAS improves at the cost of increased evaluation time and computational expense, such as SimpleScalar [3], Asim [4], M5 [5], GEMS [6]. They usually take about dozens of hours to run a benchmark suite. The computational and time cost of simulation tools

* Corresponding author (email: qzzhai2015@163.com)

place high demands on the exploration efficiency of DSE optimization algorithms. Second, it is challenging for most data-driven methods to incorporate design constraints into the model systematically. In this paper, the mixed integer linear programming (MILP) method has been used to model these design rules as constraints. Third, the challenge of multiple design objectives in processor design is to find a balance between them. This requires a trade-off analysis between the different metrics, where designers must identify which objectives are more critical and prioritize them accordingly. In this paper, we mainly consider the performance optimization under the area constraint, which is relatively easy to be formulated.

We focus on the DSE optimization method based on the MILP to reduce the number of iterations and the design schemes that need to be simulated. The knowledge of microarchitecture and manual experience are very valuable for optimization algorithm. They will be fully utilized in our approach. To our best knowledge, the current data-driven DSE approaches for processor require a large amount of simulation data, as discussed in Subsection 2.2, which can be challenging for expensive CAS. Therefore, a DSE optimization method with low dependence on simulation data needs to be proposed for modern processor design.

In this paper, we propose a binary integer programming (BIP) formulation and optimization approach for processor DSE. The biggest challenge for the application of the BIP method is the lack of processor analytical model, which makes it difficult to accurately formulate the optimization objective function. Based on the knowledge of the microarchitecture and the insight of the data analysis, we make the assumption that the effect of microarchitectural parameters on the performance of the processor is approximately separable. Based on the assumption, we successfully constructed the order-preserving surrogate objective function for the BIP DSE model. The experimental results not only demonstrate the effectiveness of the proposed method in reducing the number of iterations required for processor DSE, but also provide evidence to support our assumption. Besides, the proposed method can achieve better design schemes with fewer simulations compared to the baseline method heteroscedastic evolutionary Bayesian optimization (HEBO) [7], evolutionary constrained optimization with hybrid constraint-handling technique (ECO-HCT) [8] and self-adaptive resources allocation-based differential evolution (SRADE) [9].

Our contributions are summarized as follows:

A novel BIP model is developed to enable the attainment of optimal design schemes from an enormous design space with limited simulation overhead.

To facilitate the optimization process for the processor DSE, a surrogate objective function is formulated by fitting the ordinal relationship between the candidates of the parameters and their respective performances.

The exploration effectiveness and efficiency of the proposed approach are validated by applying it to a real-world industrial processor DSE case and 35 black-box benchmark functions.

The remainder of this paper is organized as follows. Section 2 summarizes the processor model and DSE methods in recent years. Section 3 gives the problem formulation of the processor DSE and presents the proposed BIP DSE method based on the surrogate objective. Section 4 reports the experimental setup and the results. The performance of the proposed surrogate objective-based binary programming (SOBP) DSE algorithm is compared with HEBO in engineering DSE case, as well as with HEBO, ECO-HCT and SRADE in benchmark functions. Finally, we summarize this article in Section 5.

2 Related work

In this section, we review the models for processor systems and divide them into three categories: white-box, black-box and gray-box models respectively (Subsection 2.1). Processor systems are typical example of expensive simulation systems, where DSE is widely used to assist designers in comprehending the trade-offs and making informed decisions based on the desired metrics. Therefore, we investigate the DSE optimization methods in various fields in recent years (Subsection 2.2).

2.1 Processor model

White-box model, or mechanistic model, can provide fundamental insight and generate new knowledge potentially [10]. The work in [11] proposed a mechanistic model for out-of-order super-scalar processors by dividing execution time into intervals. The calculation results of the interval model differ from CAS by an average of 7%. The work in [12] presented a microarchitecture independent profiler and associated analytical models that can produce performance and power estimates almost instantaneously. However,

this model has an average performance error of 9.3% in a large design space. In conclusion, while model-based methods are typically faster than simulator-based methods, their limited precision restricts their practical application in the industrial field.

Black-box model, or data-driven model, is constructed based on existing simulation data, employing regression models, neural networks and other methodologies to establish a surrogate model. The work in [13] developed a linear regression model that relates the microarchitectural parameters (along with some of their interactions) to the overall processor performance. It establishes the relationship between the performance (cycle-per-instruction, CPI) and 26 key microarchitectural parameters. However, the performance of the processor is rarely a linear function of the parameters. Nonlinear form mapping must be converted to linear form. Typical transformations are square root, logarithmic, power, etc. [10]. As shown in [14,15], instruction-per-cycle (IPC) is related to the reorder buffer size following an approximate square root relation. Therefore, the work in [16] presented spline-based regression modeling for the non-linearity. With these models, statistical inference can be performed computationally efficiently by simulating just 1 in 5 million points of a joint microarchitecture-application design space, which shows the application prospect of regression method in large scale DSE problems. Meanwhile, this can achieve median error rates as low as 4.1 percent for performance. In order to deal with nonlinearity, the studies in [17,18] explored the idea of using artificial neural networks (ANNs) and radial basis function (RBF) networks respectively to build accurate predictive models for processor performance. The work in [19] introduced a new machine learning model that can predict the performance and energy consumption of any set of programs on any microarchitectural configuration in a fast and accurate manner. This approach is centered on the architecture and leverages prior knowledge obtained from offline training to generalize across benchmarks. The work in [20] proposed an adaptive sampling scheme for a general surrogate model based on the leaving-samples-out (LSO), similar to the cross-validation and inter quartile range (IQR). Inspired by it, the surrogate objective (Subsection 3.2) is constructed in our approach. Black-box models are the most widely studied among the three categories. However, a drawback of using neural networks and regression models is that they require a significant amount of simulation data to train the model, which can be time-consuming and costly in the case of an expensive simulation system. Moreover, these models provide less insight than a mechanistic modeling approach [10].

Gray-box model, or hybrid mechanistic-empirical model, is a combination of the above two types of models, which starts from a performance model based on some insight from the system. The rest of the system, which is unknowns, is fitted using training data, similar to what is done in empirical modeling. The work in [21] proposed a hybrid mechanistic-empirical model for studying optimum pipeline depth. However mechanistic-empirical model for the complete processor system is not yet mature. Therefore, in this paper, we propose a hybrid mechanistic-empirical model for processor system. The optimization objectives are obtained by mechanistic analysis combined with simulation data fitting, which is described detailed in Subsection 3.2. The model is modified by iteratively interacting with the CAS, which overcomes the shortcomings of the other two types of models.

2.2 DSE optimization method

There are many DSE optimization algorithms for the design of specific processor. Due to the evolutionary approach, such as ant colony optimization algorithm, genetic algorithm, and particle swarm optimization algorithm, tends to be stochastic and lacks theoretical guarantees [22], our primary focus lies in investigating approaches that are applicable to engineering problems. The work in [23] presented an automatic DSE framework for register bypasses, named PBExplore. PBExplore can effectively explore multidimensional performance-area-energy trade-off in embedded processor design. Meanwhile, the cheaper (in area and power) design alternatives have been explored with minimal loss of performance. The work in [24] presented a method of acceleration for DSE by classifying knobs and exploring them sequentially, which significantly reduces the exploration space. Meanwhile, they in turn explore knobs with the highest probabilities that lead to new dominating designs. The work in [25] introduced a platform, BOOM-Explorer for the RISC-V register-transfer level architecture DSE, which is the first work introducing an automatic DSE framework to the RISC-V community. Researchers can quickly instantiate complete working RISC-V multi-core systems with synthesizable RTL on the platform. The experimental results indicate that BOOM-Explorer surpasses other baseline methods in terms of the average distance to the reference set.

In the past two decades, the DSE problem of universal processor systems has received extensive attention. The work in [26] used the interval model, proposed in [11], to explore the processor design space

automatically and identify the processor configurations that represent Pareto-optimal design schemes with respect to performance, energy and chip area for a particular application or a set of applications. They validated the efficiency of the method in a design space containing two thousand points. However, obtaining the required model terms can be difficult in real-world processor design. In [27], the authors proposed a novel approach for DSE that combines Latin hypercube sampling for initial dataset construction with a semi-supervised ensemble learning technique called SemiBoost. By integrating AdaBoost and semi-supervised learning technology, SemiBoost can generate a more robust and accurate predictive model. Experimental results show that the proposed method outperforms the random sampling strategy and traditional supervised learning methods by significantly reducing the number of simulations required for DSE. However, it should be noted that the 1/10000th simulation cost may be challenging to accept in real industrial scenarios. For multicore processors, the work in [2] proposed a methodology that combines exhaustive searching, greedy searching and one-shot searching, to prune the design space to identify the best settings. Initially, one-shot searching determines the initial settings for each tunable parameter, and a set partitioning algorithm is used to divide the parameters into three groups based on their significance, with each group assigned to one of the three exploration methods. The experimental results demonstrate that the methodology yields solution quality within 1.3%–3.69% of fully exhaustive search results while exploring only 2.74%–3% of the design space. For the same problem, the method was only verified in the design space containing 57600 points. To our best knowledge, the proportion of simulation points mentioned in the literature makes most of the methods challenging to start and apply directly in real industrial scenarios, such as the scenario described in Subsection 4.2. Therefore, there is an urgent need for DSE methods that can effectively handle very large scale design spaces in processor design.

Pure-exploration bandit optimization is also a viable approach that can be adapted to address black-box optimization problems. The work in [28] proposed HYPERBAND, leverages a systematic early-stopping strategy to allocate resources effectively. However, HYPERBAND may encounter a bottleneck due to the high probability of randomly selected parameter configurations being of low quality in the SOC DSE problem. The work in [29] proposed a two-layer algorithm where a thoughtfully selected Meta-algorithm utilizes misspecified bandit algorithms as arms. The work in [30] proposed novel algorithms GRUB (GRaph based UcB) and ζ-GRUB for the pure exploration problem in stochastic multi-armed bandits. However, the methods in [29, 30] assume smoothness of the reward function, which is hard to guarantee in discrete DSE problem.

The BO algorithm is a popular approach used for optimizing device sizing in automated circuit design. The work in [31] proposed weighted expected improvement based on the Bayesian optimization (WEIBO) approach, which is a BO framework with constraints for analog circuits optimization. WEIBO achieves better optimization results with significantly less number of simulations compared with the sequential quadratic programming (SQP) local search, the differential evolution algorithm, the particle swarm intelligence algorithm, and the simulated annealing algorithm. The latest research in [32] proposed an efficient parallelizable Bayesian optimization algorithm via multi-objective acquisition function ensemble, which outperforms WEIBO [31], GASPAD [33], MSP [34], DE [35], PSO [36], and SA [37] in solving the constrained optimization problems. The algorithm's low start-up costs and ability to use only dozens of simulation points in a design space containing 36 design variables make it an attractive solution for solving very large scale processor DSE problems. HEBO [7] represents a recent advancement in Bayesian optimization algorithms, where the author addresses the challenges and limitations associated with data input, data output, surrogate models, and acquisition functions in classical Bayesian optimization. The optimizations implemented in each step include the transformation and calibration of data input and output, joint optimization of data transformation and calibration with Gaussian process kernel functions, and the introduction of multi-objective acquisition functions to enable more robust exploration. Based on the outstanding performance of HEBO in the NeurIPS 2020 Black-Box Optimization challenge and QQ browser 2021AI algorithm competition, we have selected it as a baseline for comparison with our proposed method in experiments (Subsection 4.3).

The work in [38] proposed a batch BO technique called the parallel knowledge gradient method. Experimental results demonstrate the superior efficiency of the parallel knowledge gradient method compared with MOE-qEI [39], Spearmint-qEI [40], the parallel UCB algorithm (GP-BUCB) and parallel UCB with pure exploration (GP-UCB-PE) [39], Spearmint-qEI [40], on both synthetic test functions and tuning hyperparameters of practical machine learning algorithms. However, the experiments in [38] were limited to problems with dimensions up to 6, and the efficacy of the algorithm remains uncertain when dealing with SOC DSE problems approximately ten times larger in dimensionality.

3 Proposed model and approach

In this section, we present the proposed formulation and optimization approach based on BIP for processor DSE. The problem formulation and the BIP model for processor DSE are presented in Subsection 3.1. In order to provide the simple enough objective function for the BIP model, we propose the surrogate objective function acquisition model, which takes advantage of the microarchitectural mechanism and expert experience and significantly reduces the dependence on simulation data (Subsection 3.2). Finally, we propose SOBP optimization algorithm to improve the accuracy of the surrogate objective by iterating with the CAS (Subsection 3.3).

3.1 Binary programming DSE model construction

In the initial design stage of the processor, performance and area are mainly considered. The area is mainly determined by parameters, module selection and layout. The area can be calculated directly by the values of the parameter. In other words, it can be treated as a constraint in the DSE model. The performance is influenced by the microarchitectural parameters, program quality, and input data. However, program quality and input data are not controllable during the design phase. The benchmark suites are used to substitute for actual applications. Moreover, microarchitectural parameters have the most significant impact on the overall performance of the processor. Therefore, we ignore the uncertainty of the program as well as the input data to simplify the problem. The model can be formulated as

$$\begin{aligned} &\max_{x} F(x) \\ &\text{s.t.} \begin{cases} g^{\text{Sys}}(x) \leqslant 0, \\ g^{\text{Cus}}(x) \leqslant 0, \\ g^{\text{Area}}(x) \leqslant 0, \end{cases} \end{aligned} \tag{1}$$

where x is the design scheme composed of parameter values and $x = \{x_1, x_2, \ldots, x_n\}$, where n is the number of parameters; $F(x)$ represents the functional relationship between processor performance and microarchitectural parameters; $g^{\text{Sys}}(x)$ includes parameter value range, coupling relationship between parameters and special design requirements of some dedicated processors; $g^{\text{Cus}}(x)$ represents the special requirements of customers, such as the lower bound of cache size and memory size; $g^{\text{Area}}(x)$ represents the pre-defined limit of total areas that should not be exceeded.

The main challenge in (1) is that $F(x)$ lacks clear analytic expressions. In Subsection 3.2, we will show how to acquire substitute expressions. In addition, the large size of the design space makes it impossible to search exhaustively. Heuristic acceleration methods, such as random traversal, genetic algorithms, simulated annealing, and tabu search algorithms, cannot take full advantage of design rules to prune the design space and converge to the near-optimal solution with limited simulation resources [10]. Meanwhile, they are more likely to get stuck in local optimal, and they do not provide insight into the fundamentals that support optimal choices [26]. However, expensive simulators and time-to-market require DSE optimization algorithms to converge quickly. MILP can be used to address these issues as long as a sufficiently accurate optimization objective function is provided. Besides, the candidates of parameters are mainly discrete integer variables and the microarchitectural design is to select a certain combination of parameter values from the candidate sets. Based on the problem structure, we choose the BIP model as the basis, expressed as

$$\begin{aligned} &\max_{y} \phi(y) \\ &\text{s.t.} \begin{cases} Ay \leqslant b, \\ y = \{0,1\}^l, \end{cases} \end{aligned} \tag{2}$$

where y represents an l-dimensional vector of binary decision variables, whose values are 0 or 1, and $\phi(y)$ represents the objective function.

The microarchitectural parameter set is defined as $P = \{p_1, p_2, \ldots, p_n\}$, where n is the number of parameters and the candidate set of the i-th parameter is $p_i = \{p_i^{(k)} | k = 1, 2, \ldots, m_i\}$, where $p_i^{(k)}$ is the k-th candidates of the i-th parameter and m_i represents the number of candidates of the i-th parameter. Based on the expert experience, the order of significance for the impact on performance is almost consistent. Therefore, we sort the candidate sets in ascending order based on their value. In order

to model the DSE problem as a BIP model, we define an auxiliary binary variable $y_i^{(k)}$ for each candidate of each parameter. And then, the i-th parameter can be expressed as the sum of the product of all its candidates and their corresponding binary variables, i.e.,

$$\begin{aligned} &x_i = \sum_{k=1}^{m_i} p_i^{(k)} y_i^{(k)}, \\ &\sum_{k=1}^{m_i} y_i^{(k)} = 1, \\ &y_i^{(k)} \in \{0,1\}, k = 1,2,\ldots,m_i. \end{aligned} \tag{3}$$

The second equation in (3) ensures that only one candidate can be taken from the candidate set, named unique constraint. The advantage of this transformation is that it allows for easy mapping of nonlinear forms of design parameters, such as the square, square root, logarithm, and exponential transformations, as shown in

$$\begin{aligned} &\log_2 x_i = y_i^{(1)} \log_2 p_i^{(1)} + y_i^{(2)} \log_2 p_i^{(2)} + \cdots + y_i^{(m_i)} \log_2 p_i^{(m_i)}, \\ &x_i^2 = y_i^{(1)} \left[p_i^{(1)}\right]^2 + y_i^{(2)} \left[p_i^{(2)}\right]^2 + \cdots + y_i^{(m_i)} \left[p_i^{(m_i)}\right]^2, \\ &\text{s.t.} \begin{cases} y_i^{(1)}, y_i^{(2)}, \ldots, y_i^{(m_i)} \in \{0,1\}^{m_i}, \\ i = 1,2,\ldots,n. \end{cases} \end{aligned} \tag{4}$$

After the variable transformation, a unique constraint has been defined in the model. Based on the variable transformation in (3), Eq. (1) can be reformulated as

$$\begin{aligned} &\max_y \tilde{F}(y) \\ &\text{s.t.} \begin{cases} \tilde{G}(y) \leqslant 0, \\ \sum_{k=1}^{m_i} y_i^{(k)} = 1, i = 1,2,\ldots,n, \\ y_i^{(k)} \in \{0,1\}, k = 1,2,\ldots,m_i, \end{cases} \end{aligned} \tag{5}$$

where $\tilde{F}(y)$ represents the objective function after the variable transformation and the decision variables are no longer the original candidates of design parameters, but binary variables that represent whether a candidate is included or excluded in the design schemes. $\tilde{G}(y)$ denotes other constraints after the variable transformation. Most constraints are the size relationship between two variables, such as $x_z \leqslant x_v, z, v \in \{1,2,\ldots,n\}^2$. After conversion to the standard form and the variable transformation, the constraints can be expressed as

$$\sum_{k=1}^{m_z} p_z^{(k)} y_z^{(k)} - \sum_{k=1}^{m_v} p_v^{(k)} y_v^{(k)} \leqslant 0. \tag{6}$$

A customer constraint is usually a restriction on the value range of a microarchitectural parameter based on the original set of candidates. These constraints can be added into the model easily to effectively prune design space and increase the speed of exploration. In summary, we construct the BIP model for the DSE problem as

$$\begin{aligned} &\max_y \tilde{F}(y) \\ &\text{s.t.} \begin{cases} \sum_{i=1}^{n} \sum_{k=1}^{m_i} y_i^{(k)} a_i^{(k)} - A \leqslant 0, \\ \sum_{k=1}^{m_z} p_z^{(k)} y_z^{(k)} - \sum_{k=1}^{m_v} p_v^{(k)} y_v^{(k)} \leqslant 0, \\ \sum_{k=1}^{m_i} y_i^{(k)} = 1, i = 1,2,\ldots,n, \\ y_i^{(k)} \in \{0,1\}, k = 1,2,\ldots,m_i, \end{cases} \end{aligned} \tag{7}$$

where y is the set of decision variable $y_i^{(k)}$, that represents corresponding binary variable for k-th candidate of i-th parameter; $a_i^{(k)}$ is the corresponding area of the k-th candidate of the i-th parameter; A is the predefined total area limit; z, v are numbers of coupling parameters from system constraints or customer constraints, such as (6). The acquisition method of the objective function $\tilde{F}(y)$ will be introduced in Subsection 3.2. So far, the DSE problem for the processor has been successfully modeled as a BIP model, which can be solved by commercial solvers, such as Cplex, Gurobi, or open source solvers, such as CBC. However, since the solution is a set composed of a series of binary variables, we need to substitute binary variables into (3) to get the design schemes, which will be evaluated by CAS.

3.2 Surrogate objective function acquisition method

It is well known that the functional relationship between the performance and the microarchitectural parameters is complicated and nonlinearity [16–19]. Additionally, the coupling relationships between parameters make it difficult to formulate the DSE objective directly. However, based on the architect experience and data analysis, we innovatively propose Assumption 1 as follows to simplify the DSE objective function.

Assumption 1. For processor, the effects of microarchitectural parameters on performance is almost separable in the sense of order preservation.

To better illustrate the fundamental concept of Assumption 1, we utilize two-dimensional functions as toy examples. Let $f_1(u_1, u_2) = \mathrm{e}^{u_1^2+u_2^2}$ represent the real function, where the effects of $u = \{u_1, u_2\}$ is inseparable. However, in the sense of order preservation, we can construct the surrogate function $h_1(u_1, u_2) = u_1^2 + u_2^2$, where the effects of u_1, u_2 are separable, to guide the exploration. It can be denoted as $f_1(u) \overset{\mathrm{OP}}{\Longleftrightarrow} h_1(u)$, where OP represents order-preserving. Similarly, $f_2(u) = u_1^{\frac{2}{3}} \times u_2^3$ and $h_2(u) = \frac{2}{3}\log u_1 + 3\log u_2$ can be denoted as $f_2(u) \overset{\mathrm{OP}}{\Longleftrightarrow} h_2(u)$. In other words, $f_1(u_1, u_2)$ and $f_2(u_1, u_2)$ are separable in the sense of order preservation. Obviously, higher-dimensional examples exist as well.

Besides, a linear regression model is proposed in [13], where the performance of a processor can be represented as a sum of k terms expressed in a generic form as

$$\mathrm{Per} = \beta_0 + \beta_1 x_{i^1} + \beta_2 x_{i^2} + \cdots + \beta_{k-1} x_{i^{k-1}} + \varepsilon, \tag{8}$$

where Per is performance or its transformations, β are weights, and each x_{i^j} is a distinct term that can be the interaction of single parameter, two parameters, three parameters or any higher order. Their experiment results confirm that the constructed linear models can be used to predict the response at parameters and interaction terms. It is noted that the effect from the term of a single parameter is more significant than higher order. This work provides theoretical support for the validity of Assumption 1. It is important to note that Assumption 1, which is required for the proposed method in this paper, is more lenient than the linearity assumption in [13]. This is because Assumption 1 only requires separability in the sense of order preservation, while the linearity assumption is a stricter condition.

Based on Assumption 1, a separability function is proposed to be used as a surrogate objective. In Figure 1, we take a two-dimensional continuous function as an example to illustrate the core idea of the surrogate objective acquisition method. The blue curve in the figure denotes the real functional relationship between response and inputs, which is inseparable and difficult to model directly. The red curve is separable and much simpler than the blue curve. Although the value of response has no direct relationship, the ordinal relationship is consistent between them. In other words, in Figure 1, $f(d) \overset{\mathrm{OP}}{\Longleftrightarrow} h(d)$. Thus, if the surrogate objective does not change the ordinal relationship between the performance of any two design schemes, the optimal solution of the surrogate objective will be consistent with the real objective. The relevant theorem and proof are as follows.

Theorem 1. If the ordinal relationship between the surrogate objective function and the real objective function is consistent, then the optimal solution of the surrogate objective function will also be the optimal solution of the real objective function.

Proof. Let $f(d)$ represent the real objective function and $h(d)$ represent the surrogate objective function. As an illustrative example, we will consider an unconstrained maximization problem, i.e.,

$$\max f(d), \tag{9}$$

where $d \in D$, where D represents the design space, and

$$\forall d_i, d_j \in D, f(d_i) \geqslant f(d_j) \Leftrightarrow h(d_i) \geqslant h(d_j), \tag{10}$$

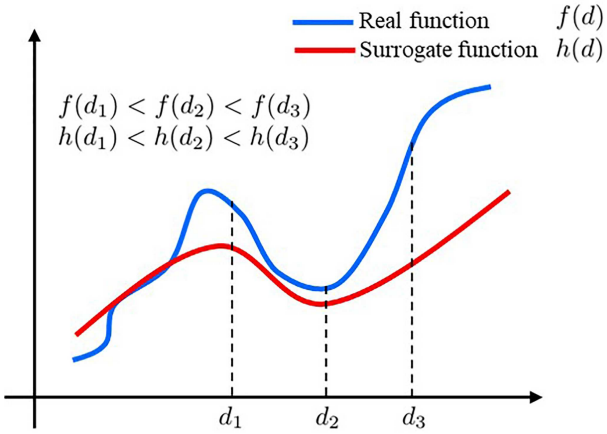

Figure 1 (Color online) Schematic diagram of surrogate objective in the sense of order preservation. Any two points have the same ordinal relationship between the surrogate function and the real function.

if we can find the optimal solution d_{opt} of surrogate objective function, i.e.,

$$\exists d_{\text{opt}} \in D, \forall d_k \in D, h\left(d_{\text{opt}}\right) \geqslant h\left(d_k\right). \tag{11}$$

Obviously, combined with (10), we have

$$\forall d_k \in D, f\left(d_{\text{opt}}\right) \geqslant f\left(d_k\right). \tag{12}$$

Thus, d_{opt} is the optimal solution of (9).

For processor DSE, the real functional relationship between the performance and the microarchitectural parameters is complicated and nonlinear, which is difficult to obtain directly based on the processor operating mechanism. Thus, we use the surrogate objective function to approximate the real objective function. Based on Assumption 1, we have

$$F(x_1, x_2, \ldots, x_n) \overset{\text{OP}}{\Longleftrightarrow} H_1(x_1) + H_2(x_2) + \cdots + H_n(x_n), \tag{13}$$

where $F(x_1, x_2, \ldots, x_n)$ represents the real objective function; $H_i(x_i)$ is the separable item of surrogate objective function; OP represents order-preserving. The number of separable terms may be less than n due to the combination of strongly coupled parameters, such as cache set and cache way, which will be described in detail in Subsection 4.2. However, we still use n for the sake of description. Benefit by the variable transformation in (3), we can further simplify the surrogate objective function as

$$\begin{aligned} H_i(x_i) &= H_i\left(p_i^{(1)} y_i^{(1)} + p_i^{(2)} y_i^{(2)} + \cdots + p_i^{(m_i)} y_i^{(m_i)}\right) \\ &= \sum_{k=1}^{m_i} y_i^{(k)} H_i(p_i^{(k)}),\ i = 1, 2, \ldots, n, \end{aligned} \tag{14}$$

where $H_i(p_i^{(k)})$ denotes the weight of the k-th candidate of the i-th parameters. Because the decision variables $y_i^{(k)}$ is 0 or 1, it can be extracted from the expression. Even so, it is difficult to obtain the exact value of $H_i(p_i^{(k)})$ directly. By fitting the functional relationship between the order of microarchitectural parameters' candidates and the performance, the proposed method can effectively solve nonlinear problems using a BIP model with greater accuracy and efficiency than traditional data-driven methods that simply fit the relationship between parameters and performance.

The acquisition method offers a significant advantage by requiring less simulation data compared to traditional data-driven methods, which is particularly crucial for expensive simulation systems. We define simulation dataset as $\{\mathbb{X}, \mathbb{P}\}$,where $\mathbb{X} = \{X_r \,|r = 1, 2, \ldots, R\}$ and $\mathbb{P} = \{P_r \,|r = 1, 2, \ldots, R\}$. X_r is a design scheme and the corresponding performance is P_r, which is the index of performance from the CAS or its algebraic transformation for linearization. We make the following assumptions for the initial dataset in order to start the algorithm smoothly.

Assumption 2. The volume of the initial simulation dataset is more than the total number of candidates of all parameters, i.e., $\sum_{i=1}^{n} m_i < R$.

This assumption can be easily satisfied in the context of processor DSE. During the initial design phase, architects often suggest certain configurations for simulation based on their experience. Furthermore, Latin hypercube sampling, orthogonal arrays, and other statistical sampling methods can be employed to generate an initial simulation dataset. In the most basic and crude scenario, randomly generating $\sum_{i=1}^{n} m_i$ design schemes for simulation would not result in a significant waste of resources.

Before constructing the acquisition model of the surrogate objective function, we transform design schemes in the dataset into the corresponding binary variables based on (15). This transformation reduces the impact of the complexity of parameter values on the model complexity.

$$\begin{aligned} y_{r,i}^{(k)} &= 1 \Leftarrow x_{r,i} = p_i^{(k)}, \\ y_{r,i}^{(k)} &= 0 \Leftarrow x_{r,i} \neq p_i^{(k)}, \\ k = 1, 2, \ldots, m_i,\ r &= 1, 2, \ldots, R,\ i = 1, 2, \ldots, n, \end{aligned} \tag{15}$$

where $x_{r,i}$ represents the value of the i-th parameter of the r-th simulation data. Based on the Assumptions 1 and 2, we propose the surrogate objective acquisition model, expressed as (16), to acquire the approximate value $c_i^{(k)}$ of $H_i(p_i^{(k)})$ in (14).

$$\begin{aligned} &\min_{c} \sum_{r=1}^{R} \epsilon_r^+ + \epsilon_r^- \\ &\text{s.t.} \begin{cases} \sum_{i=1}^{n} \sum_{k=1}^{m_i} c_i^{(k)} y_{r,i}^{(k)} + \epsilon_r^+ - \epsilon_r^- = P_r, \\ c_i^{(1)} \geqslant 0, i = 1, 2, \ldots, n, \\ c_i^{(q+1)} \geqslant c_i^{(q)} + \delta, q = 1, 2, \ldots, m_i - 1, \\ \epsilon_r^+ \geqslant 0, r = 1, 2, \ldots, R, \\ \epsilon_r^- \geqslant 0, r = 1, 2, \ldots, R, \end{cases} \end{aligned} \tag{16}$$

where c is the set of decision variables $c_i^{(k)}$, that is the surrogate weight of the k-th candidate of the i-th parameter; ϵ_r^+ and ϵ_r^- are the error of ordinal fitting and the optimization objective is to minimize the sum of the absolute value of errors; R is the number of simulation data in the dataset; P_r is the index of performance from the CAS or its algebraic transformation for linearization; δ is a pre-defined tiny positive number for order-preserving. The second and third constraints in (16) limit the sequence of surrogate weights, which is helpful in reducing data dependency. Based on the surrogate weights, the surrogate objective function of $\max_y \tilde{F}(y)$ in (7) can be expressed as $\max_y \sum_{i=1}^{n} \sum_{k=1}^{m_i} y_i^{(k)} c_i^{(k)}$. Now that we have established the DSE BIP model, and the next step is to focus on improving the precision of surrogate weights through an interactive iteration process with the CAS.

3.3 Surrogate objective-based binary programming DSE algorithm

The optimal solution based on the surrogate function is also the optimal solution of the real problem as long as the order relationship between the surrogate function and the real function is consistent. However, the accuracy of surrogate objective depends on the quality and quantity of the dataset. In addition, due to the error between the CAS and the tape-out, it is more efficient to find a batch of good enough solutions than the optimal [41]. It can be accomplished by adding new constrains to (7) and removing previous optimal solutions from design space. The solution of (16) provides the objective function for (7). And the solution of (7) with simulation results expands the dataset for (16). In order to address the above issues, we propose an efficient SOBP DSE algorithm, the flow of which is summarized in Figure 2.

The implementation details of SOBP DSE algorithm are outlined in Algorithm 1, which can improve the accuracy of the surrogate objective function by iteratively interacting with the simulator. As the exploration space expands, the disparity in order consistency between the surrogate function and the actual function diminishes gradually. In Algorithm 1, T is the maximum number of iterations and S is the number of recommended schemes per round, which is not greater than the parallelism of the simulator.

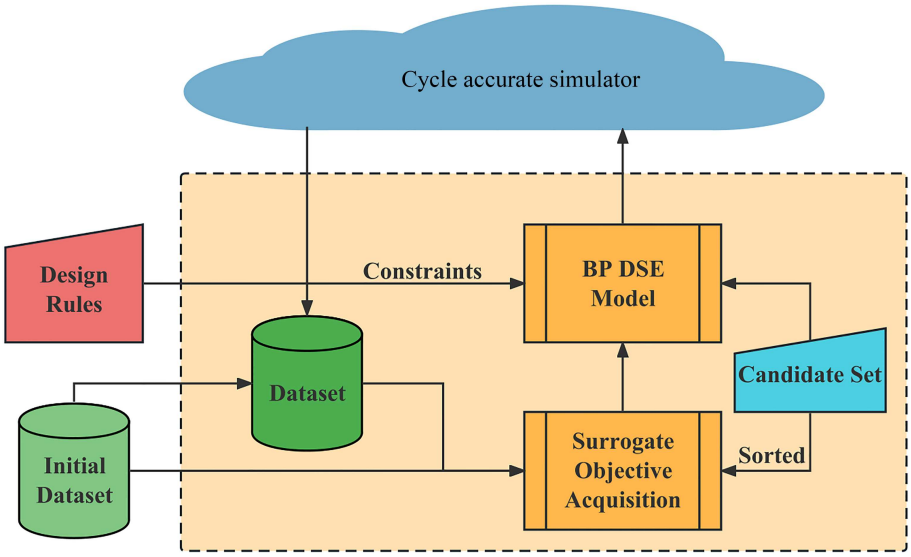

Figure 2 (Color online) Framework of SOBP DSE algorithm.

Algorithm 1 SOBP DSE algorithm.

Require: Initializing a set of training data $\{\mathbb{X}, \mathbb{P}\}$;
  for $t = 1$ to T **do**
    Acquiring surrogate objective by solving (16) based on the current dataset $\{\mathbb{X}, \mathbb{P}\}$;
    $\boldsymbol{X}_t = \emptyset$;
    for $s = 1$ to S **do**
      Eliminating design schemes in $\boldsymbol{X}_t$ and $\mathbb{X}$ from design space by adding constraints to (7);
      Solving (7) with above constraints and substitute solutions into (3) to get the current optimal design schemes $\boldsymbol{x}_s$;
      Adding $\boldsymbol{x}_s$ to the set $\boldsymbol{X}_t$;
    end for
    Simulating the design schemes in $\boldsymbol{X}_t$ and get their performances P_t;
    Adding $(\boldsymbol{X}_t, P_t)$ to the dataset $\{\mathbb{X}, \mathbb{P}\}$;
    if convergence condition **then**
      Break;
    end if
  end for
  return The best design scheme in $\{\mathbb{X}, \mathbb{P}\}$.

Firstly, we construct the surrogate objective acquisition model based on the initial dataset. Then, we maximize the surrogate function to recommend the best S schemes in $\boldsymbol{X}_t$. The design schemes in $\boldsymbol{X}_t$ are simulated and added to the dataset to update surrogate weights. If the optimal design is satisfactory or simulation results get no improvement for several consecutive rounds, the algorithm converges. Otherwise, the optimization continues until convergence or the simulation budget is exhausted.

4 Experiment setup and result analysis

In this section, we describe the processor and benchmark suites used in the experiments (Subsection 4.1) and list the microarchitectural parameters and the design rules (Subsection 4.2). Finally, we demonstrate the effectiveness and efficiency of the SOBP DSE optimization algorithm by comparing it with the state-of-the-art BO algorithm (Subsection 4.3).

4.1 Processor and benchmark suites

We select a real-world DSE case for the experiments to effectively showcase the benefits and practical applications of the SOBP DSE algorithm in industrial engineering. The processor adopts the Berkeley out-of-order machine (BOOM) architecture, whose architecture schematic is presented in Figure A1 in Appendix A. BOOM is industry-competitive in low-power, embedded application scenario [42]. It is an open-source superscalar out-of-order RISC-V architecture and is popular in the field of academic research [25, 43]. Consisting of five main parametric modules: IFU, ROU, EU, LSU and L2C, BOOM can execute benchmarks with distinct behaviors via choosing different candidates for each component

inside these modules. IFU, instruction fetch unit, is the first module for fetching instructions from L2 cache and branch predictors. ROU, reorder unit, consists of decoder module and related logic. It is the dividing line between in-order instruction and out-of-order μops, which is dispatched to execute unit (EU). LSU, load store unit, is the module for loading data and storing data after getting calculation results from EU. In this work, we separate the cache of the next hierarchy into an independent module, L2C, in order to analyze the effect of L2 cache. We employ an industrial-grade CAS provided by the cooperation partner to interact with the DSE optimization algorithms. The error between the simulator and tape-out is less than 1%. However, simulating a round on a cluster of servers takes 1–2 days and the maximum parallelism of the cluster is 50. Therefore, an effective DSE optimization algorithm needs to significantly reduce both simulation and time cost and take full advantage of parallelism.

We adopt 3 general benchmark suites (SPEC2006, Geekbech4, Geekbench5) and ones from the cooperation partner, to test the processor performance. SPEC is the most widely used benchmark in academia and industry. Geekbench4 and Geekbench5 focus on the computing power of CPUs and memory systems. The goal of DSE is to find the design schemes with the highest total score among the four benchmark suites. Because of the disparity of the program characteristics contained in different benchmark suites, we construct the surrogate objective model for each benchmark suite to obtain four sets of surrogate weights, i.e.,

$$c_b = \{c_{b,i}^{(k)} \,|k = 1, 2, \ldots, m_i; i = 1, 2, \ldots, n\}, \tag{17}$$

where b represents the index of the benchmark suite and $b = 1, 2, 3, 4$. We implement the proposed SOBP DSE optimization algorithm in python with MIP library. Since the simulation time dominates the overall execution time, the algorithm solving time is almost negligible. Additionally, the size of the model is not very large, so we have chosen to use the open-source CBC solver, which is more convenient in industrial scenarios.

4.2 Microarchitectural parameters and constraints

We test 43 key microarchitectural parameters listed in Table B1 in Appendix B, i.e., $n = 43$. The types of the parameter are L and E, where L represents that the candidates vary linearly at regular intervals, and E represents that the candidates are $\{2^z | z = \min, \min+1, \ldots, \max-1, \max\}$. The total number of design schemes in design space is $\prod_{i=1}^{n} m_i \approx 1.4 \times 10^{33}$.

Some parameters have strong correlation, such as cache set and cache way of L1Icache, L1Dcache L2cache and BTB. We merge strong coupling parameters into a new variable and sort them by their product plus cache way. The reason is that cache size is positively correlated with performance and the higher the cache way, the better the performance when the cache size is equal. Other parameters, not listed in Table B1, are fixed, such as parameters of branch predictors. Various combinations of parameter candidates generate different microarchitecture design schemes. However, not all design schemes meet design rules, shown in Table B2. In industrial scenarios, there are not only system constraints from the architect, but also customer constraints from the customer, which are not always good for performance improvement. The first two constraints in Table B2 are the specific requirements the customer. Generally, system constraints are more valuable. For example, if a design scheme does not satisfy the fifth and sixth rules, rename width may not reserve enough access for each decoded instruction or may contain redundant entries that cannot be fully used at all. The last three rules in Table B2 indicate the same number of issue queue sizes or write ports in different components, that are added to reduce the design space and the system complexity. Based on the 0-1 transformation in (3) and constraints formulation in (6), the design rules can be modeled as constraints in Table B3. By adding the constraints listed in Table B3, the size of the design space is reduced by 4–5 orders of magnitude.

4.3 Engineering experiment results and analysis

In this subsection, we present the results and analysis of two experiments. The first experiment aims to compare the exploration efficiency of our proposed SOBP DSE algorithm with the state-of-the-art black box optimization algorithm, HEBO [7], which were placed first on the NeurIPS 2020 Black-Box Optimisation challenge. The second experiment aims to examine the effectiveness enhancement of our method on HEBO. Due to the high simulation cost of the industrial CAS, the comparison experiment with more methods is not allowed. Therefore, we have to choose one of the current most advanced BO method in the literatures.

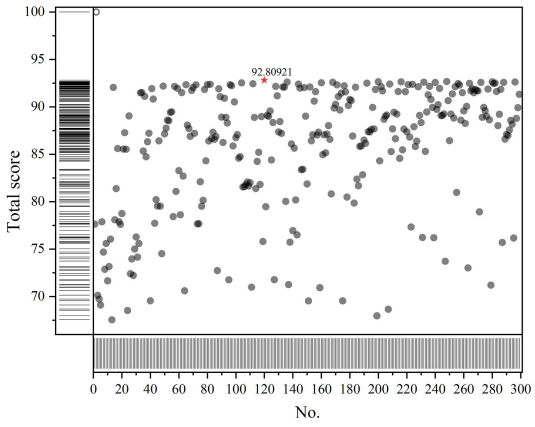

Figure 3 (Color online) Initial simulation dataset for model construction. The hollow circle represents that all parameters take the maximum value, when ignoring design rules. The red star represents the optimal design scheme that satisfies the design rules in the initial dataset.

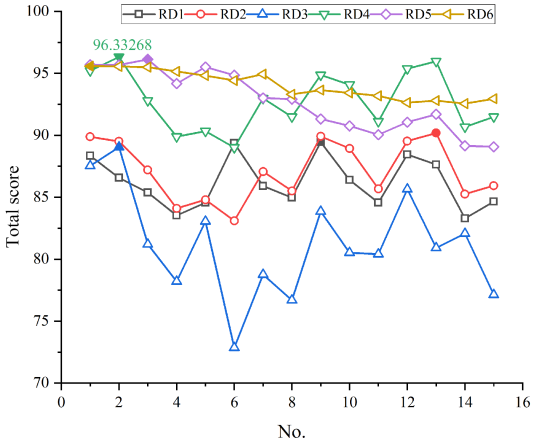

Figure 4 (Color online) Exploration results of SOBP DSE optimization algorithm with initial dataset. Solid dots indicate the optimal design schemes for each round.

In recent years, BO has gained popularity as a method for solving black-box optimization problems and has become a widely used method for DSE in expensive simulation system [7,31,32,44]. It is a global optimization approach in case that the objective function only needs to satisfy local smoothness assumptions such as uniform continuity or Lipschitz continuity. It enables obtaining approximate solutions of complex objective functions with fewer evaluations. In this study, HEBO, one of the state-of-the-art BO methods, is selected as the control method.

Before the DSE algorithms are applied, 300 design schemes with simulation results obtained from artificial recommendations are used to construct the initial dataset. The scores are normalized based on the score of 100 with the maximal candidates of all parameters, shown in Figure 3. The highest score is 92.8 in initial dataset. There are 50 points scoring more than 92, indicating that some good design schemes can be found based on expert experience. However, further enhancements are difficult due to the unclear direction of artificial exploration, which may lead to a great waste of simulation resources. Additionally, we set the maximum iteration rounds as 10, i.e., $T = 10$. The maximum simulation number is 500.

Figure 4 shows the experiment results of SOBP DSE algorithm, where RD represents the iterative round. Although the simulator can simulate up to 50 design schemes per round in parallel, the design schemes recommended by our method have a high similarity per round. It is costly to test algorithm hyperparameters. We test the algorithm by training a neural network based on the initial dataset. We find that exploring 15 points per round, i.e., $S = 15$ in Algorithm 1, is a best balance between convergence speed and simulation overhead. We define diversity as the Hamming distance between different design schemes. In order to balance the exploration and exploitation, we select top 5 design schemes based on surrogate objective and additional 10 design schemes with the greatest diversity from the best design scheme to be evaluated through the simulator. Our algorithm converges after 6 iterations. The first three iterations fail to find better design schemes than the optimal in initial dataset. Because the initial dataset is too small and lacks simulation data of many critical areas, the error from the surrogate objective function misleads the DSE model to recommend design schemes of areas without simulation information. However, once the simulation information of those areas is obtained, the algorithm can converge to the satisfactory space quickly. The optimal design scheme with total score of 96.33 appears in round 4. The last two rounds of exploration find some more design solutions above the baseline. However, the highest score has no improvement, which meets the convergence condition, no promotion for two consecutive rounds. Our approach obtains 3.8% performance improvement by recommending 90 design schemes with 6 rounds.

Figure 5 shows the experiment results of HEBO based on the initial dataset. HEBO takes full advantage of the parallelism of the simulator, where 90% is recommended by algorithm and 10% is sampled randomly in order to avoid getting trapped in a local optimum. Since the constraints are modeled as a penalty term, some of the recommended design solutions violate hard constraints. Figure 6 shows the simulation

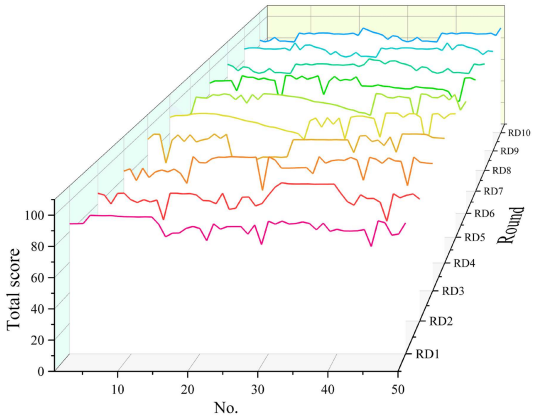

Figure 5 (Color online) Exploration results of HEBO algorithm with initial dataset. Some schemes do not satisfy the design rules. HEBO struggles in the last three rounds.

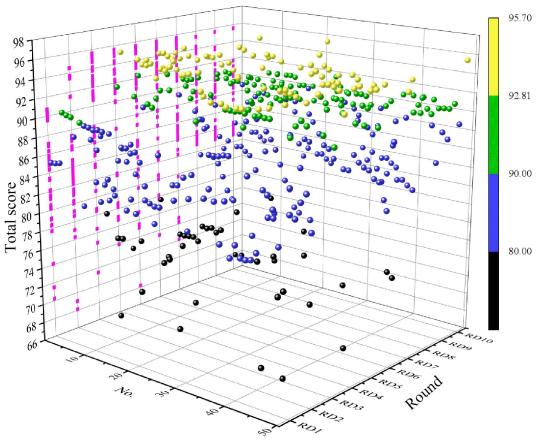

Figure 6 (Color online) All feasible schemes of HEBO algorithm. 89% of the schemes satisfied the constraints, and 90 design schemes outperformed the best in the initial dataset.

Table 1 Algorithm overhead comparison.

Method	Iteration	Simulation	Highest score	Feasible rate (%)
HEBO	10	500	95.64	89
SOBP	6	90	96.33	100

result of points satisfying the constraints. On average, 15.6% of the points in each round are non-feasible solutions, resulting in a waste of simulation resources. The optimal design scheme with total score of 95.64 appears in round 7. The last three rounds of exploration found some better design schemes above the baseline of the initial dataset. However, the highest total score has no improvement. Eventually, the algorithm stops due to the exhaustion of simulation resources. For real-world processor DSE, HEBO requires the addition of a termination condition to avoid the waste of expensive simulation resources. Since the running time of each iteration of the DSE optimization algorithm is a few minutes, it is almost negligible relative to the running time of the CAS. In other words, the efficiency of the DSE algorithm depends on the number of iterations [27].

In summary, both methods find better design schemes than initial dataset. We compared the metrics of the two algorithms in various aspects in Table 1. SOBP DSE algorithm saves 82% simulation overhead and 40% time overhead approximately compared to HEBO algorithm. The exploration efficiency of SOBP is mainly thanks to three aspects: (1) constraint pruning of exploration space reduces much unnecessary effort; (2) the processor's mechanistic knowledge and expert experience make the exploration more purposeful; (3) the surrogate objective acquisition method effectively reduces data requirements. Trend of the score of the optimal design scheme from two algorithms with rounds of simulation is shown in Figure 7. The reasons for the poor performance of our method in the first three rounds are the lack of simulation information in the several critical spaces for surrogate objective in the sense of order-preserving. Once the information is acquired, the algorithm converges to the optimal space. The reason for the degradation of optimal performance in the last 2 rounds is that the design schemes recommended previously is eliminated from the design space by adding constraints. HEBO is better in the first three rounds, but the optimal design scheme is inferior to the SOBP DSE algorithm. HEBO finds over a hundred better design schemes than the optimal in the initial dataset and may converge to the suboptimal space because of the extremely limited simulation budget.

In experiment 2, we combine the original initial dataset with the simulation data from the previous seven rounds of HEBO into a new initial dataset. Subsequently, we apply the SOBP DSE algorithm to evaluate its enhancement effect on HEBO and examine the dataset's impact. Notably, we retain design schemes that may not meet the constraints as they provide valuable insights into system characteristics. Due to simulation overhead limitations, we only tested our algorithm briefly and interacted with the CAS for two rounds.

The experimental results of the SOBP DSE algorithm with the new initial dataset are presented in Figure 8, which shows that our algorithm discovers better design schemes compared to HEBO. The total

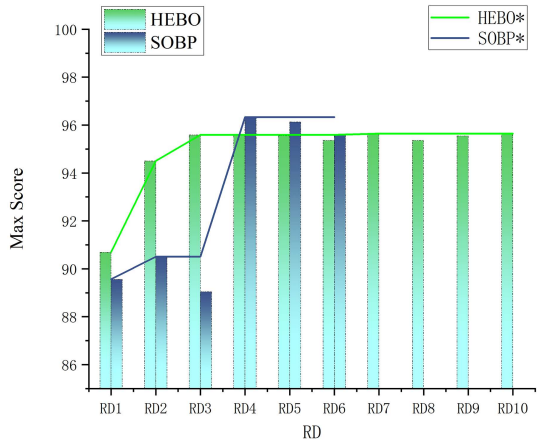

Figure 7 (Color online) Optimal performance comparison with iteration rounds. HEBO iterates for 10 rounds, while SOBP iterates for 6 rounds. The '∗' represents the score of the historical optimal design schemes. In SOBP, the reason for the degradation of optimal performance in the last 2 rounds is that the design schemes recommended previously are eliminated from the design space.

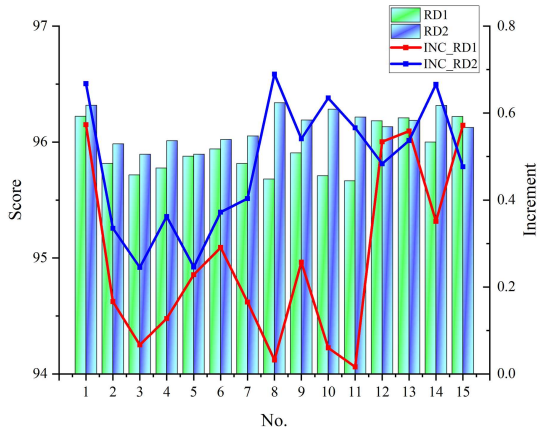

Figure 8 (Color online) Exploration results of SOBP DSE algorithm with the new initial dataset. When comparing the improvement of the SOBP over the HEBO, the highest score achieved by HEBO is used as the baseline.

Table 2 Optimal solution of two DSE algorithms.

No.	Design scheme
1	40,12,**8**,64,**9**,4,**8**,4,4,14,10,44,136,72,28,80,4,3,4,3,24,24,24,20,20,20,2,4,4,154,100,20,16,58,**8**,4,44,40,28,15,**10**,8,26
2	56,12,**8**,64,**9**,4,**8**,4,4,14,10,44,136,72,28,80,4,3,4,3,24,24,24,20,20,20,2,4,4,154,100,20,16,58,**8**,4,42,40,28,15,**10**,8,26

score of the best design scheme obtained by SOBP DSE algorithm is 96.32, which is nearly equal to the highest score obtained in the first experiment. Therefore, we terminated the exploration. Although the improvement is modest, it is very valuable to improve in the space that is close to the full score.

The results demonstrate that SOBP DSE algorithm can improve the optimization effect with a reasonable simulation and time cost when the HEBO algorithm struggles. Furthermore, it is not difficult to see that with the improvement of the quantity and quality of datasets, the exploration effect of the SOBP DSE algorithm will become better. The optimal design schemes of both experiments are listed in Table 2, where bold indicates the candidates are exponential. Only two design parameters, instruction queue size and memory data block size, are different in the two design schemes, which indicates that our approach in both experiments converges to the same optimal design space.

4.4 Numerical experiment results and analysis

In this subsection, we adopt commonly used black-box optimization functions for the evaluation of the optimization algorithm instead of the actual simulator. The performance of SOBP is compared with the latest improved evolutionary algorithms and HEBO. Meanwhile, the actual performance of the algorithm was tested when the assumptions did not hold.

A comprehensive comparison is performed with three other state-of-the-art algorithm: HEBO [7], SRADE [8], ECO-HCT [9]. SRADE and ECO-HCT are state-of-the-art advanced evolutionary algorithms capable of handling constraints. In SRADE, three mutation strategies with distinct focuses are collaboratively employed and adaptively assigned to different individuals based on their performance feedback. The information of the population evolution process is used by HCT to maintain the objective function and constraints balance, and combines the evolutionary algorithm and HCT to propose ECO-HCT to solve constrained optimization problems. In the experiments, the algorithm hyperparameters and code were all based on the data publicly disclosed by the authors. Considering the specific characteristics of our problem, we discretized the solutions obtained by the algorithm. Additionally, the fitness function is adjusted according to the evaluation function.

All experiments are carried out on the windows 10 platform of Intel(R) Core(TM) i5-9500 CPU 3.00 GHz and 16 GB RAM. In the implementation of the SOBP DSE algorithm, an initial dataset containing 500 points is generated by using Latin hypercube sampling. And then, let $S = 10$, $T = 50$.

Table 3 Experimental results of SOBP and HEBO, ECO-HCT, SRADE on 35 test benchmarks[a)].

Problem	ECO-HCT	SRADE	HEBO	SOBP
	Mean OFV ± Std Dev	Mean OFV | success/total	Mean OFV ± Std Dev	Mean OFV ± Std Dev
A01	7.10E10 ± 1.47E10 −	8.46E10 | 12/25 −	9.40E10 ± 1.01E10 −	**3.23E11** ± *
A02	1.60E12 ± 1.33E11 −	1.72E12 | 12/25 −	4.38E12 ± 3.42E10 ≈	**4.28E12** ± *
A03	2.64E09 ± 1.09E09 −	1.68E09 | 20/25 −	5.02E09 ± 5.46E08 −	**5.91E09** ± *
A04	1.62E06 ± 9.39E04 −	1.94E06 | 13/25 −	8.84E06 ± 7.25E04 −	**1.40E07** ± *
A05	5.22E02 ± 9.34E−2 ≈	5.22E02 | 16/25 ≈	5.22E02 ± 1.14E−1 ≈	5.22E02 ± *
A06	6.96E02 ± 4.48E00 −	7.01E02 | 15/25 ≈	6.91E02 ± 6.04E00 −	**7.11E02** ± *
A07	1.04E04 ± 6.08E02 −	— | 00/25	2.62E04 ± 2.55E02 ≈	**2.67E04** ± *
A08	3.56E03 ± 1.21E02 −	— | 00/25 −	5.80E03 ± 2.26E01 ≈	**5.84E03** ± *
A09	7.12E03 ± 4.47E02 −	6.61E03 | 23/25 −	8.44E03 ± 7.71E02 ≈	**8.46E03** ± *
A10	1.92E04 ± 1.09E03 −	2.01E04 | 18/25 −	2.47E04 ± 3.46E02 ≈	**2.48E04** ± *
A11	2.08E04 ± 1.53E03 −	— | 00/25	**2.75E04** ± 9.95E01 +	2.54E04 ± *
A12	1.21E03 ± 3.42E00 −	1.22E03 | 21/25 −	1.03E04 ± 4.46E00 −	**1.23E04** ± *
C13	1.35E03 ± 2.40E−1 ≈	1.34E03 | 20/25 ≈	**1.37E03** ± 1.09E00 ≈	1.36E03 ± *
A14	3.76E03 ± 1.12E02 −	3.92E03 | 23/25 −	4.29E03 ± 2.12E01 −	**5.53E03** ± *
A15	6.42E12 ± 3.04E12 −	3.51E12 | 24/25 −	3.34E13 ± 2.91E13 −	**3.69E14** ± *
A16	1.62E03 ± 2.39E01 ≈	— | 00/25 −	1.63E03 ± 5.36E00 ≈	1.63E03 ± *
A17	2.77E10 ± 3.99E09 −	— | 00/25 −	4.42E10 ± 6.64E09 −	**4.63E10** ± *
A18	4.40E10 ± 8.73E09 −	— | 00/25 −	9.42E10 ± 2.27E09 −	**1.01E11** ± *
A19	6.80E05 ± 1.49E05 +	— | 00/25 −	**6.82E05** ± 3.34E05 +	4.95E05 ± *
A20	1.16E10 ± 3.43E09 −	— | 00/25 −	1.99E10 ± 6.57E09 −	**2.52E10** ± *
A21	2.07E09 ± 1.21E09 −	— | 00/25 −	8.15E09 ± 3.27E08 −	**8.76E09** ± *
A22	1.58E08 ± 8.03E07 −	— | 00/25 −	4.61E08 ± 3.52E07 −	**5.43E08** ± *
A23	3.80E04 ± 4.41E03 −	— | 00/25 −	**4.32E04** ± 1.98E03 +	4.21E04 ± *
A24	1.49E04 ± 9.85E02 −	— | 00/25 −	2.06E04 ± 6.17E02 −	**2.24E04** ± *
A25	3.94E03 ± 2.99E02 −	— | 00/25 −	1.00E04 ± 9.43E02 ≈	**1.14E04** ± *
A26	2.93E03 ± 2.47E02 −	— | 00/25 −	2.30E03 ± 2.84E02 −	**6.26E03** ± *
A27	8.49E04 ± 1.13E04 −	— | 00/25 −	1.99E04 ± 1.33E04 −	**2.85E05** ± *
A28	3.56E04 ± 4.92E03 −	— | 00/25 −	**6.41E04** ± 5.05E03 ≈	6.40E04 ± *
A29	1.83E08 ± 5.30E07 −	— | 00/25 −	7.05E08 ± 1.54E07 −	**2.04E09** ± *
A30	1.93E08 ± 2.99E08 −	— | 00/25 −	1.21E09 ± 2.38E08 −	**2.38E09** ± *
B31	2.01E13 ± 4.85E12 −	1.98E13 ± 2.44E12	8.82E13 ± 8.61E11 −	**1.17E14** ± *
B32	8.13E05 ± 3.55E04 −	8.05E05 | 14/25 −	8.96E05 ± 9.94E03 −	**1.47E06** ± *
B33	**1.98E07** ± 1.85E06 +	1.66E07 | 11/25 −	1.96E07 ± 6.64E06 ≈	1.95E07 ± *
B34	2.13E04 ± 4.43E02 ≈	2.20E04 | 18/25 ≈	**2.23E04** ± 2.15E02 ≈	2.22E04 ± *
C35	1.43E04 ± 5.64E02 −	1.56E04 ± 7.81E02 −	1.54E04 ± 3.56E02 −	**2.03E04** ± *
−	29	32	22	/
+	2	0	3	/
≈	4	3	10	/

a) "Mean OFV" and "Std Dev" respectively represent the mean value and variance of the OFVs obtained in 25 independent runs. "∗" indicates that the proposed method is deterministic and does not have this indicator. The best solutions found by the optimization algorithms are shown in bold.

The black-box test benchmarks are categorized into three sources. The first category consists of the test suite of the IEEE CEC2014 competition (A01–A30) [45]. The second category consists of commonly black-box optimization test functions, including Rosebrock, Rastrigin, Sum Squares and Schwefel function (B31–B34) [46]. The third category is a simple separable function designed by ourselves (C35), $F(x) = \sum_{i=1}^{n} w_i x_i$, where w_i is generated randomly. Since the test suite of the IEEE CEC2014 competition can only evaluate questions in 10, 30, 50 and 100 dimensions, we expand the search space to 50 dimensions. The maximum evaluation number is 1000.

The experimental results are summarized in Table 3, where "−", "+", "≈" represent that the performance of the corresponding algorithm is worse than, better than, and similar to that of SOBP, respectively. When a method cannot achieve at least a feasible solution on a test function at the end of some runs, "—" is placed to fill the mean objective function value (OFV). SOBP performs the best on 30 test functions. The best value of SOBP is approximately one order of magnitude ahead of other algorithms across 7 test

functions. However, the performance of SOBP will be limited, when the real objective function does not satisfy the assumption of order-preserving separability, such as A19 [45] in Table 3. The comparison of algorithm running time is shown in Table C1 in Appendix C. With the progress of iteration, SOBP will solve an integer programming problem with more and more constraints, and the solution time will be slightly longer. However, in the actual processor DSE, the running time of the optimization algorithm is almost negligible compared with the simulation time.

5 Conclusion and future work

The engineering experiment results provide evidence that sufficiently simple separable surrogate objective functions exist for the performance model of processor. The proposed surrogate objective relates the candidates of microarchitectural parameters to the processor performance, enabling efficient exploration of the design space through the BIP DSE model. The experiments show that the SOBP DSE approach is more effective and efficient than the HEBO [7], which is considered one of the most advanced BO algorithms, in an industrial processor DSE case. Due to the error from CAS and models, many DSE optimization algorithms are difficult to find the optimal design scheme, resulting in low exploration efficiency in the later stage. The second experiment demonstrates that SOBP can step in to improve effectiveness and efficiency when HEBO falls into inefficiency. Furthermore, the model can incorporate expert experience and customer requirements, reducing the design space and improving exploration efficiency.

The comparison with the advanced evolutionary algorithms that can handle constraints is added in the numerical experiment. The experiment in black-box benchmark functions proves that simple separable surrogate objective functions exist for many black-box functions in the sense of order preserving. Meanwhile, the performance of SOBP will be limited if the assumptions do not hold.

The SOBP DSE method is not restricted to processor design space exploration and can be extended to other expensive simulation systems that are likely to satisfy the assumption of order-preserving separability. By incorporating the system characteristics and expert knowledge, a surrogate objective function can be simplified, and then a MILP model can be constructed. However, further research is needed to evaluate the generalizability of the model. Currently, we are working on developing an approach that does not require an initial dataset and using MILP models for multi-objective optimization of expensive simulation systems without analytical models.

Acknowledgements This work was supported by Hisilicon Huawei.

Supporting information Appendixes A–C. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Hu W W, Zhang Y F, Fu J. An introduction to CPU and DSP design in China. Sci China Inf Sci, 2016, 59: 012101
2 Vahid F, Givargis T. Embedded System Design: A Unified Hardware/Software Introduction. Hoboken: John Wiley & Sons, 2023
3 Austin T, Larson E, Ernst D. SimpleScalar: an infrastructure for computer system modeling. Computer, 2002, 35: 59–67
4 Emer J, Ahuja P, Borch E, et al. Asim: a performance model framework. Computer, 2002, 35: 68–76
5 Binkert N L, Dreslinski R G, Hsu L R, et al. The M5 simulator: modeling networked systems. IEEE Micro, 2006, 26: 52–60
6 Martin M M K, Sorin D J, Beckmann B M, et al. Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset. SIGARCH Comput Archit News, 2005, 33: 92–99
7 Cowen-Rivers A I, Lyu W, Tutunov R, et al. HEBO: an empirical study of assumptions in Bayesian optimisation. J Artif Intell Res, 2022, 74: 1269–1349
8 Peng H, Xu Z, Qian J, et al. Evolutionary constrained optimization with hybrid constraint-handling technique. Expert Syst Appl, 2023, 211: 118660
9 Qiao K, Liang J, Yu K, et al. Self-adaptive resources allocation-based differential evolution for constrained evolutionary optimization. Knowl-Based Syst, 2022, 235: 107653
10 Eeckhout L. Computer architecture performance evaluation methods. Synthesis Lect Comput Archit, 2010, 5: 1–145
11 Eyerman S, Eeckhout L, Karkhanis T, et al. A mechanistic performance model for superscalar out-of-order processors. ACM Trans Comput Syst, 2009, 27: 1–37
12 Van den Steen S, Eyerman S, de Pestel S, et al. Analytical processor performance and power modeling using micro-architecture independent characteristics. IEEE Trans Comput, 2016, 65: 3537–3551
13 Joseph P J, Vaswani K, Thazhuthaveetil M J. Construction and use of linear regression models for processor performance analysis. In: Proceedings of the 12th International Symposium on High-Performance Computer Architecture (HPCA), 2006. 99–108
14 Michaud P, Seznec A, Jourdan S. Exploring instruction-fetch bandwidth requirement in wide-issue superscalar processors. In: Proceedings of the International Conference on Parallel Architectures and Compilation Techniques, 1999. 2–10
15 Riseman E M, Foster C C. The inhibition of potential parallelism by conditional jumps. IEEE Trans Comput, 1972, 21: 1405–1411

16 Lee B C, Brooks D M. Accurate and efficient regression modeling for microarchitectural performance and power prediction. SIGOPS Oper Syst Rev, 2006, 40: 185–194
17 Dubach C, Jones T, O'Boyle M. Microarchitectural design space exploration using an architecture-centric approach. In: Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2007. 262–271
18 Joseph P J, Vaswani K, Thazhuthaveetil M J. A predictive performance model for superscalar processors. In: Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2006. 161–170
19 Ïpek E, McKee S A, Caruana R, et al. Efficiently exploring architectural design spaces via predictive modeling. SIGOPS Oper Syst Rev, 2006, 40: 195–206
20 Zhang Y, Kim N H, Haftka R T. General-surrogate adaptive sampling using interquartile range for design space exploration. J Mech Des, 2020, 142: 051402
21 Hartstein A, Puzak T R. The optimum pipeline depth for a microprocessor. SIGARCH Comput Archit News, 2002, 30: 7–13
22 Zhang K P, Xu L, Yi X L, et al. Predefined-time distributed multiobjective optimization for network resource allocation. Sci China Inf Sci, 2023, 66: 170204
23 Shrivastava A, Sanghyun A P, Earlie E, et al. Automatic design space exploration of register bypasses in embedded processors. IEEE Trans Comput-Aided Des Integr Circ Syst, 2007, 26: 2102–2115
24 Carrion Schafer B. Probabilistic multiknob high-level synthesis design space exploration acceleration. IEEE Trans Comput-Aided Des Integr Circ Syst, 2016, 35: 394–406
25 Bai C, Sun Q, Zhai J, et al. BOOM-Explorer: RISC-V BOOM microarchitecture design space exploration framework. In: Proceedings of the IEEE/ACM International Conference On Computer Aided Design (ICCAD), 2021. 1–9
26 Karkhanis T S, Smith J E. Automated design of application specific superscalar processors: an analytical approach. In: Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA), 2007. 402–411
27 Li D, Yao S, Liu Y H, et al. Efficient design space exploration via statistical sampling and AdaBoost learning. In: Proceedings of the 53rd Annual Design Automation Conference, 2016. 1–6
28 Li L, Jamieson K, DeSalvo G, et al. Hyperband: a novel bandit-based approach to hyperparameter optimization. 2017. ArXiv:1603.06560
29 Liu Y, Wang Y, Singh A. Smooth bandit optimization: generalization to holder space. In: Proceedings of the International Conference on Artificial Intelligence and Statistics, 2021. 2206–2214
30 Thaker P, Malu M, Rao N, et al. Maximizing and satisficing in multi-armed bandits with graph information. 2022. ArXiv:2108.01152
31 Lyu W, Xue P, Yang F, et al. An efficient Bayesian optimization approach for automated optimization of analog circuits. IEEE Trans Circ Syst I, 2017, 65: 1954–1967
32 Zhang S, Yang F, Yan C, et al. An efficient batch-constrained Bayesian optimization approach for analog circuit synthesis via multiobjective acquisition ensemble. IEEE Trans Comput-Aided Des Integr Circ Syst, 2021, 41: 1–14
33 Liu B, Zhao D, Reynaert P, et al. GASPAD: a general and efficient mm-wave integrated circuit synthesis method based on surrogate model assisted evolutionary algorithm. IEEE Trans Comput-Aided Des Integr Circ Syst, 2014, 33: 169–182
34 Yang Y, Zhu H, Bi Z, et al. Smart-MSP: a self-adaptive multiple starting point optimization approach for analog circuit synthesis. IEEE Trans Comput-Aided Des Integr Circ Syst, 2017, 37: 531–544
35 Liu B, Wang Y, Yu Z, et al. Analog circuit optimization system based on hybrid evolutionary algorithms. Integration, 2009, 42: 137–148
36 Vural R A, Yildirim T. Analog circuit sizing via swarm intelligence. AEU Int J Electron Commun, 2012, 66: 732–740
37 Phelps R, Krasnicki M, Rutenbar R A, et al. Anaconda: simulation-based synthesis of analog circuits via stochastic pattern search. IEEE Trans Comput-Aided Des Integr Circ Syst, 2000, 19: 703–717
38 Wu J, Frazier P. The parallel knowledge gradient method for batch Bayesian optimization. 2016. ArXiv:1606.04414
39 Wang J, Clark S C, Liu E, et al. Metrics optimization engine. 2014. http://yelp.github.io/MOE/
40 Snoek J, Larochelle H. Spearmint. 2015. https://github.com/HIPS/Spearmint
41 Ho Y C, Sreenivas R S, Vakili P. Ordinal optimization of DEDS. Discrete Event Dyn Syst, 1992, 2: 61–88
42 Celio C, Patterson D A, Asanovic K. The Berkeley Out-of-order Machine (BOOM): an Industry-Competitive, Synthesizable, Parameterized Risc-V Processor. Technical Report UCB/EECS-2015-167, 2015
43 Feng D G, Liu J B, Qin Y, et al. Trusted computing theory and technology in innovation-driven development (in Chinese). Sci Sin Inform, 2020, 50: 1127–1147
44 Zhang S, Lyu W, Yang F, et al. Bayesian optimization approach for analog circuit synthesis using neural network. In: Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE), 2019. 1463–1468
45 Liang J J, Qu B Y, Suganthan P N. Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, 2013
46 Qi X, Zhu S, Zhang H. A hybrid firefly algorithm. In: Proceedings of the 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2017. 287–291