ITC 1/53
Information Technology and Control
Vol. 53 / No. 1 / 2024
pp.280-301
DOI 10.5755/j01.itc.53.1.35922

A Multi-level Surrogate-assisted Algorithm for ExpensiveOptimization Problems

Received 2024/01/02 | Accepted after revision 2024/02/06

HOW TO CITE: Hu, L., Wu, X., Che, X. (2024). A Multi-level Surrogate-assisted Algorithm for Expensive Optimization Problems. *Information Technology and Control*, 53(1), 280-301. https://doi.org/10.5755/j01.itc.53.1.35922

A Multi-level Surrogate-assisted Algorithm for Expensive Optimization Problems

Liang Hu
Key Laboratory of Symbolic Computation and Knowledge Engineering of the Ministry of Education, Jilin University, Changchun, China:
College of Computer Science and Technology, Jilin University, Changchun, China; e-mail: hul@jlu.edu.cn

Xianwei Wu
College of Computer Science and Technology, Jilin University, Changchun, China; e-mail: xwwu18@mails.jlu.edu.cn

Xilong Che
Key Laboratory of Symbolic Computation and Knowledge Engineering of the Ministry of Education, Jilin University, Changchun, China;
College of Computer Science and Technology, Jilin University, Changchun, China; e-mail: chexilong@jlu.edu.cn

Corresponding author: chexilong@jlu.edu.cn

With the development of computer science, more and more complex problems rely on the help of computers for solving. When facing the parameter optimization problem of complex models, traditional intelligent optimization algorithms often require multiple iterations on the target problem. It can bring unacceptable costs and resource costs in dealing with these complex problems. In order to solve the parameter optimization of complex problems, in this paper we propose a multi-level surrogate-assisted optimization algorithm (MLSAO). By constructing surrogate models at different levels, the algorithm effectively explores the parameter space, avoiding local optima and enhancing optimization efficiency. The method combines two optimization algorithms, differential evolution (DE) and Downhill simplex method. DE is focused on global level surrogate model optimization. Downhill simplex is concentrated on local level surrogate model update. Random forest and inverse distance weighting (IDW) are constructed for global and local level surrogate model, respectively. These methods leverage their respective advantages at different stages of the algorithm. The MLSAO algorithm is evaluated against other state-of-the-art approaches using benchmark functions of varying dimensions. Comprehensive results from the comparisons showcase the superior performance of the MLSAO algorithm in addressing expensive

optimization problems. Moreover, we implement the MLSAO algorithm for tuning precipitation parameters in the Community Earth System Model (CESM). The outcomes reveal its effective enhancement of CESM's simulation accuracy for precipitation in the North Indian Ocean and the North Pacific region. These experiments demonstrate that MLSAO can better address parameter optimization problems under complex conditions.

KEYWORDS: Surrogate-assisted optimization; Random Forest; Inverse distance weighting; Multi level optimization.

1. Introduction

Parameter optimization issues are prevalent in the fields of science and engineering, necessitating the need for highly efficient computational algorithms to address these challenges. In recent years, a variety of intelligent optimization algorithms, such as differential evolution (DE) algorithm, genetic algorithm, particle swarm optimization (PSO) algorithm, have been proposed and effectively employed in a multitude of engineering optimization scenarios. Their effectiveness can be partly attributed to the circumstance that intelligent optimization methods do not require objective functions to be analytical or differentiable, while also possessing enhanced global search capabilities. Nevertheless, the primary trait of these algorithms is their tendency to demand a substantial number of fitness evaluations in order to pinpoint a solution that is close to optimal. Some optimization problems may involve intensive computation and costly simulation [16]. Applying these algorithms to these problems with high computational costs present a significant challenge.

To tackle this issue, surrogate model is proposed as an alternative to the costly performance evaluations in order to mitigate the computational expenses. Surrogate model is an effective tool for building a simplified model of the actual complex system for rapid testing, verification and optimization. Surrogate model uses approximate methods to construct models instead of complex models to simplify the optimization process, so as to improve computational efficiency while ensuring accuracy. Surrogate model predicts information of unknown points through known sampling points, which can actually be attributed to construct an equation to instead of the large complex model involving costly simulation. It is an approximate method based on experimental design technology. Surrogate model establishes a mathematical model between input parameter and output objective function values, predicts the output objective function values under different input parameters. The surrogate model offers the advantage of significantly lowering the cost and complexity of analysis, enhancing analysis efficiency and accuracy, and optimizing real systems during the design stage to minimize the need for trial and error.

Considering that the process of building surrogate models simplifies the complex processes of actual complex systems into mathematical models, it becomes challenging to fully simulate the overall behavior of real systems. This error increases with the growing complexity of real systems. As various optimization problems become increasingly complex, it becomes challenging to ensure the accuracy of a single-level surrogate model. Such models may struggle to adequately represent intricate systems and meet precision requirements for minimizing errors. Moreover, there is a tendency for it to become ensnared in local optima, potentially failing to discover the genuine global optimum throughout the optimization process. Thus, improving the fitting accuracy of surrogate models and evading local optima stands out as one of the most crucial challenges confronted by optimization algorithms based on surrogate model. In previous research, when dealing with such problems, more emphasis was placed on using different algorithms in various search stages, rather than constructing surrogate models at different levels. However, in practical optimization problems, when the problem to be optimized is complex, it is necessary to set up multi-level surrogate models. Constructing multiple levels of surrogate models and conducting searches at various stages can achieve a more optimal balance between exploration and exploitation, thereby increasing the likelihood of identifying potential optimal solutions.

In this paper, we propose a multi-level surrogate assisted optimization (MLSAO) algorithm. We attempt to build a global-level surrogate model by a machine

learning method random forest, which can effectively handle high-dimensional data and non-linear relationships, and has good robustness and stability. Then a differential evolution algorithm was applied to obtain the optimal solution of the global surrogate model and update the global surrogate model. To navigate away from local optimal solution and explore the global optimum, we establish a local surrogate model based on inverse distance weighting. This model helps explore regions where optimal solutions may be present. The local model is simpler and smaller in scale than the global surrogate model. We select a small portion of high-quality data to build this new surrogate model, aiming to construct a surrogate model with high-quality and avoid searching meaningless parameter space. Considering the data scale of the local surrogate model, we design a simplex downhill method-oriented local model update strategy. Compared with intelligent algorithms, the simplex downhill method converges quickly and does not require repetitive iterations. It is suitable for small-scale optimization problems. We combine two level surrogates with different update strategies to realize an algorithm that can solve complex model optimization problems. To assess the effectiveness of the proposed algorithm, experiments are conducted using diverse mathematical function benchmarks. The results indicate that the proposed algorithm has more advantages compared to previous algorithms. Finally, we apply the proposed algorithm to a complex optimization problem: the parameter tuning of the earth system model, the precipitation simulation results of CAM5 has been improved over several regions.

The contributions of this work can be summarized as follows:

1 We propose the MLSAO algorithm, which utilizes Random Forest to construct a global surrogate model and inverse distance weighting (IDW) to construct a local-level surrogate model. This approach enhances the search capabilities for target problem and helps avoid falling into local optimal solutions.

2 We integrate two powerful optimization algorithms, differential Evolution (DE) and Simplex Downhill, into MLSAO, applying them to different search stages. This capitalizes on the strengths of each method, striking a balance between exploration and exploitation.

3 We conduct tests on multiple benchmark functions with varying dimensions and apply MLSAO to parameter tuning in CESM, demonstrating its effectiveness.

The manuscript is structured as follows: In Section 2, we introduce some related works about the proposed algorithm. We provide an overview of the algorithms incorporated in the proposed MLSAO in Section 3 and elaborates on the details of the proposed algorithm in Section 4. Experimental results are presented in Section 5, and the study's conclusions are summarized in Section 6.

2. Related Works

In recent years, various methods have been employed for constructing surrogate model, including polynomial response surface (PRS) [15], support vector regression (SVR) [50], Kriging [11], radial basis function (RBF) [46], artificial neural networks (ANN) [12, 10, 28], multivariate adaptive regression [49], and random forests [2, 3], etc. Building upon this foundation, numerous studies have explored the distinctions between these models and the scenarios in which they are best suited [1, 9, 14, 41], contributing to their widespread application across various engineering domains. For instance, paper [24] utilized a PRS model to assist in making parameter choices and to facilitate the comparison of sensitivity properties among climate models. Xu et al. [44] introduced an optimization method for land model parameter tuning using a RBF surrogate model. Müller et al. [23] used RBF model for methane emission estimation. Yue et al. [48] used RBF adaptive surrogate model optimization to search for a combination of parameters relevant to the geometry and elasticity of track structures. [32, 45] proved that SVR model can be used for parameter calibrate of finite element and satellite systems. Chu et al. [7] used Kriging model for resonance frequency analysis of dental.

To address computationally expensive problems more effectively, researchers have introduced surrogate-assisted optimization algorithms. Over the past decades, the literature has documented various algorithms that leverage surrogate models to enhance optimization processes. For example, Yu et al. [47] propose a surrogate-assisted hierarchical particle

swarm optimization (SHPSO) algorithm, combine RBF model with different PSO method. Liu et al. [21] propose bagging-based surrogate-assisted evolutionary algorithm (B-SAEA). This approach incorporates bagging to construct high-quality surrogate model for each costly objective. Wang et al. [40] proposed the Evolutionary Sampling Assisted Optimization (ESAO) method, leveraging two capabilities to account for both global and local searches. Sun et al. [36] proposed the Two-Layer Surrogate-Assisted PSO (TLSAPSO) algorithm, utilizing both global and several local surrogate models for fitness approximation. Li et al. [19] proposed a surrogate-assisted hybrid swarm optimization (SASHO). Two swarms are respectively used in different optimization states. Xin et al. [42] proposed a surrogate and autoencoder-assisted multitask particle swarm optimization algorithm to solve multimodal optimization problems. A surrogate-assisted differential evolution with knowledge transfer (SADE-KT) [20] integrate knowledge transfer and the surrogate-assisted evolutionary search proposed for expensive incremental optimization problems.

In recent years, some parallel-based optimization methods have been proposed to address parameter optimization for complex problems. They aim to enhance optimization efficiency by leveraging additional computational resources. For example: Parallel BNN-GA [6] has been demonstrated to better address large-scale data optimization problems. HAS-EA [22] proposed a surrogate-assisted optimization algorithm based on heterogeneous platforms. Xing et al [43] proposed a parallel kinging surrogate model optimization method and Improved the expect improvement function,which improved global performance and solution accuracy. However, these methods have high requirements for computational resources. When computational resources are limited or the target problem is difficult to parallelize, the optimization efficiency still cannot be improved.

These algorithms combine surrogate model with many kinds of optimization algorithms. Further enhance the advantages of surrogate models in solving complex problems. However, some of these algorithms involve different level surrogate model, they do not consider using different models at different search stages.

3. Preliminaries

In this section, we provide background information pertinent to this study.

3.1. Problem Definition

The optimization problem can be formulated as follows in equations (1-3):

$$\min_{x \in D} f(x) \tag{1}$$

$$subject\ to\ u_i(x) \le 0, \quad i = 1,2,\dots,n \tag{2}$$

$$v_j(x) = 0, j = 1,2,\dots,r, \tag{3}$$

where the x represents the vector of parameters need to be adjusted. $f(x)$ represents the objective function. While dealing with complex optimization problem, the objective function may not have a clear mathematical expression or the expression may be difficult to compute within a finite time. In the proposed algorithm, the surrogate is considered as the objective function to fit the parameters and corresponding values obtained by the complex model. The functions $u(x)$ and $v(x)$ denote the inequality and equality constraints, respectively. They have different forms of expressions in different problems.

3.2. Differential Evolution Algorithm

Differential evolution (DE) is an evolutionary algorithm for global optimization, introduced in [33]. The algorithm generates new individuals by differential mutation of individuals in the population, and updates the population through selection operation to systematically explore the search space for the global optimum. The primary stages of the DE algorithm encompass population initialization, differential mutation operation, selection operation, and termination condition. The core of the DE algorithm lies in the differential mutation operation, where new individuals are generated through the linear transformation of individuals in the population. The population is then updated through the selection operation. The key steps of DE include:

Initialize optimization conditions: Initialization of optimization conditions involves defining the control parameters of the differential evolution algorithm and the fitness function. These control parameters consist of the population size *NP*, scaling factor *F*, nd crossover probability *CR*. Subsequently, generate and

assess the initial population. The initial population is shown as follows in equations (4-5):

$$X_t(0) \mid x^{L}_{\{t,w\}}(0) \le x_{\{t,w\}} \le x^{U}_{\{t,w\}}(0), \quad (4)$$

$$t \in [1, NP]; \quad w \in [1, Demension], \quad (5)$$

where $X_t(0)$ denotes the t-th individual, while $x^{L}_{\{t,\ w\}}$ and $x^{U}_{\{t,\ w\}}$ denote the lower and upper bounds of the w-th dimension, respectively. The fitness function value for each individual is calculated in the initial population.

Mutation: Conduct mutation operations to acquire the intermediate population.

$$V_t(n+1) = X_{m1}(n) + F\big(X_{m2}(n) - X_{m3}(n)\big). \quad (6)$$

Equation (6) describes the mutation operation where m1 m2, and m3are three random numbers in intervals [1, NP]. F is called the scaling factor, which is a fixed constant. n represents the n-th generation.

Crossover: crossover operations are shown in Equation (7).

$$U_{i,j}(g+1) = \begin{cases} V_{i,j}(g+1)\ if\ rand(0,1) < CR \\ x_{i,j}\ otherwise \end{cases}, \quad (7)$$

where CR is the crossover probability. New individuals are randomly generated by probability.

Selection: Choose individuals from the initial population and the intermediate population to form a new generation population.

$$X_i(n+1) = U_i(n+1), \quad (8)$$

$$if\ f\big(U_i(n+1)\big) \le f\big(X_i(n)\big), \quad (9)$$

$$X_i(n+1) = X_i(n)\ otherwise. \quad (10)$$

As shown in Equations (8)-(10), in DE, a greedy selection strategy is employed, where the superior individual is chosen as the new individual.

DE algorithm has the advantages of simplicity, easy implementation, no need for gradient information, and global convergence. It has found extensive applications in function optimization, parameter estimation, machine learning, and various other domains.

3.3. Nelder-Mead Algorithm

The Nelder-Mead algorithm [25], also referred to as the downhill simplex method, is a widely utilized optimization algorithm designed to locate the minimum or maximum of a given objective function. It falls under the category of direct search algorithms, indicating that it does not necessitate knowledge of the gradient of the objective function.

The term "simplex" denotes a geometric shape created by a set of n + 1 points, ranging from p_0 to p_n, in n-dimensional space. the shape satisfies

$$det\begin{bmatrix} p_0 & p_1 \cdots p_n \\ 1 & 1 \quad 1 \end{bmatrix} \neq 0.$$

It means that in one-dimensional space, two points cannot overlap, so the simplex is a line segment. In two-dimensional space, three points cannot be collinear, so the simplex is a triangle. In three-dimensional space, four points cannot be coplanar, so the simplex is a tetrahedron. This pattern continues for higher dimensions, where the n+1 points cannot lie in an n-dimensional plane, forming an n-dimensional simplex.

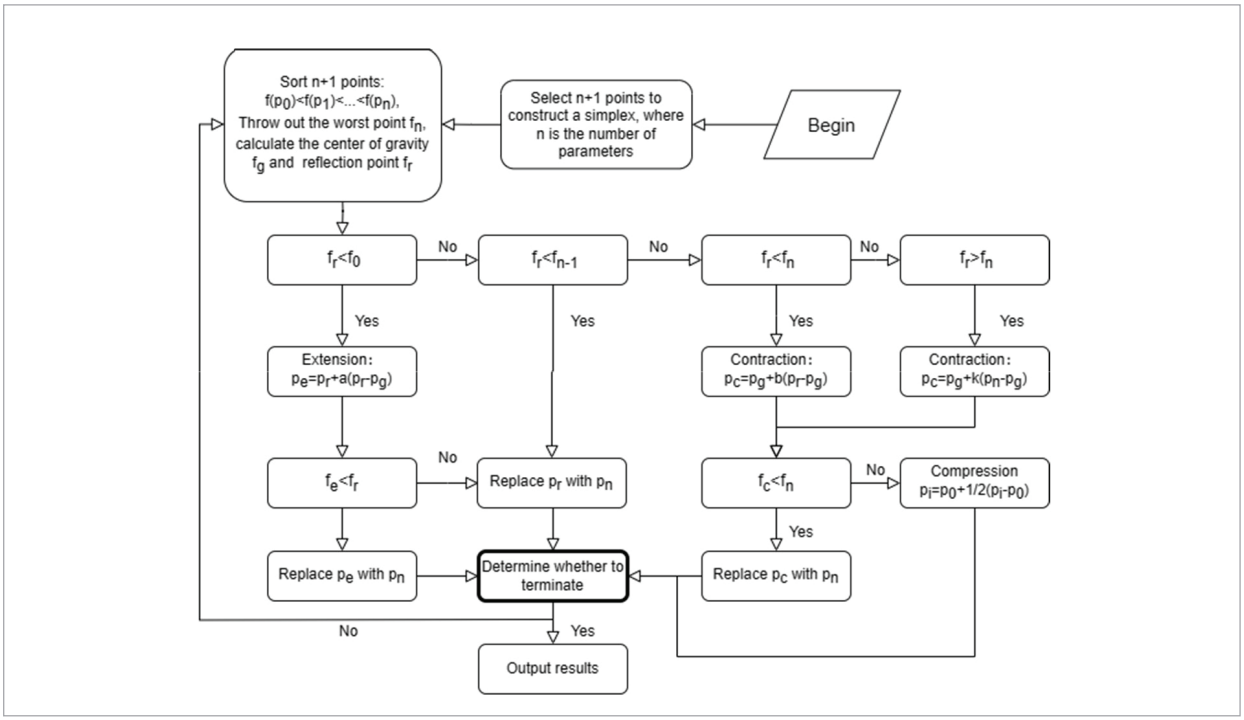
The algorithm operates by maintaining a simplex, which is a geometric shape consisting of n+1 vertices in n-dimensional space. At each iteration, the algorithm evaluates the objective function at each vertex of the simplex, and then performs a series of operations to transform the simplex. These operations include reflection, expansion, contraction, and shrinkage, which move the simplex towards the minimum or maximum of the objective function value.

The process of Nelder-Mead algorithm is as follows and flowchart is shown in Figure 1.

1. Select initial point x_0, generate the remaining n points. Construct a simplex based on these points.
2. 2. Sort the n+1 points according to their objective function values: $f(p_0) \le f(p_1) \le \ldots \le f(p_n)$.
3. Remove the worst point p_n and calculate the center of gravity $p_g = \sum_{i=0}^{n-1} \frac{p_i}{n}$. Reflect the worst point using a reflection coefficient $\rho \le 0$ and ρ is usually set to 1.
4. If $f_0 < f_r < f_{n-1}$, replace point p_n with the reflected point p_r to construct a new simplex.

If $f_r < f_0$, it means the objective function value of the reflected point p_r is smaller than all the points in the simplex. It is favorable for the function value to decrease in this direction. Extend in this direction $p_e = p_g + \mathcal{X}(p_r = p_g)$ with a extension coefficient $\mathcal{X} > 1$ and $\mathcal{X}$ can be set as 2. If $f_e < f_r$, which represents the exten-

Figure 1
The flowchart of Nelder Mead Algorithm



sion is successful. Replace p_l with p_r, otherwise, and replace p_n with p_r.

If $f_r \leq f_{n-1}$, it means the reflected point is still the worst point. A contraction operation is needed: If $f_n > f_r \geq f_{n-1}$, obtain the contraction point $p_c = p_g + \gamma(p_r - p_g)$, where $0 < \gamma < 1$ is the contraction coefficient, which can be set as 0.5. This operation is called an outer contraction. If $f_r \geq f_n$, replace the p_r with p_n, and update the value of p_c based on $p_c = p_g + \gamma(p_n - p_g)$. This operation is called an inner contraction. If $f_c \leq f_n$, the contraction is considered successfully and replace p_n with p_c. Otherwise, the contraction is failed. A new simplex is constructed: keep p_0 and halve the distance between p_0 and the other points, which is call a compression operation.

4. The Multi-level Surrogate-assist Optimization Algorithm

In this section, the procedure of the proposed multi-level surrogate-assisted optimization algorithm will be introduced in each subsection.

4.1. The Procedure of the Proposed Algorithm

Building upon the aforementioned methods, we integrate them to formulate the multi-level surrogate-assisted optimization algorithm. The primary procedure is outlined in Figure 2.

In Figure 2, line 1-5 describes the process of sampling. All the samples in the sampling set is generated by selected sampling method which is described in 3.2.1. Each of these samples is subsequently forwarded to the real complex model, which, in this problem, symbolizes the optimization objective. The model is executed to compute the objective function value for each sample. The key-value pairs <parameters, objective function value> will be utilized for constructing the global surrogate model in line 6. The random forest will fit the parameters and objective function values. Given any vector within the range of each parameter, random Forest can generate predictions for the current vector based on the fitting results. In line 7-12, the constructed global surrogate model will update by DE algorithm, during each iterative process, DE will explore the entire parameter space of the

Figure 2
The algorithm of the MLSAO

Algorithm 1 A multi-level surrogate-assisted optimization algorithm

1: Generate sampling set by Latin hypercube sampling method.
2: **for** $i = 1$ to *Sampling number* **do**
3: Input each sample to the real model as parameters.
4: Calculate corresponding value of objective function for each sample.
5: **end for**
6: Construct random forest global surrogate model $\hat{f}_g(x)$ based on the sampling set.
7: **for** $i = 1$; $i < maxiter_g$; $i++$ **do**
8: Find the optimal X_i of the surrogate model $\hat{f}_g(x)$ by DE algorithm;
9: Calculate the real model result $f(X_i)$ for X_i;
10: Sampling set=Sampling set + $(X_i, f(X_i))$;
11: Update surrogate model by new sampling set.
12: **end for**
13: Construct MARS local surrogate model $\hat{f}_l(x)$ based on the subset which contains the top n best data in the set.
14: **for** $i = 1$ to $maxiter_l$ **do**
15: Find the optimal X_i of the surrogate model $\hat{f}_l(x)$ by Nelder-Mead algorithm;
16: Calculate the real model result $f(X_i)$ for X_i;
17: Update the sampling set and global surrogate model.
18: **end for**
19: **return** The optimal parameter X_{min} and minimum objective function value $F(X_{min})$

surrogate model to search for the optimal solution of the surrogate model, which is then determined by DE. A new key-value pair is added to the sampling set established in line 1-5, then we update the surrogate model after increase in sample size. In this proposed method, we construct the local-level surrogate model once the global-level surrogate model has converged. Different from the global model. The role of the local surrogate model is to identify potential optimal solutions based on the outcomes of the global model. So that the sample size of the local model is much smaller, we choose the optimal subset of samples according to their objective function values. The subset will update during each iteration if generate better solution in line 14-18. At last, the optimal solution along with its corresponding objective function value is output as the final result, concluding the algorithm. The following of this section will provide detailed descriptions of the critical parts in each step.

4.2. Sampling

At the beginning of the algorithm, we select the latin hypercube sampling (LHS) [30] method to generate samples. The sampling method plays a crucial role in determining the quality of the sample set, and the excellence of the sample set directly influences the quality of the surrogate model. Various sampling methods may introduce distinct biases and variances in the sample set. Improper sampling methods may result in insufficient samples or excessive sample biases, thereby affecting the quality of the surrogate model. Hence, in the selection of a sampling method, factors such as sample distribution, sample size, and sample quality should be taken into account to ensure that the sampled data accurately reflect the characteristics of the original data. This, in turn, enhances the prediction accuracy of the surrogate model. Latin hypercube sampling (LHS) and Monte Carlo sampling [8] are suitable methods for building the surrogate model. They possess flexible sample sizes and excellent space-filling capabilities, efficiently covering the entire sample space with relatively few points [38]. LHS belongs to stratified sampling technique employed for random sample selection in multi-dimensional space. This method aims to better represent the population using fewer sampling points. It is an extension of the Latin square sampling technique and introduces the concept of hypercubes, which helps to distribute samples more evenly across the entire sample space.

One-dimensional Latin hypercube sampling involves dividing the cumulative density function (CDF) into n equal partitions and then selecting a random data point within each partition. Considering that each parameter follows a uniform distribution within its specified range, to obtain N samples, the cumulative density function of the uniform distribution is utilized. Each parameter is divided into N non-overlapping groups within its defined range, with each group having a probability of $1/N$ to be selected. Within each group's interval, a parameter value is randomly chosen. Following these rules, M vectors are generated, each representing the sampling results for one parameter and containing N elements. The goal is to create an $M*N$ matrix, where each row represents a sample point. By randomly selecting one element from each vector, a new M-dimensional vector is formed, resulting in a total of N vectors. This ensures that the sampling points are evenly distributed across the solution space.

4.3. Global Surrogate Model Construction

In the proposed algorithm, the global surrogate model is constructed based on random forest method, ran-

dom forest is a supervised ensemble learning method that belongs to the Bagging class and is implemented based on decision trees. It improves prediction accuracy by combining multiple decision trees. It is the most commonly used ensemble learning model. Firstly, the data set is sampled using the Bootstrap algorithm, and each group of data subsets is selected. Then, decision tree model parameters are set and trained on each data subset. Finally, these decision tree models are used to vote and obtain the results. The key to Random Forest is the decision tree structure, and the key to decision trees is the feature splitting method, i.e., how to determine the quality of the splitting features. Random Forest uses the out-of-bag error method, which does not use all samples when generating trees. Instead, a portion of the samples are reserved for validation. Approximately one-third of the samples in each decision tree are reserved for assessing the model's performance and calculating its error, not used during the model training process. The generation rules of each tree are as follows:

Initially, the parameter values are confirmed, where N represents the number of training cases (samples), and M represents the number of features. The parameter m is crucial for determining the decision at a tree node, and it should be much smaller than M. Next, the training set is formed by repeatedly sampling with replacement from the N training cases (bootstrap sampling). The unselected cases are then used for making predictions to evaluate their errors. For each node, m features are randomly chosen, and the decision at each node is based on these selected features. The best splitting method is determined based on these m features. Each tree is allowed to grow fully without pruning, although pruning may be considered after constructing a standard tree classifier.

In the proposed algorithm, the surrogate model plays a crucial role in establishing the mapping between parameters and the objective function. Instead of engaging in the lengthy process of calculation and simulation of the complex model, a mathematical statistical model is employed to efficiently generate the objective function values. Hence, during the surrogate model construction process, the essential key-value pairs <parameters, objective function value> are utilized as inputs and outputs for data fitting in the model. For a surrogate model constructed through random forest, given any set of parameter vectors that satisfy the parameter range constraints, the random forest surrogate model can generate simulated target function values based on the training.

4.4. Update Global Surrogate Model

In general, solving such complex problems relies on optimization algorithms that directly iterate in the real model. When we optimize a statistical model with relatively high accuracy using such algorithms, the iterations are almost cost-free. Achieving optimization effects requires running only a small number of real models. This is the primary advantage of surrogate models. However, constructing a surrogate model involves a limited number of sample points, and the initial surrogate model may inevitably contain some errors. Therefore, it is essential to judiciously increase the number of sampling points to continuously update the surrogate model, thereby reducing errors, improving simulation accuracy, and enhancing the representation of the real model. There are numerous methods for updating the surrogate model, including the minimum interpolating surface (MIS) [17], maximum expected improvement (MEI) [31], and candidate point approach (CAND) [29]. In this work, we are more concerned about the optimal values rather than the complete shape of the fitting surface. MIS is a method that does not concentrate on constructing the entire surrogate but rather focuses on the region containing the optimum. Hence, our focus is on ensuring the accuracy of the surrogate model in the vicinity of the optimal point. Employing the MIS method, we utilize DE to discover the optimal value of the surrogate model. These optimal parameters are subsequently applied to the real model to iteratively acquire new key-value pairs, enrich the sampling set, and augment the number of points surrounding the optimal point. This iterative process enhances the modeling accuracy in the proximity of the optimum.

4.5. Local Surrogate Model Construction

In this section, we select inverse distance weighting (IDW) [4] model to construct local-level surrogate model. It assigns weights to known values based on the inverse of their distances to the target location, resulting in a weighted average calculation for the unknown value. IDW assumes that closer points have a greater influence on the estimation, while distant points have less influence. The IDW method is as follows:

$$y = \begin{cases} \sum_{j=1}^{n} \frac{d(p,pt_j)}{\sum_{j=1}^{n} d(p,pt_j)} yt_j \ \ if\ p \neq pt_j \\ yt_j \ \ if\ p = x_j \end{cases}, \quad (11)$$

where p denotes the prediction input point, and y represents the prediction result of the point p. Dt_j belongs to the training set $T = \{yt_1, yt_2, ..., yt_n\}$, and yt_j is the output value for the j-th training point. The weighting function d is defined as:

$$d(p_u, p_v) = |p_u, p_v|_2^{-q},$$

where q is termed the power parameter, and it must be strictly greater than 1 to ensure the continuity of derivatives.

The local surrogate model is constructed to leverage the insights from the global surrogate model and explore potential improvements. Therefore, to enhance the accuracy of the surrogate model, careful consideration is given to both the quantity and quality of the selected points for constructing the model. For relatively small surrogate models, IDW has higher modeling performance. When constructing the model, we choose the top x fitness function-ranked points in the set to build the surrogate model and manually remove those poor quality solutions. This ensures the quality of the samples and allows the surrogate model to maintain a high level of simulation performance, enhancing its modeling ability in the region where the optimal solution may appear.

4.6. Get Final Optimal

Unlike the updating strategy of the global model, we want to quickly traverse the local surrogate model and find the possible optimal solution. Therefore, algorithms with simple processes, such as Nelder Mead Algorithm, are more suitable to use. Compared with algorithms such as DE, it can explore the optimal value of the current target function with fewer iterations. Similar to the updating of the global model, each iteration generates a new key-value pair. When a new point is added, we will re-queue and continue to select the top n points, rather than using a sampling set with an increasing number of points. This is crucial because a high-quality initial set for the surrogate model is essential to efficiently identify optimal solutions that might be challenging for the global model to uncover. When this step is completed, the algorithm also ends, and the obtained optimal solution is output.

5. Experiments Results

In this section, we aim to demonstrate the effectiveness of the proposed algorithm in addressing the optimization of complex model parameters. At first, some mathematical benchmark functions optimization experiments are conducted. Subsequently, a precipitation parameter tuning experiment with CESM confirmed the capability of the proposed algorithm in addressing intricate practical parameter optimization challenges. These experiments are supported by [5, 27, 37].

Table 1
Benchmark function description in this experiment

Function number	Function name	Global optimum
F1	Ellipsoid	0
F2	Rosenbrock	0
F3	Ackley	0
F4	Griewank	0
F5	Shifted Rotated Rastrigin (F10 in [34])	-330
F6	Rotated Hybrid composition function (F19 in [34])	10

5.1. Mathematical Function Benchmark Optimization Experiments

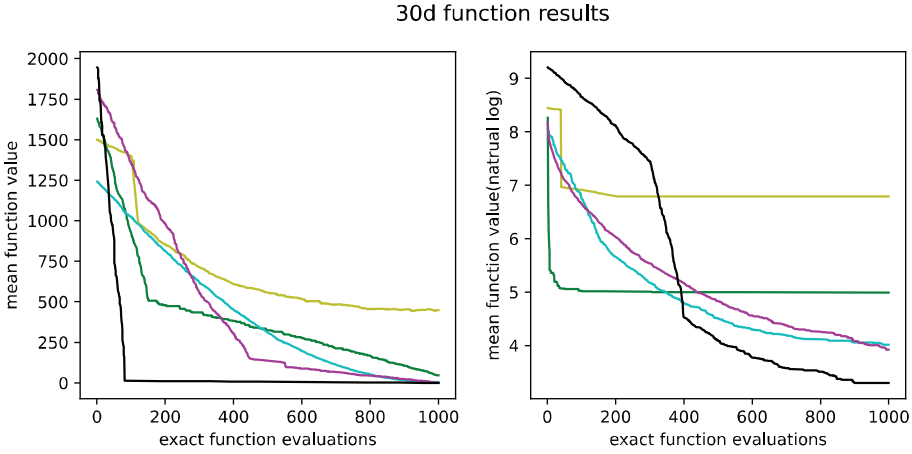
Benchmark functions with diverse characteristics are employed to assess the effectiveness of the MLSAO algorithm. We include GA [13], RBF-PSO, SA-COSO [35], and SAHO [26] algorithms, along with six problems of varying dimensions, to evaluate the performance of the MLSAO algorithm. These benchmark functions are detailed in Table 1. Each function undergoes optimization thirty times, and the mean value and standard deviation are calculated. All compared algorithms are implemented in Python.

5.1.1. Experimental Results on 30d Benchmark Problems

Table 2 and Figure 3 describe the performance of five algorithms on 30-dimensional problems. In the graphs, the horizontal axis represents the complexity of the problem, which in this experiment is the number of

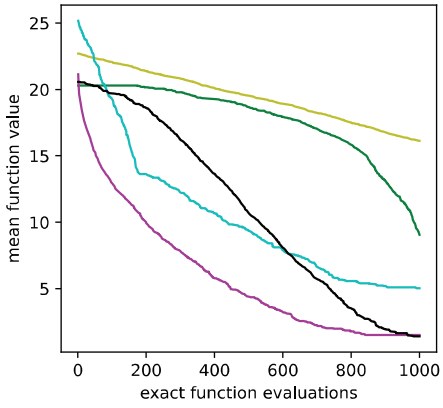
Figure 3

Convergence curves of GA, RBF-PSO, SA-COSO, SAHO, and MLSAO on 30D functions

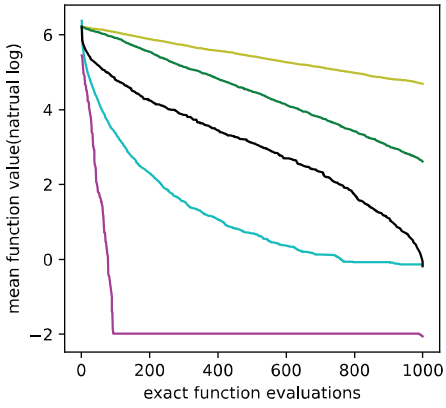


(a) 30D function1

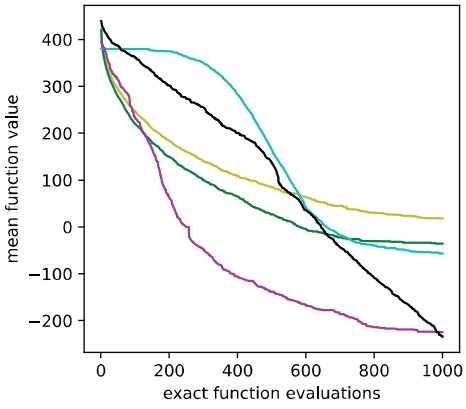
(b) 30D function2



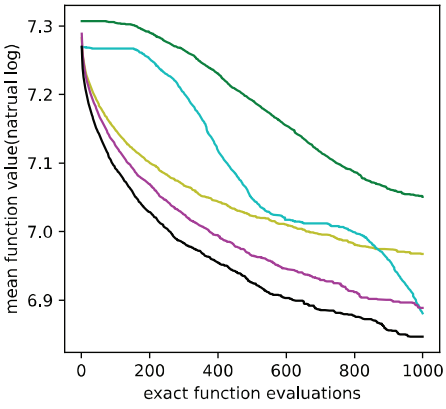
(c) 30D function3



(d) 30D function4



(e) 30D function5



(f) 30D function6

Table 2
Statistical comparisons of results on 30-dimensional benchmark problems

Function No	Algorithm	Mean	Std
F1	GA	4.4869E+02	1.1733E+02
	RBF-PSO	4.6740E+01	8.3591E+00
	SA-COSO	4.3620E+00	2.9579E+00
	SAHO	**1.3392E-01**	1.5735E-01
	MLSAO	3.1476E-01	2.2098E-01
F2	GA	8.4663E+02	1.7412E+02
	RBF-PSO	1.4769E+02	5.3282E+01
	SA-COSO	5.9851E+01	2.4556E+01
	SAHO	5.9070E+01	3.0016E+01
	MLSAO	**2.8445E+01**	4.7331E-01
F3	GA	1.6114E+01	1.1471E+00
	RBF-PSO	9.0326E+00	1.0429E+00
	SA-COSO	5.0152E+00	1.2214E+00
	SAHO	1.9901E+00	6.6895E-01
	MLSAO	**1.4361E+00**	6.3830E-01
F4	GA	1.0908E+02	3.0358E+01
	RBF-PSO	1.3664E+01	5.2129E+00
	SA-COSO	8.8940E-01	1.1844E-01
	SAHO	**1.2759E-01**	6.5921E-01
	MLSAO	8.2555E-01	8.9065E-02
F5	GA	1.8780E+02	4.0290E+01
	RBF-PSO	-3.5296E+01	3.7411E+01
	SA-COSO	-5.7357E+01	1.7545E+01
	SAHO	-2.1792E+02	3.7846E+01
	MLSAO	**-2.3456E+02**	2.2514E+01
F6	GA	1.0605E+03	5.1189E+01
	RBF-PSO	9.8923E+02	1.2191E+01
	SA-COSO	9.7358E+02	2.4022E+01
	SAHO	9.8201E+02	3.9012E+01
	MLSAO	9.4091E+02	1.1976E+01

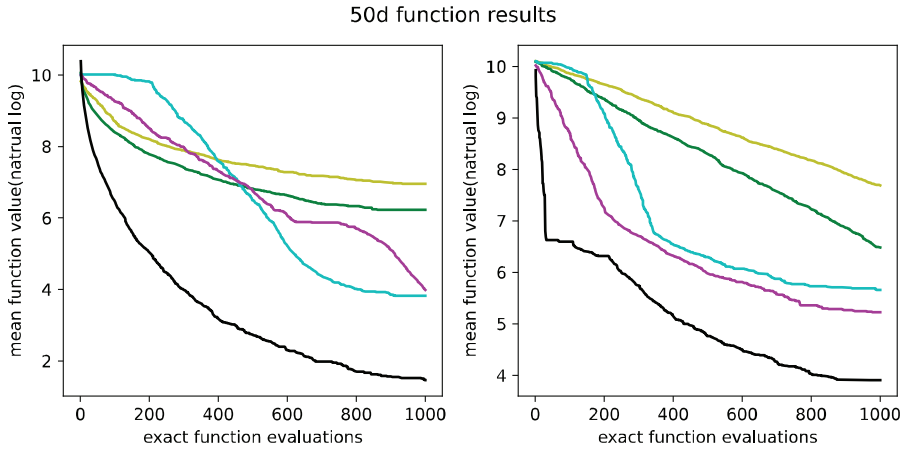
executions of benchmark test functions. It can be observed that in most test cases, MLSAO outperforms the other four methods. Among the six functions, MLSAO achieves the best average performance in four of them. For function F1, both SA-COSO and SAHO, as well as MLSAO, can find solutions close to the optimum. SAHO performs the best, with MLSAO slightly behind but significantly outperforming the other three control functions. The optimization results for F2 show that MLSAO performs significantly better than the other four algorithms, approaching the optimal solution and having the lowest standard deviation, indicating greater stability during optimization. Regarding the Ackley function, MLSAO and SAHO continue to perform the best and significantly outperform GA, RBF-PSO, and SA-COSO. Unlike the results for F1, MLSAO performs slightly better than SAHO in comparison to the control functions. The optimization results for F4 differ somewhat from the trends observed in the first three functions. SAHO performs the best, while MLSAO and SA-COSO achieve optimization results of a similar magnitude. This suggests that MLSAO still has room for improvement in this type of problem. For the more complex functions F5 and F6, MLSAO demonstrates superior performance and better exploration capabilities to varying degrees. Similar to F3, the optimization results for F5 show MLSAO slightly outperforming SAHO, with both significantly outperforming the other three algorithms. However, for the even more complex F6, MLSAO exhibits a significantly superior ability compared to the other algorithms. In summary, in the context of 30-dimensional problems, MLSAO demonstrates a significant advantage. Furthermore, GA, which does not utilize surrogate models, lags far behind the other algorithms, confirming the effectiveness of surrogate model-based approaches for solving complex problems.

5.1.2. Experimental Results on 50d Benchmark Problems

Compared to the results on 30-dimensional benchmark functions, MLSAO performs even better on 50-dimensional benchmark functions. This is evident in the results presented in Figure 4 and Table 3, where MLSAO consistently outperforms the other four algorithms, demonstrating superior optimization performance. In all optimization results, MLSAO outperforms the other four control algorithms significantly. For function F1, MLSAO's average fitness value is ap-

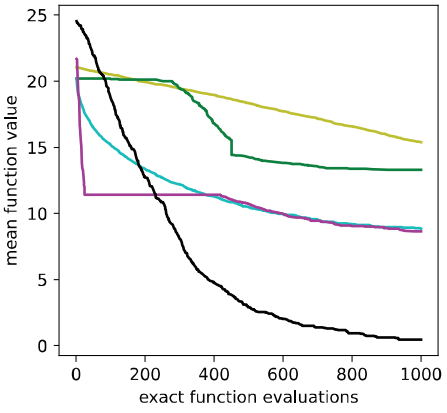
Figure 4

Convergence curves of GA, RBF-PSO, SA-COSO, SAHO, and MLSAO on 50D functions

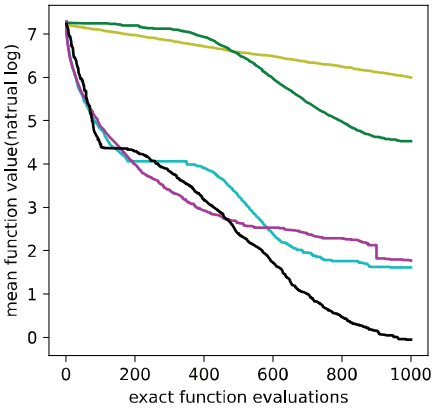


(a) 50D function1

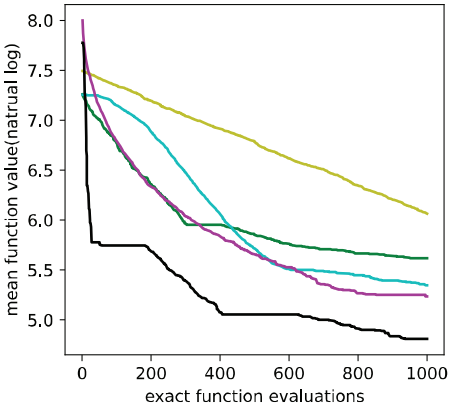
(b) 50D function2



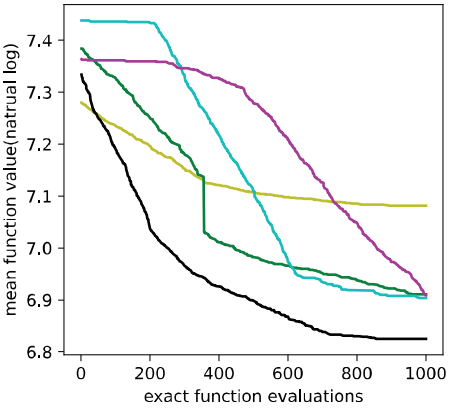
(c) 50D function3



(d) 50D function4



(e) 50D function5



(f) 50D function6

Table 3
Statistical comparisons of results on 50-dimensional benchmark problems

Function No	Algorithm	Mean	Std
F1	GA	2.2363E+03	4.3286E+02
	RBF-PSO	5.2168E+02	1.2364E+02
	SA-COSO	2.0327E+02	8.1653E+02
	SAHO	5.3468E+01	2.7324E+01
	MLSAO	**4.3287E+00**	2.6413e+00
F2	GA	2.1705E+03	7.1463E+02
	RBF_PSO	6.3548E+02	1.8932E+02
	SA-COSO	2.8811E+02	8.1744E+01
	SAHO	1.9358E+02	4.5258E+01
	MLSAO	**5.0308E+01**	4.0024E+00
F3	GA	1.5372E+01	3.6429E-01
	RBF-PSO	1.3290E+01	8.4063E-01
	SA-COSO	1.2346E+01	1.3853E+00
	SAHO	8.6818E+00	1.9303E+00
	MLSAO	**2.1369E+00**	9.6370E-01
F4	GA	4.0389E+02	4.4932E+01
	RBF-PSO	9.2733E+01	2.0329E+01
	SA-COSO	3.3176E+01	1.4464E+01
	SAHO	5.8982E+00	1.0977E+00
	MLSAO	**9.4104E-01**	7.8356E-02
F5	GA	4.2994E+02	6.4873E+01
	RBF_PSO	2.7548E+02	5.7724E+01
	SA-COSO	3.1036E+02	5.2617E+01
	SAHO	1.8805E+02	2.9981E+01
	MLSAO	**1.2672E+02**	2.8506E+02
F6	GA	1.1894E+03	4.5169E+01
	RBF_PSO	1.0067E+03	3.1456E+01
	SA-COSO	1.0818E+03	4.7503E+01
	SAHO	9.9971E+02	2.4517E+01
	MLSAO	**9.2985E+02**	1.4371E+01

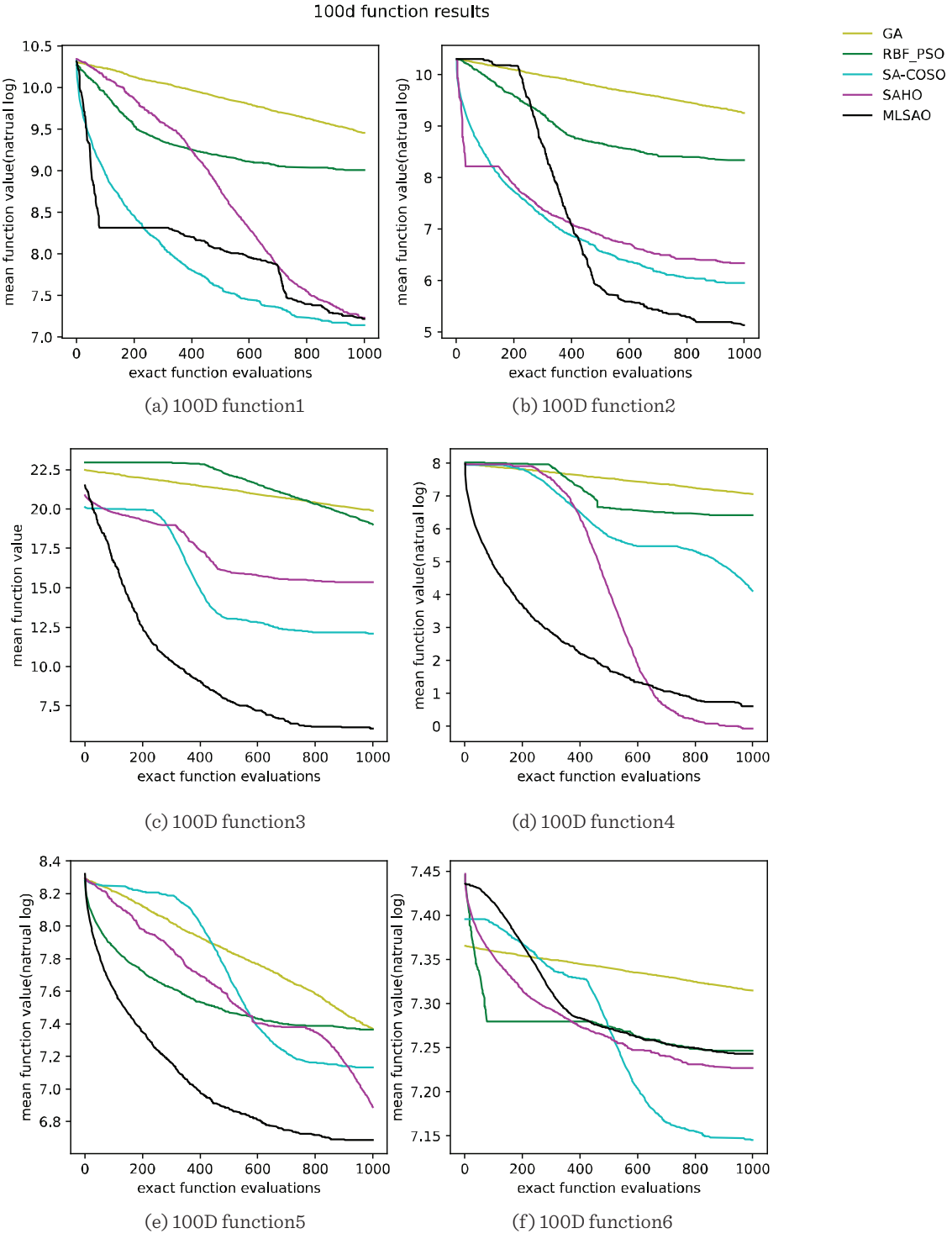
proximately 10% better than the second-best-performing algorithm, surpassing SA-COSO and SAHO by an order of magnitude, while both SA-COSO and SAHO outperform RBF-PSO by approximately 10%, and GA, the worst-performing algorithm, lags behind MLSAO by nearly three orders of magnitude. The optimization results for the Rosenbrock function reveal that MLSAO achieves the second-best average optimization results, which are approximately one-fourth of the SAHO algorithm's results, and MLSAO exhibits lower standard deviation. Similar patterns are observed in the optimization results for the F3 function. MLSAO's average optimization results are roughly one-fourth of the second-best algorithm's results, and MLSAO has a lower and more stable standard deviation. Among the other four algorithms, SAHO and SA-COSO perform similarly, while GA and RBF-PSO have similar results. MLSAO and SA-COSO perform the best on the F4 function, with SA-COSO's average results being approximately 5.65 times that of MLSAO. MLSAO significantly outperforms the other four algorithms in this test. For the two more complex functions, in F5, MLSAO's average optimization results are improved by 32.7% compared to the second-best algorithm, and in F6, this improvement is 6.75%. MLSAO also demonstrates better stability with lower standard deviations in both cases. In summary, MLSAO exhibits excellent optimization performance on 50-dimensional benchmark functions, consistently achieving the best optimization performance to varying degrees in each function. This advantage is particularly pronounced in most functions. Therefore, we consider MLSAO to be better suited for optimizing problems of mid-dimension complexity problems.

5.1.3. Experimental Results on 100d Benchmark Problems

We attempted to challenge more complex application scenarios by testing various algorithms' performance in solving high-dimensional problems using 100-dimensional benchmark functions. Table 4 displays the average best fitness function values obtained by the five algorithms after 30 independent runs, highlighting the best average values for each test function. MLSAO achieved the best fitness function values in F2, F3, and F5, while SA-COSO performed the best in F1 and F6, and SAHO performed better in F4. Figure 5 shows the fitness changes of each algorithm on different test functions. Combined with the table,

Figure 5

Convergence curves of GA, RBF-PSO, SA-COSO, SAHO, and MLSAO on 100D functions



SA-COSO performed the best in the optimization experiment for F1, but MLSAO and SAHO were not far behind, with all three almost on the same level. MLSAO demonstrated a significant advantage in the 100-dimensional F2 function test, with an average optimization result far higher than the other four functions and the smallest standard deviation. The optimization results for the F3 function also favored MLSAO, where the average result for the second-best SAHO was approximately 2.5 times that of the best result. The results for the other four algorithms were relatively similar. The best optimization result for the 100-dimensional Griewank function belonged to SAHO. The difference between these two algorithms and the other three was particularly pronounced. In the experiment for F5, MLSAO performed the best, followed by SHAO, with both achieving average results below 1000, while the results for the other three algorithms were all above 1000. The experimental results for the F6 function showed that the average results for all five algorithms were generally within the same range. However, SA-COSO performed the best, while MLSAO ranked only third, with its performance being less satisfactory. Overall, MLSAO achieved almost the best fitness function values or equivalent levels in most functions. Even in cases where MLSAO did not achieve the optimal solution, it still managed to come very close to an approximate optimal result to a large extent. However, in F6, the surrogate model may have difficulty providing useful information, leading to the algorithm being trapped in a local optimum and unable to discover the true global optimum. Overall, MLSAO still performed well on most 100-dimensional problems, demonstrating its ability to handle high-dimensional complex problems and proving its suitability for high-dimensional scenario.

5.1.3. Experimental Results on 200d Benchmark Problems

To further assess the capability of MLSAO in addressing high-dimensional and complex problems, we conducted tests with several benchmark functions selected from Table 1, using a parameter dimension of 200. We compared MLSAO with SAHSO[19]and GL-SADE [39], and the results are shown in Table 5. Despite suboptimal performance in F1, the outcomes for F2 and F4 indicate that MLSAO remains the algorithm with the strongest optimization capability.

Table 4
Statistical comparisons of results on 100-dimensional benchmark problems

Function No	Algorithm	Mean	Std
F1	GA	1.2766E+04	1.5483E+03
	RBF_PSO	8.2347E+03	9.0105E+02
	SA-COSO	**1.3224E+03**	3.0408E+02
	SHAO	1.3861E+03	1.4210E+02
	MLSAO	1.3724E+03	2.3379E+01
F2	GA	1.0426E+04	1.0697E+03
	RBF_PSO	4.1645E+03	5.9803E+02
	SA-COSO	4.0199E+02	1.8129E+02
	SAHO	5.7284E+02	4.3681E+01
	MLSAO	**1.7192E+02**	2.7322E+01
F3	GA	1.9894E+01	4.2056E-01
	RBF_PSO	1.9021E+01	4.3126E-01
	SA-COSO	1.2127E+01	8.9256E-01
	SAHO	1.5396E+01	4.8945E-01
	MLSAO	**6.1134E+00**	6.0183E-01
F4	GA	1.1684E+03	1.0013E+02
	RBF_PSO	6.1963E+02	5.7765E+01
	SA-COSO	6.1253E+01	2.0021E+01
	SAHO	**9.3503E-01**	1.0853E-01
	MLSAO	1.9258E+00	2.0905E-01
F5	GA	2.1895E+03	2.5614E+02
	RBF_PSO	1.5842E+03	2.1104E+02
	SA-COSO	1.2561E+03	1.1033E+02
	SAHO	9.7908E+02	7.9437E+01
	MLSAO	**8.1588E+02**	8.9033E+01
F6	GA	1.5017E+03	3.1302E+01
	RBF_PSO	1.4035E+03	4.7821E+01
	SA-COSO	**1.2677E+03**	2.5304E+01
	SAHO	1.3793E+03	1.1042E+02
	MLSAO	1.3998E+03	3.9211E+01

Table 5
Statistical comparisons of results on 200-dimensional benchmark problems

Function No	Algorithm	Mean	Std
F1	SAHSO	8.5726E+02	9..8398E+01
	GL-SADE	**5.1297E+02**	8.9146E+01
	MLSAO	2.7524E+03	1.8329E+02
F2	SAHSO	9.1682E+02	1.7957E+02
	GL-SADE	7.3085E+02	5.9803E+01
	MLSAO	**4.3583E+02**	5.6634E+01
F3	SAHSO	**7.5684E+00**	7.3696E-01
	GL-SADE	2.0751E+01	3.0311E-01
	MLSAO	8.7712E+00	8.9768E-01
F4	SAHSO	2.7841E+00	3.4071E-01
	GL-SADE	3.9983E+00	5.0702E-01
	MLSAO	**2.6304E+00**	3.0015E-01

In comparison with SAHSO, F3 shows only a slight disadvantage than SAHSO. Overall, compared to A and B, MLSAO still achieves relatively better optimization results in the majority of the experimental optimization functions. These results demonstrate that MLSAO still possesses sufficient optimization prowess when dealing with high-dimensional and complex computational problems.

5.2. CESM Parameter Tuning Experiments

The Community Earth System Model (CESM) [18] is a fully coupled global climate model that offers advanced computer simulations of Earth's historical, current, and future climate conditions. The community atmosphere model (CAM) is the atmosphere component of the community earth system model. It simulates various chemical reactions and energy exchange in the atmosphere. CESM simulation experience is a high-performance computing program that requires a significant amount of time and computational cost. A complete simulation process of CESM usually takes more than ten hours or even several days.

The precipitation process in CAM is complicated and precipitation-related processes is related to many parameterization schemes, include cumulus convection processes, large-scale circulation, microphysical processes, boundary layer processes, etc. These schemes contain some adjustable parameters, which can be tuned to change the simulation result of CAM. The computational and time cost generated by traditional algorithms are unacceptable during the optimization process. Therefore, we applied the proposed algorithm to address the CESM parameter tuning problem. The results demonstrate that the proposed algorithm effectively enhanced the precipitation simulation results of the Community Atmosphere Model (CAM) in several regions with a relatively small number of iterations.

In this work, we use the proposed MLSAO algorithm to optimize the precipitation related parameters of CAM, so as to make the optimization results of the CAM model more close to the observation data. This algorithm can greatly reduce the optimization time by building a surrogate model to replace the complicated CAM model running process. Therefore, we must account for the disparities between the model simulation results and the observational data in the objective function. To guide the algorithm towards optimizing by minimizing these disparities, we have chosen the root-mean-square error (RMSE) as the objective function. The RMSE is calculated as follows:

$$\mathrm{RMSE} = \sqrt{\frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}}(\mathrm{mod_i} - \mathrm{obs_i})^2},$$

where N is the total number of the grid points of simulation region, mod_i and obs_i are the model simulated and observation data values at grid point i. A smaller RMSE indicates a smaller error between model simulation and observational data. The optimization goal is to minimize the RMSE for each region. The parameters are shown in Table 6, and the selected regions are presented in Table 7. These parameters are chosenfrom the Zhang-McFarlane parameterization scheme [51].

The optimization results are shown in Figure 6, the numerical results indicate that the RMSE of selected regions has decreased compared to the default experiment, indicating that the optimized experimental results are closer to the observed data. The RMSE for

Table 6
Parameters description of Zhang-McFarlane parameterization scheme. CAPE means the convective available potential energy

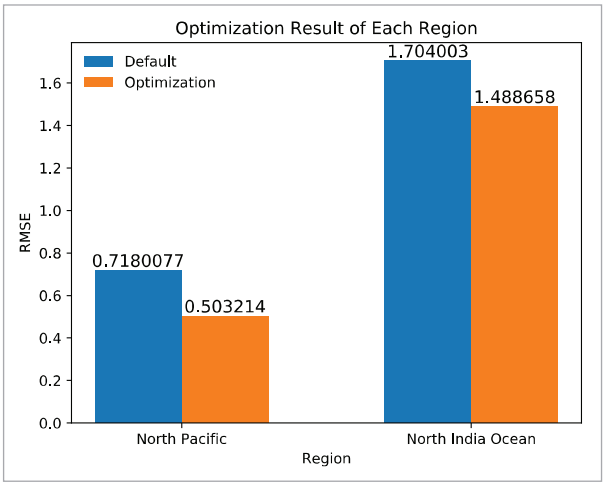
Parameter name	Meaning	Range	Default value
zmconv_dmpdz	Parcel fractional mass entrainment rate	$-2.0 \times 10^{-3} \sim -0.2 \times 10^{-3}$	-1.0×10^{-3}
zmconv_c0_ocn	Deep convection precipitation efficiency over ocean	$1.0 \times 10^{-3} \sim 0.1$	0.045
zmconv_tau	Time scale for consumption rate deep CAPE	1800~28800	3600

Table 7
Regions selected in this study and their range

Name	Gegion
North Pacific Ocean	15°-40°N, 150°-210°E
North India	0°-20°N, 50°-90°E

the two regions has decreased by 29.77% and 12.63%, respectively. To demonstrate that the decrease of RMSE has indeed improved the accuracy of precipitation simulation, we draw precipitation distribution plots for each region, detailing the changes in precipitation over these regions caused by optimized parameters obtained from the proposed algorithm. The results for the North Indian Ocean, and the North Pacific are shown in Figures 6-7, respectively. We will now introduce the changes in precipitation compared to the default parameters.

Figure 6
RMSE of default experience and optimization results over each region

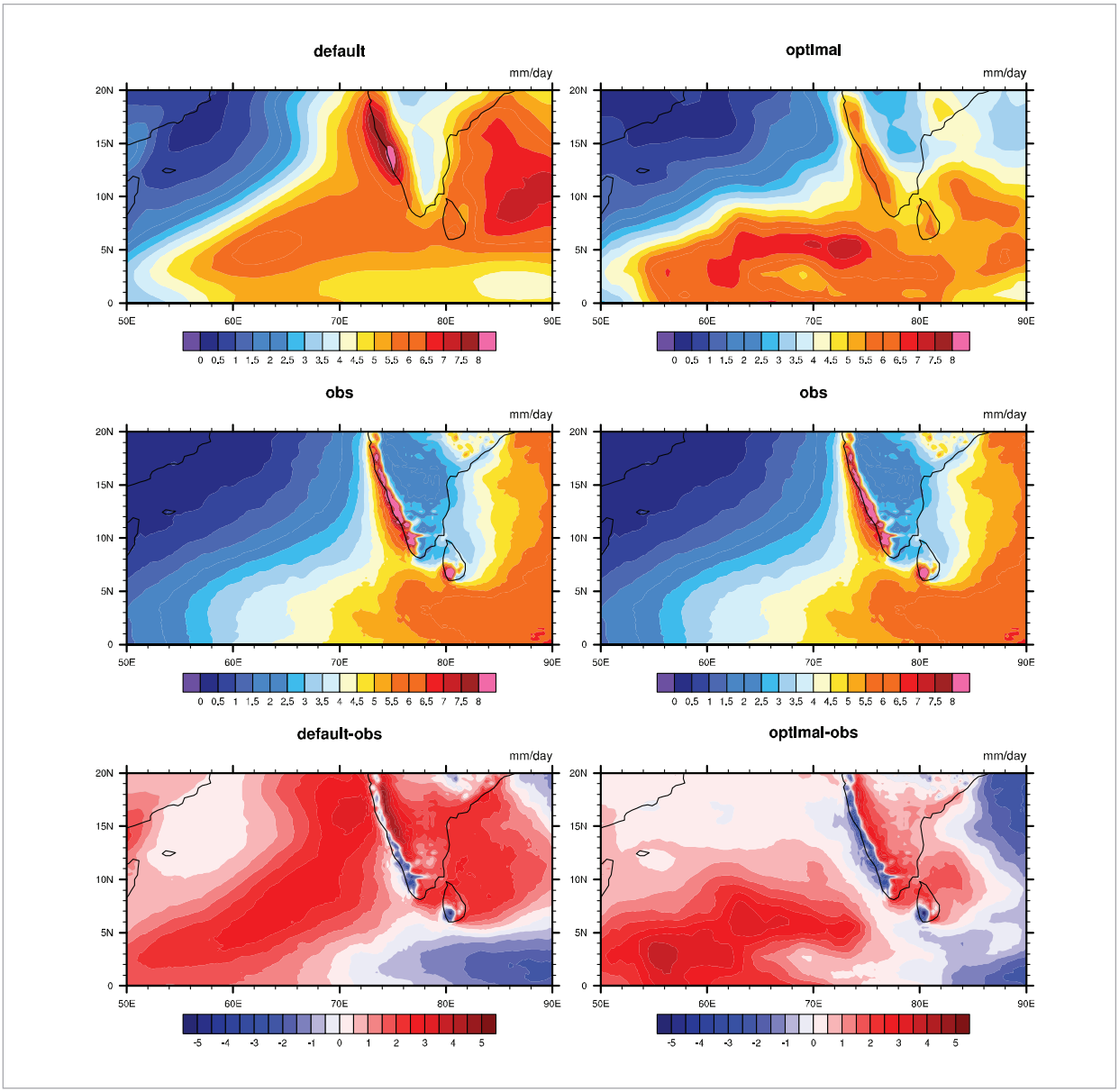


The improvement in the North Indian Ocean region is quite evident, as shown in the Figure 7. In the default experiment, it is noticeable that the precipitation in the ocean areas on both sides of the Indian Peninsula is considerably higher than the observed data, resulting in a significant positive error. In contrast, in the optimized experiment, these errors have been significantly reduced. This difference is especially prominent on the western side of the peninsula, where precipitation values now align more closely with those observed. The positive error areas on eastern side of the peninsula have also been significantly reduced, with some remain ing errors near the island of SriLanka. Furthermore, near the Arabian Peninsula, the results of the optimized experiment are also superior to the default experiment. The optimized parameters have reduced precipitation in this region, even eliminating the positive errors that were present in the default experiment. In the southeastern marine area, unlike the improvement seen in other regions, the optimized experiment has increased precipitation in this area, bringing the results closer to the observed data. In summary, in the North Indian Ocean region, the default experiment had significant positive errors. Our improvements have effectively reduced most of these positive errors without introducing new errors.

Figure 8 illustrates the optimization results for the North Pacific region. Similar to the North Indian Ocean region, the default experiment still exhibits a significant amount of positive errors. Our optimization results aim to reduce these positive errors in the North Pacific region. As shown in the figure, our optimization results have substantially reduced precipitation in these ocean areas, primarily in the central region of the selected area. Precipitation in this central area has been significantly reduced and is now very close to the results from observed data. The re-

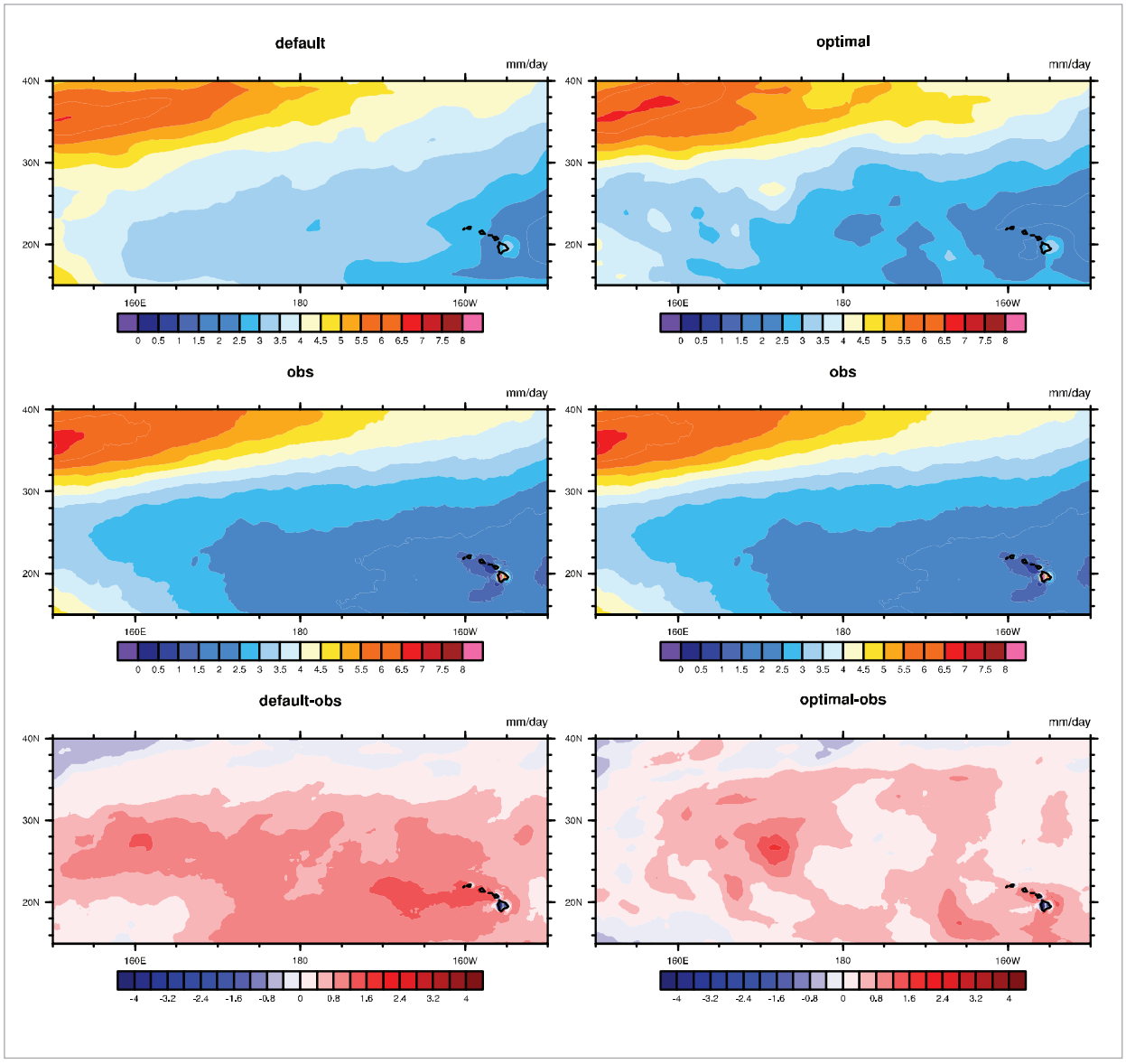
Figure 7

The precipitation distribution of the North India optimization result. The left column displays the default experiment, observation data, and the difference between the default experiment and the observation data from top to bottom. The right column exhibits the optimal experiment, observation data, and the difference between the optimal experiment and the observation data from top to bottom



sults for the eastern and western sides of the region have also seen considerable improvements. Additionally, in the southeastern areas around the islands, precipitation has noticeably decreased. Furthermore, in the northwestern region, the default experiment underestimated precipitation, while the optimized experiment increased precipitation in this area, eliminating negative errors. It is evident that the optimized parameters can enhance the capabilities of CAM in the North Pacific region.

Figure 8
The precipitation distribution of the North Pacific optimization result. The left column displays the default experiment, observation data, and the difference between the default experiment and the observation data from top to bottom. The right column exhibits the optimal experiment, observation data, and the difference between the optimal experiment and the observation data from top to bottom



6. Conclusions

In this work, we propose a multi-level surrogate-assisted optimization algorithm MLSAO, primarily designed to address the parameter optimization challenges associated with complex models. MLSAO introduces a multi-level search mechanism that enhances its exploration capability of the parameter space, thereby mitigating the risk of falling into local optima to some extent. The proposed MLSAO integrates random forest and DE to construct and update the global level surrogate model, respectively. For lo-

cal level, we select IDW to build the surrogate model and simplex downhill method to update this surrogate model. Whether it is the optimization tests on benchmarks or the practical application of CESM parameter tuning, MLSAO has demonstrated excellent optimization capabilities.

In future work, we aim to broaden the application scope of MLSAO, enhance its capability to tackle multi-objective optimization, and facilitate its utilization in a wider range of domains to address real-world optimization challenges. Additionally, as data scales continue to increase, we aim to improve its optimization efficiency. Moreover, we are considering the introduction of parallel and multi-threading techniques to enhance the efficiency and utilization of computational resources for MLSAO.

Competing Interests

The authors have no relevant financial or non-financial interests to disclose.

Funding

This work is funded by: National Key R&D Plan of China under Grant No. 2017YFA0604500, and by Key scientific and technological R&D Plan of Jilin Province of China under Grant No. 20180201103GX.

References

1. Alizadeh, R., Allen, J. K., Mistree, F. Managing Computational Complexity Using Surrogate Models: A Critical Review. Research in Engineering Design, 2020, 31, 275-298. https://doi.org/10.1007/s00163-020-00336-7
2. Anderson, G. J., Lucas, D. D. Machine Learning Predictions of a Multiresolution Climate Model Ensemble. Geophysical Research Letters, 2018, 45, 4273-4280. https://doi.org/10.1029/2018GL077049
3. Aria, M., Cuccurullo, C., Gnasso, A. A Comparison Among Interpretative Proposals for Random Forests. Machine Learning with Applications, 2021, 6, 100094. https://doi.org/10.1016/j.mlwa.2021.100094
4. Bemporad, A. Active Learning for Regression by Inverse Distance Weighting. Information Sciences, 2023, 626, 275-292. https://doi.org/10.1016/j.ins.2023.01.028
5. Bouhlel, M. A., Hwang, J. T., Bartoli, N., Lafage, R., Morlier, J., Martins, J. R. R. A. A Python Surrogate Modeling Framework with Derivatives. Advances in Engineering Software, 2019, p. 102662. https://doi.org/10.1016/j.advengsoft.2019.03.005
6. Briffoteaux, G., Gobert, M., Ragonnet, R., Gmys, J., Mezmaz, M., Melab, N., Tuyttens, D. Parallel Surrogate-assisted Optimization: Batched Bayesian Neural Network-assisted GA versus q-EGO. Swarm and Evolutionary Computation, 2020, 57, 100717.https://doi.org/10.1016/j.swevo.2020.100717
7. Chu, L., Shi, J., Souza de Cursi, E. Kriging Surrogate Model for Resonance Frequency Analysis of Dental Implants by a Latin Hypercube-Based Finite Element Method. Applied Bionics and Biomechanics, 2019, 2019. https://doi.org/10.1155/2019/3768695
8. Dalgaty, T., Castellani, N., Turck, C., Harabi, K.-E., Querlioz, D., Vianello, E. In Situ Learning Using Intrinsic Memristor Variability via Markov Chain Monte Carlo Sampling. Nature Electronics, 2021, 4(2), 151-161. https://doi.org/10.1038/s41928-020-00523-3
9. Diaz-Manriquez, A., Toscano, G., Coello Coello, C. A. Comparison of Metamodeling Techniques in Evolutionary Algorithms. Soft Computing, 2017, 21, 5647-5663. https://doi.org/10.1007/s00500-016-2140-z
10. Fan, F. L., Xiong, J. J., Li, M. Z., Wamh, G. On Interpretability of Artificial Neural Networks: A Survey. IEEE Transactions on Radiation and Plasma Medical Sciences, 2021, 5(6): 741-760. https://doi.org/10.1109/TRPMS.2021.3066428
11. Fuhg, J. N., Fau, A., Nackenhorst, U. State-of-the-Art and Comparative Review of Adaptive Sampling Methods for Kriging. Archives of Computational Methods in Engineering, 2021, 28: 2689-2747. https://doi.org/10.1007/s11831-020-09474-6
12. Golzari, A., Haghighat Sefat, M., Jamshidi, S. Development of an Adaptive Surrogate Model for Production Optimization. Journal of Petroleum Science and Engineering, 2015, 133, 677-688. https://doi.org/10.1016/j.petrol.2015.07.012
13. Holland, J. H. Genetic Algorithms. Scientific American, 1992, 267, 66-73. https://doi.org/10.1038/scientificamerican0792-66
14. Hwang, J. T. A Fast-Prediction Surrogate Model for Large Datasets. Aerospace Science and Technology, 2018. https://doi.org/10.1016/j.ast.2017.12.030
15. Jiang, K., Wang, J., Chen, Z., Zhou, Q., Jiang, S.-H. Influence of Coupling Factors on Structural Reliability Ba-

sed on Polynomial Response Surface Optimization Model. Advances in Mechanical Engineering, 2023, 15(8). https://doi.org/10.1177/16878132231184145

16. Jie, H., Wu, Y., Zhao, J., Ding, J., Liangliang. An Efficient Multi-Objective PSO Algorithm Assisted by Kriging Metamodel for Expensive Black-Box Problems. Journal of Global Optimization, 2017, 67, 399-423. https://doi.org/10.1007/s10898-016-0428-2
17. Jones, D. R. A Taxonomy of Global Optimization Methods Based on Response Surfaces. Journal of Global Optimization, 2001, 21, 345-383. https://doi.org/10.1023/A:1012771025575
18. Kay, J. E., Deser, C., Phillips, A., Mai, A., Hannay, C., Strand, G., Arblaster, J. M., Bates, S. C., Danabasoglu, G., Edwards, J., Holland, M., Kushner, P., Lamarque, J.-F. ., Lawrence, D., Lindsay, K., Middleton, A., Munoz, E., Neale, R., Oleson, K., Polvani, L. The Community Earth System Model (CESM) Large Ensemble Project: A Community Resource for Studying Climate Change in the Presence of Internal Climate Variability. Bulletin of the American Meteorological Society, 2015, 96(8), 1333-1349. https://doi.org/10.1175/BAMS-D-13-00255.1
19. Li, F., Li, Y., Cai, X. Gao, L. A Surrogate-Assisted Hybrid Swarm Optimization Algorithm for High-Dimensional Computationally Expensive Problems. Swarm and Evolutionary Computation, 2022, 72, 101096. https://doi.org/10.1016/j.swevo.2022.101096
20. Liu, Y., Liu, J., Ding, J., Yang, S., Jin, Y. A Surrogate-Assisted Differential Evolution with Knowledge Transfer for Expensive Incremental Optimization Problems. IEEE Transactions on Evolutionary Computation, 2023, 1-1. https://doi.org/10.1109/TEVC.2023.3291697
21. Liu, Y., Liu, J., Tan, S., Yang, Y., Li, F. A Bagging-Based Surrogate-Assisted Evolutionary Algorithm for Expensive Multi-Objective Optimization. Neural Computing and Applications, 2022, 34, 12097-12118. https://doi.org/10.1007/s00521-022-07097-5
22. Li, Y., Zhong, J. HAS-EA: A Fast Parallel Surrogate-Assisted Evolutionary Algorithm. Memetic Computing, 2022. https://doi.org/10.1007/s12293-022-00376-7
23. Müller, J., Paudel, R., Shoemaker, C. A., Woodbury, J., Wang, Y. Mahowald, N. M. CH4 Parameter Estimation in CLM4.5bgc Using Surrogate Global Optimization. Geoscientific Model Development, 2015, 8(10), 3285-3310. https://doi.org/10.5194/gmd-8-3285-2015
24. Neelin, J. D., Bracco, A., Luo, H., McWilliams, J. C., Meyerson, J. E. Considerations for Parameter Optimization and Sensitivity in Climate Models. Proceedings of the National Academy of Sciences, 2010, 107, 21349-21354. https://doi.org/10.1073/pnas.1015473107
25. Nelder, J. A., Mead, R. A Simplex Method for Function Minimization. The Computer Journal, 1965, 7, 308-313. https://doi.org/10.1093/comjnl/7.4.308
26. Pan, J. S., Liu, N., Chu, S. C., Lai, T. An Efficient Surrogate-Assisted Hybrid Optimization Algorithm for Expensive Optimization Problems. Information Sciences, 2021, 561, 304-325. https://doi.org/10.1016/j.ins.2020.11.056
27. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Louppe, G., Prettenhofer, P., Weiss, R., Weiss, R. J., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E. Scikit-Learn: Machine Learning in Python. J. Mach. Learn. Res, 2011. https://doi.org/10.5555/1953048.2078195.
28. Rasamoelina, A. D., Adjailia, F., Sincak, P. A Review of Activation Function for Artificial Neural Network. 2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI), 2020. https://doi.org/10.1109/SAMI48414.2020.9108717
29. Regis, R. G., Shoemaker, C. A. A Stochastic Radial Basis Function Method for the Global Optimization of Expensive Functions. INFORMS Journal on Computing, 2007, 19, 497-509. https://doi.org/10.1287/ijoc.1060.0182
30. Saurette, Daniel D., Asim Biswas, Adam W. Gillespie. Determining Minimum Sample Size for the Conditioned Latin Hypercube Sampling Algorithm. Pedosphere, 2022. https://doi.org/10.1016/j.pedsph.2022.09.001
31. Schonlau, M., Welch, W. J., Jones, D. R. Global Versus Local Search in Constrained Optimization of Computer Models. Lecture Notes-Monograph Series, 1998, pp. 11-25. https://doi.org/10.1214/lnms/1215456182
32. Shi, R., Liu, L., Long, T., Wu, Y., Wang, G. G. Multidisciplinary Modeling and Surrogate Assisted Optimization for Satellite Constellation Systems. Structural and Multidisciplinary Optimization, 2018, 58, 2173-2188. https://doi.org/10.1007/s00158-018-2032-1
33. Storn, R., Price, K. Differential Evolution-A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization, 1997, 11, 341. https://doi.org/10.1023/A:1008202821328
34. Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y. P., Auger, A., Tiwari, S. Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization. KanGAL Report, 2005, 2005005.

35. Sun, C., Jin, Y., Cheng, R., Ding, J., Zeng, J. Surrogate-Assisted Cooperative Swarm Optimization of High-Dimensional Expensive Problems. IEEE Transactions on Evolutionary Computation, 2017, 21, 644-660. https://doi.org/10.1109/TEVC.2017.2675628

36. Sun, C., Jin, Y., Zeng, J., Yu, Y. A Two-Layer Surrogate-Assisted Particle Swarm Optimization Algorithm. Soft Computing, 2015, 19, 1461-1475. https://doi.org/10.1007/s00500-014-1283-z

37. Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J.SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 2020, 17(3), 261-272. https://doi.org/10.1038/s41592-019-0686-2

38. Wang, C., Duan, Q., Gong, W., Ye, A., Di, Z., Miao, C. An Evaluation of Adaptive Surrogate Modeling Based Optimization with Two Benchmark Problems. Environmental Modelling & Software, 2014, 60, 167-179. https://doi.org/10.1016/j.envsoft.2014.05.026

39. Wang, W., Liu, H. L., Tan, K. C. A Surrogate-Assisted Differential Evolution Algorithm for High-Dimensional Expensive Optimization Problems. IEEE Transactions on Cybernetics, 2023, 53(4), 2685-2697. https://doi.org/10.1109/TCYB.2022.3175533

40. Wang, X., Wang, G. G., Song, B., Wang, P., Wang, Y. A Novel Evolutionary Sampling Assisted Optimization Method for Highdimensional Expensive Problems. IEEE Transactions on Evolutionary Computation, 2019, 23, 815-827. https://doi.org/10.1109/TEVC.2019.2890818

41. Williams, B., Cremaschi, S. Selection of Surrogate Modeling Techniques for Surface Approximation and Surrogate-Based Optimization. Chemical Engineering Research and Design, 2021, 170, 76-89. https://doi.org/10.1016/j.cherd.2021.03.028

42. Ji, X., Zhang, Y., He, C., Cheng, J., Gong, D., Gao, X., Guo, Y. Surrogate and Autoencoder-Assisted Multitask Particle Swarm Optimization for High-Dimensional Expensive Multimodal Problems. IEEE Transactions on Evolutionary Computation, 2023, 1-1. https://doi.org/10.1109/TEVC.2023.3287213

43. Xing, J., Luo, Y., Gao, Z. A Global Optimization Strategy Based on the Kriging Surrogate Model and Parallel Computing. Structural and Multidisciplinary Optimization, 2020, 62(1), 405-417. https://doi.org/10.1007/s00158-020-02495-6

44. Xu, H., Zhang, T., Luo, Y., Huang, X., Xue, W. Parameter Calibration in Global Soil Carbon Models Using Surrogate-Based Optimization. Geoscientific Model Development, 2018, 11, 3027-3044. https://doi.org/10.5194/gmd-11-3027-2018

45. Yan, C., Yin, Z., Shen, X., Mi, D., Guo, F., Long, D. Surrogate-Based Optimization with Improved Support Vector Regression for Non-Circular Vent Hole on Aero-Engine Turbine Disk. Aerospace Science and Technology, 2020, 96, 105332. https://doi.org/10.1016/j.ast.2019.105332

46. You, X., Li, W., Chai, Y. A Truly Meshfree Method for Solving Acoustic Problems Using Local Weak Form and Radial Basis Functions. Applied Mathematics and Computation, 2020, 365, 124694. https://doi.org/10.1016/j.amc.2019.124694

47. Yu, H., Tan, Y., Zeng, J., Sun, C., Jin, Y. Surrogate-Assisted Hierarchical Particle Swarm Optimization. Information Sciences, 2018, 454-455, 59-72. https://doi.org/10.1016/j.ins.2018.04.062

48. Shang, Y., Nogal, M., Teixeira, R., Wolfert, A. R. (Rogier) M. Optimal Design of Rail Level Crossings and Associated Transition Zones Using Adaptive Surrogate-Assisted Optimization. Engineering Structures, 2023, 282, 115740. https://doi.org/10.1016/j.engstruct.2023.115740

49. Zhang, C., Di, Z., Duan, Q., Xie, Z., Gong, W. Improved Land Evapotranspiration Simulation of the Community Land Model Using a Surrogate-Based Automatic Parameter Optimization Method. Water, 2020, 12, 943. https://doi.org/10.3390/w12040943

50. Zhang, F., Lauren, J. O'Donnell. Support Vector Regression. Machine Learning. Academic Press, 2020: 123-140. https://doi.org/10.1016/B978-0-12-815739-8.00007-9

51. Zhang, G. J., McFarlane, N. A. Sensitivity of Climate Simulations to the Parameterization of Cumulus Convection in the Canadian Climate Centre General Circulation Model. Atmosphere-Ocean, 1995, 33, 407-446. https://doi.org/10.1080/07055900.1995.9649539