Complex & Intelligent Systems (2024) 10:2129–2149
https://doi.org/10.1007/s40747-023-01262-6

ORIGINAL ARTICLE

Cooperative coevolutionary surrogate ensemble-assisted differential evolution with efficient dual differential grouping for large-scale expensive optimization problems

Rui Zhong[1] · Enzhi Zhang[1] · Masaharu Munetomo[2]

Received: 19 April 2023 / Accepted: 29 September 2023 / Published online: 26 October 2023

Abstract
This paper proposes a novel algorithm named surrogate ensemble assisted differential evolution with efficient dual differential grouping (SEADECC-EDDG) to deal with large-scale expensive optimization problems (LSEOPs) based on the CC framework. In the decomposition phase, our proposed EDDG inherits the framework of efficient recursive differential grouping (ERDG) and embeds the multiplicative interaction identification technique of Dual DG (DDG), which can detect the additive and multiplicative interactions simultaneously without extra fitness evaluation consumption. Inspired by RDG2 and RDG3, we design the adaptive determination threshold and further decompose relatively large-scale sub-components to alleviate the curse of dimensionality. In the optimization phase, the SEADE is adopted as the basic optimizer, where the global and the local surrogate model are constructed by generalized regression neural network (GRNN) with all historical samples and Gaussian process regression (GPR) with recent samples. Expected improvement (EI) infill sampling criterion cooperated with random search is employed to search elite solutions in the surrogate model. To evaluate the performance of our proposal, we implement comprehensive experiments on CEC2013 benchmark functions compared with state-of-the-art decomposition techniques. Experimental and statistical results show that our proposed EDDG is competitive with these advanced decomposition techniques, and the introduction of SEADE can accelerate the convergence of optimization significantly.

Keywords Cooperative coevolution (CC) · Large-scale expensive optimization problems (LSEOPs) · Efficient dual differential grouping (EDDG) · Surrogate ensemble assisted differential evolution (SEADE)

Introduction

As the development of the evolutionary computation (EC) community, large-scale optimization problems (LSOPs) become more and more ubiquitous in the research community [1–3] and real-world applications [4–6]. Meanwhile, due to the existence of the curse of dimensionality [7], the search space and the complexity of problems increase explosively, which degenerates the performance of conventional evolutionary algorithms (EAs) rapidly. Inspired by the divide-and-conquer strategy, the cooperative coevolution (CC) [8] framework decomposes LSOPs into multiple sub-components and optimizes them alternately, which is an efficient and flexible approach to dealing with LSOPs.

Two critical components of the implementation of the CC framework are the decomposition strategy and the optimization technique. Up to now, hundreds of techniques have been proposed to decompose the LSOPs, and we will briefly survey the development of these grouping methods in "A brief survey of grouping methods". One of the most popular mechanisms of the decomposition is the differential grouping (DG) [9] and its extensions [10–12], which identify the additive interactions by fitness difference of perturbed samples and are regarded as the most precise among the decomposition mechanisms. However, the original DG needs a high computational cost for realizing the decomposition. In the extreme case, DG will consume 1,001,000 fitness evaluations (FEs) for a 1000-D fully separable function, which is impossible to extend DG for large-scale expensive optimization problems (LSEOPs) in real-world applications such as

✉ Rui Zhong
rui.zhong.u5@elms.hokudai.ac.jp

[1] Graduate School of Information Science and Technology, Hokkaido University, Kita Ward, Kita 14 Jonishi, Sapporo, Hokkaido 060-0814, Japan

[2] Information Initiative Center, Hokkaido University, Kita Ward, Kita 11 Jonishi, Sapporo, Hokkaido 060-0811, Japan

aerodynamic design optimization [13], flow-shop scheduling problems [14], and so on. However, the recursive DG (RDG) [15] makes the extension of DG to LSEOPs possible. RDG examines the interactions between subset-to-subset rather than variable-to-variable with the binary search fashion and greatly reduces the averaging FEs to $1.47e4$ on CEC2010 and CEC2013 benchmark functions. Moreover, efficient RDG (ERDG) [16] and merged DG (MDG) [17] further reduce the computational budget and improve the decomposition accuracy, which provides a great potential to tackle the LSEOPs based on the CC framework with DG-based decomposition techniques.

Another component of the CC framework is optimization. Conventional EAs are inclined to be determined as a stochastic search [18], they usually consume lots of FEs to find an acceptable solution, which severely limits the scalability of EAs to deal with expensive optimization problems (EOPs) [19]. To overcome this challenge, a mature approach is adopting the relatively low-computational surrogate model to assist in the optimization of EAs. This cluster of algorithms is called surrogate-assisted EAs (SAEAs) [20]. A variety of techniques can be employed to construct surrogate models such as Radial Basis Functions (RBF) [21], Polynomial Regression model (PR) [22], Gaussian Process Regression (GPR) [23] or Kriging [24], and Artificial Neural Networks (ANN) [25]. However, the performance of SAEAs has a strong relationship with the quality of the surrogate model, and the presence of the curse of dimensionality in LSEOPs will decline the accuracy and quality of surrogate models rapidly. Although some works [26–28] have adopted dimension reduction techniques and efficient surrogate model management schemes to tackle the high-dimensional EOPs, most of them focus on the EOPs with less than 200 decision variables and rarely involve 1000-D scales. To overcome the severe challenge of 1000-D LSEOPs, we suggest combining the CC framework and the SAEAs to deal with LSEOPs, which is also the motivation of our research.

Meanwhile, scholars have achieved some contributions to dealing with LSEOPs: Ivanoe et al. [29] first introduced the SAEA into the CC framework and proposed a surrogate-assisted CC (SACC) optimizer. Ren et al. [30] follow the SACC framework and combine it with RBF-embedded success-history-based adaptive differential evolution (RBF-SHADE-SACC) to further promote the development of SACC. Sun et al. [31] proposed a surrogate ensemble assisted large-scale expensive evolutionary algorithm with random grouping (SEA-LEEA-RG) to solve LSEOPs. The random grouping strategy is adopted to decompose the original problem, while local and global RBF models are constructed with partial and all samples, and the best-predicted sample participates in the subsequent optimization process. Similarly, Sun et al. [32] combined the random grouping strategy with RBF-assisted DE to deal with LSEOPs and proposed a large-scale expensive optimization with a switching strategy (LSEO-SS), where the switching strategy controls the usage of the local and the global surrogate model, and the unique escape mechanism by re-initialization is designed to avoid premature. Besides, Gu et al. [33] proposed a surrogate-assisted differential evolution algorithm with the adaptive multi-subspace search (SADE-AMSS) for LSEOPs. In the decomposition, SADE-AMSS uses the principal component analysis (PCA) and random decision variable selection to construct the multi-subspace, and three efficient search strategies are embedded in the SADE adaptively. However, most studies focus on optimizers and only adopt the random strategy in the sub-component division. Although capturing the interactions between decision variables based on probability is sometimes efficient and accurate decomposition techniques will consume some additional FEs for interaction identification, the wrong-determined interaction may lead the direction of optimization in the wrong way [34]. Therefore, decomposition accuracy and optimization performance are two mutually restrictive metrics under the background of LSEOPs, and it is necessary to integrate the high-accuracy and computationally cheap decomposition method with efficient optimizers to deal with LSEOPs.

This paper proposes a novel decomposition method named efficient dual differential grouping (EDDG) and adopts a surrogate ensemble assisted differential evolution (SEADE) as the basic optimizer. In the decomposition stage, EDDG contains the both merits of ERDG and Dual DG (DDG) that can detect additive and multiplicative separability with high efficiency and accuracy. Inspired by RDG2 [12] and RDG3 [35], we also introduce an adaptive threshold for multiplicative interaction identification and limit the maximum scale of sub-components to alleviate the negative effect of the curse of dimensionality in optimization at the cost of a certain level of decomposition accuracy. In the optimization stage, our adopted SEADE utilizes the global and local surrogate model to estimate promising solutions, and the participation of these elites is expected to accelerate the convergence of optimization with limited computational resources. More specifically, the contributions of this paper are summarized as follows:

1. In the decomposition stage, we embed the subset-to-subset version of DDG into ERDG, which can detect both the additive and multiplicative interactions efficiently and simultaneously. Inspired by the RDG2, our proposed EDDG inherits the adaptive threshold design, and inspired by RDG3, we limit the maximum scale of sub-components and further decompose the relatively large-scale sub-components to alleviate the curse of dimensionality on solving LSEOPs. Besides, the paper [36] only proved the establishment of the variable-to-variable version of DDG, and we provide mathematical support to the establishment of the subset-to-subset version of DDG.

2. We introduce the SEADE as the basic optimizer in the optimization stage. In each generation of optimization, SEADE constructs the generalized regression neural network (GRNN) model with all samples in the archive and the GPR model with samples in the current generation, which corresponds to the global and local models. Random search as the infill sampling criterion is applied to estimate the promising solutions, and the participation of elites in the optimization is expected to accelerate the convergence of optimization.

3. We implement a set of experiments to evaluate the performance of our proposal on the CEC2013 suite with fewer FEs than the standard experiment setting. To show the superiority of EDDG, we compare the decomposition accuracy (DA), consumed FEs, and optimization performance with state-of-the-art decomposition methods including RDG,[1] RDG2,[2] ERDG,[3] and MDG.[4] To show the efficiency of SEADE, we compare the optimization results between SEADE and conventional DE. Experimental and statistical results show that our proposal has great prospects to tackle LSEOPs. To the best of our knowledge, not much work has been reported on adopting the CC framework and SAEAs to deal with LSEOPs.

The remainder of this paper is organized as follows: the following section covers the related works. Section "Our proposal: SEADECC-EDDG" provides a detailed introduction to our proposal: SEADECC-EDDG. The subsequent section describes the numerical experiments on the CEC2013 suite. Section "Discussion" discusses the experimental results and provides some future topics. Finally, Section "Conclusion" concludes the paper.

Related works

Preliminaries

Cooperative coevolution (CC)

A general CC framework contains three stages: decomposition, optimization, and combination.

Decomposition: the original problem is divided into multiple sub-problems with a certain decomposition method.

Optimization: optimization techniques (e.g., DE, PSO) are applied to optimize the sub-components. Usually, a sub-solution is not a complete solution and the fitness value cannot be calculated, the context vector [37] will participate in the evaluation of sub-solutions.

1 https://bitbucket.org/yuans/rdg.

2 https://bitbucket.org/yuans/rdg2.

3 https://github.com/ymzhongzhong/ERDG.

4 https://github.com/hzt2015/mdg.

Combination: sub-solutions of sub-components are combined, which is regarded as the optimum of the original problem.

In summary, the pseudocode of CC is shown in Algorithm 1.

Algorithm 1 CC framework

```
Require: Problem: $P$, Decomposition method: $D$
Ensure: Optimum: $X$
 1: Decompose the problem $P$ with $D$ into sub-components $[sc_1, sc_2, ..., sc_k]$
 2: Initialize context vector $CV$
 3: Initialize sub-solutions $SS$
 4: while not Terminate do
 5:     for $i$=1 to $k$ do
 6:         Optimize sub-component $sc_i$ with $SS$
 7:         Update $CV$
 8:     end for
 9: end while
10: Construct the optimum $X$
11: return $X$
```

Differential evolution (DE)

DE is first proposed by Storn and Price[38]. Due to its efficiency, robustness, applicability, and other superior characteristics, DE has been widely applied in many fields such as structural damage detection [39–41], neural architecture search [42, 43], and drug design[44, 45]. The conventional DE consists of four steps: Initialization, mutation, crossover, and selection.

Initialization: as most EAs, DE is a population-based optimization approach, and random initialization is the most common method to initialize the population first. Supposing the population size is n and the dimension of the problem is m, Eq. (1) shows the structure of the population X and the generation of each individual x_{ij} of DE.

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ x_{31} & x_{32} & \cdots & x_{3m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix} \tag{1}$$
$$x_{ij} = r_1 \cdot \left(UB_j - LB_j\right) + LB_j,$$

where LB_j and UB_j are the lower and the upper bound of the jth dimension and r_1 is a random number in (0, 1). x_{ij} represents the value of the jth dimension of the ith individual.

Mutation: the most characteristic feature of DE is generating the offspring population by the differential mutation operator, which is also the origin of the name of DE. Several variants of mutation strategies have been reported in many pieces of literature, and the commonly used schemes are listed as follows:

DE/rand/1 :

$$v_i = X_{r_1} + F \cdot \left(X_{r_2} - X_{r_3}\right). \tag{2}$$

DE/best/1 :

$$v_i = X_{best} + F \cdot \left(X_{r_1} - X_{r_2}\right). \tag{3}$$

DE/cur-to-rand/1 :

$$v_i = X_i + F \cdot (X_{r_1} - X_i) + F \cdot \left(X_{r_2} - X_{r_3}\right). \tag{4}$$

DE/cur-to-best/1 :

$$v_i = X_i + F \cdot (X_{best} - X_i) + F \cdot \left(X_{r_1} - X_{r_2}\right), \tag{5}$$

r_1, r_2, r_3 are three mutually different integers uniformly generated from $[0, n]$, X_{best} is the current best individual in the population.

Crossover: afterward, the crossover operator generates the trail vector u by Eq.(6)

$$u_i^j = \begin{cases} v_i^j, & \text{If } r \le CR \text{ } or \text{ } j = j_{rand} \\ X_i^j, & \text{Otherwise} \end{cases} \tag{6}$$

where r is a random number in (0, 1), j_{rand} is a random integer in $[1, m]$, and $CR \in (0, 1)$ is the crossover rate.

Selection: finally, the selection operator chooses the proper individual to survive. Taking the minimization problem as an example, Eq. (7) shows the selection mechanism of the conventional DE.

$$X_{i+1} = \begin{cases} v_i, & \text{If } f(v_i) < f(X_i) \\ X_i, & \text{Otherwise.} \end{cases} \tag{7}$$

RDG and ERDG

Original DG-based decomposition methods are high-accuracy but computationally expensive. To overcome this issue, RDG [15] first introduces a binary search fashion and extends the DG mechanism to subset-to-subset. Assuming that $X_1 \subset X$ and $X_2 \subset X$ are two mutually exclusive subsets of decision variables, which means $X_1 \cap X_2 = \emptyset$. If there are two unit vectors $\mathbf{u}_1 \in U_{X_1}, \mathbf{u}_2 \in U_{X_2}$, two arbitrary numbers $l_1, l_2 > 0$, and a candidate solution $\mathbf{x}^*$ in the search space, such that

$$\begin{aligned} &f\left(\mathbf{x}^* + l_1\mathbf{u}_1 + l_2\mathbf{u}_2\right) - f\left(\mathbf{x}^* + l_2\mathbf{u}_2\right) \\ &\quad \ne f\left(\mathbf{x}^* + l_1\mathbf{u}_1\right) - f\left(\mathbf{x}^*\right), \end{aligned} \tag{8}$$

then, there exist some interactions between decision variables in X_1 and X_2. Based on this corollary, RDG initially assigns the first variable x_1 to subset X_1 and the rest of the variables to subset X_2. If X_1 does not interact with X_2 detected by Eq. (8), the element in X_1 is determined as a separable variable,

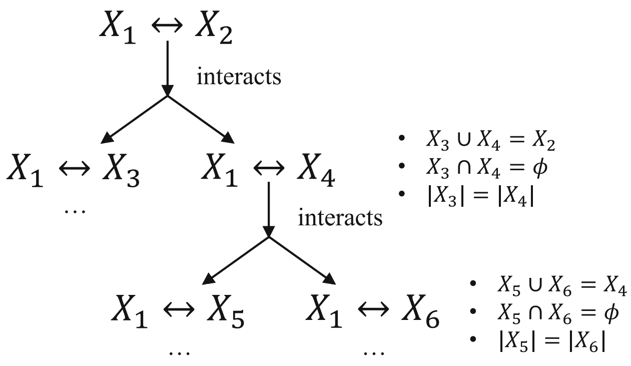

Fig. 1 The decomposition process of RDG

otherwise, X_2 is separated into two mutually exclusive subsets with equal size and recursively execute the identification. Figure 1 shows a demonstration of RDG.

This binary search fashion strategy extremely reduces the time complexity for decomposition from $O(N^2)$ to $O(N \log(N))$. Furthermore, Efficient RDG (ERDG) [16] notices the redundant interaction examinations exist in the RDG, which can be avoided by historical information. From Fig. 1, variable subset X_3 and X_4 are mutually exclusive and $X_3 \cup X_4 = X_2$, thus, Eq. (8) can be transformed to Eq. (9)

$$\begin{aligned} &f\left(\mathbf{x}^* + l_1\mathbf{u}_1 + l_2\left(\mathbf{u}_3 + \mathbf{u}_4\right)\right) - f\left(\mathbf{x}^* + l_2\left(\mathbf{u}_3 + \mathbf{u}_4\right)\right) \\ &\quad \ne f\left(\mathbf{x}^* + l_1\mathbf{u}_1\right) - f\left(\mathbf{x}^*\right) \end{aligned} \tag{9}$$

Here, we define

$$\begin{aligned} \Delta_1 &= f\left(\mathbf{x}^* + l_1\mathbf{u}_1 + l_2\left(\mathbf{u}_3 + \mathbf{u}_4\right)\right) \\ &\quad - f\left(\mathbf{x}^* + l_2\left(\mathbf{u}_3 + \mathbf{u}_4\right)\right) \\ \Delta_2 &= f\left(\mathbf{x}^* + l_1\mathbf{u}_1\right) - f\left(\mathbf{x}^*\right) \end{aligned} \tag{10}$$

If X_1 interacts with X_2, X_2 will be separated into X_3 and X_4, and the interaction identification of $X_1 \leftrightarrow X_3$ can be computed as

$$\begin{aligned} \Delta_1^{'} &= f\left(\mathbf{x}^* + l_1\mathbf{u}_1 + l_2\mathbf{u}_3\right) - f\left(\mathbf{x}^* + l_2\mathbf{u}_3\right) \\ \Delta_2^{'} &= f\left(\mathbf{x}^* + l_1\mathbf{u}_1\right) - f\left(\mathbf{x}^*\right) = \Delta_2. \end{aligned} \tag{11}$$

Similarly, Eq. (11) can be applied to identify the interaction of $X_1 \leftrightarrow X_4$.

$$\begin{aligned} \Delta_1^{''} &= f\left(\mathbf{x}^* + l_1\mathbf{u}_1 + l_2\mathbf{u}_4\right) - f\left(\mathbf{x}^* + l_2\mathbf{u}_4\right) \\ \Delta_2^{''} &= \Delta_2. \end{aligned} \tag{12}$$

And we can reasonably infer that if $(\Delta_1 - \Delta_2) = (\Delta_1^{'} - \Delta_2^{'})$, X_1 does not interact with X_4; otherwise, X_1 interacts with X_4, and the calculation of $\Delta_1^{''}$ in Eq. (12) can be avoided to save the FEs, more details can refer to [16]. Although the time

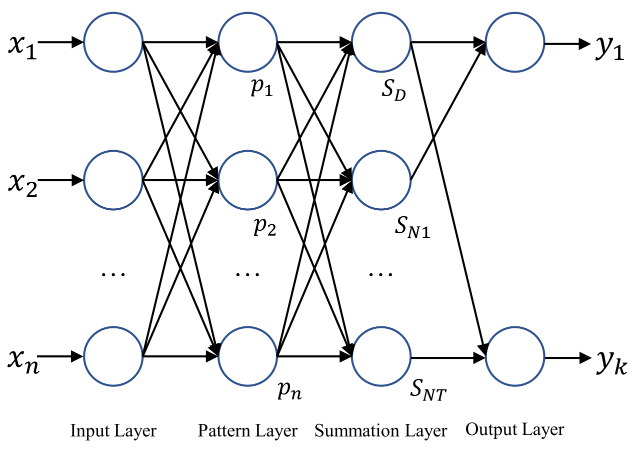

Fig. 2 The structure of GRNN [47]

complexity of ERDG is identical to RDG and its variants theoretically, the necessary FEs in practice can be significantly reduced to averaging $7.62e3$ on CEC2013 benchmark functions.

Generalized regression neural network (GRNN)

GRNN is a highly parallel radial basis function network based on the one-pass algorithm [46, 47], and the universal approximation theorem [48] endows an ability to GRNN for approximating any functions theoretically. Specifically, the topology of GRNN is shown in Fig. 2

GRNN consists of four layers: the input layer, pattern layer, summation layer, and output layer. Equation (13) summarizes the GRNN logic in an equivalent nonlinear regression formula [49]:

$$E[Y/X] = \left(\int_{-\infty}^{+\infty} Yf(X,Y)\mathrm{d}Y\right) / \left(\int_{-\infty}^{+\infty} f(X,Y)\mathrm{d}Y\right), \tag{13}$$

where $X = \{X_1, X_2, ..., X_n\}$ denotes the n-D input vector, Y is the predicted label by GRNN, $E[Y/X]$ means the expectation of Y given an input X, and $f(X,Y)$ is the joint probability density of X and Y [50].

Gaussian process regression (GPR)

GPR as a popular probabilistic surrogate model has been widely applied in SAEAs [51–53]. Deriving from the Bayesian theory, the GPR can be seen as a random process to undertake the non-parametric regression with the Gaussian processes [54]. For any inputs, the corresponding probability distribution over function $f(x)$ follows the Gaussian distribution as:

$$f(x) \sim \mathcal{GPR}\left(m(x), k\left(x, x'\right)\right), \tag{14}$$

where $m(x)$ and $k(x,x')$ are the mean and the covariance functions respectively, which can be expressed by:

$$\begin{aligned} m(x) &= E(f(x)) \\ k(x,x') &= E\left[\left(m(x) - f\left(x'\right)\right)\left(m(x) - f\left(x'\right)\right)\right], \end{aligned} \tag{15}$$

$E(\cdot)$ represents the expectation. $m(x)$ is generally zero to simplify the computation practically. $k(x,x')$ is also named the kernel function to explain the relevance degree between a target observation of the training data set and the prediction based on the similarity of the respective inputs.

In the regression problem, the prior distribution of output y can be denoted as

$$y \sim N\left(0, k\left(x, x'\right) + \sigma_n^2 I_n\right) \tag{16}$$

where $N(\cdot)$ represents the normal distribution. σ_n^2 is the noise term. Assuming the training dataset x and the testing set x' follow the identical Gaussian distribution, and the prediction y' would follow a joint prior distribution with the training output y as [55]

$$\begin{bmatrix} y \\ y' \end{bmatrix} \sim N\left(0, \begin{bmatrix} k(x,x) + \sigma_n^2 I_n & k\left(x,x'\right) \\ k\left(x,x'\right)^T & k\left(x',x'\right) \end{bmatrix}\right) \tag{17}$$

where $k(x,x)$, $k(x,x')$ and $k(x',x')$ represent the covariance matrices among inputs from the training set, the training and testing sets, as well as the testing set.

To guarantee the performance of the GPR, some hyper-parameters θ in the covariance function require to be optimized with n samples in the training process. One efficient optimization solution is to minimize the negative log marginal likelihood $L(\theta)$ as [56]

$$\begin{aligned} L(\theta) &= \frac{1}{2}\log\left[\det\lambda(\theta) + \frac{1}{2}y^T\lambda^{-1}(\theta)y + \frac{n}{2}\log(2\pi)\right] \\ \lambda(\theta) &= k(\theta) + \sigma_n^2 I_n. \end{aligned} \tag{18}$$

After the hyper-parameters optimization of the GPR, the prediction y' can be obtained at data set x' through calculating the corresponding conditional distribution $p(y'|x',x,y)$ as

$$\begin{aligned} & p\left(y'|x',x,y\right) \sim N\left(y'|\bar{y}', \operatorname{cov}\left(y'\right)\right) \\ & \bar{y}' = k\left(x,x'\right)^T\left[k(x,x) + \sigma_n^2 I_n\right]^{-1} y \\ & \operatorname{cov}\left(y'\right) = k\left(x',x'\right) \\ & \quad - k\left(x,x'\right)^T\left[k(x,x) + \sigma_n^2 I_n\right]^{-1} k\left(x,x'\right), \end{aligned} \tag{19}$$

where $\bar{y}'$ stands for the corresponding mean values of prediction. $\operatorname{cov}(y')$ denotes a variance matrix to reflect the uncertainty range of these predictions. More details of mathematical support can be found in [57].

The classification of SAEAs

In the past decades, many SAEAs have been published to deal with EOPs, which can be briefly divided into three categories:

Global surrogate model assisted EAs This kind of method applies the global surrogate model to approximate the whole fitness landscape of optimization problems. Liu et al. [23] focused on median-scale EOPs and proposed a surrogate model-aware mechanism to search in a small promising area carefully. Given the presence of the curve of dimensionality, the Sammon mapping is introduced to reduce the dimension of the problem. Yu et al. [58] introduced a generation-based optimal restart strategy to assist social learning PSO for solving medium-scale EOPs, where the global RBF model is re-constructed every few generations with the best samples archived in the historical database. Nuovo et al. [59] adopted an inexpensive fuzzy function to approximate the global fitness landscape, and the samples obtained from the evolution are applied to construct and refine the fuzzy approximation function. When the model becomes reliable, it is employed to evaluate solutions, and elites evaluated by the fuzzy approximation function will be evaluated by the real objective function. Nikolaus et al. [60] embedded a global quadratic model and a simple portfolio algorithm to covariance matrix adaptation evolution strategy (CMA-ES). In real-world applications, the global surrogate assisted technique also makes contributions: Pan et al. [61] hybridized the gannet optimization algorithm (GOA) and the differential evolution (DE) algorithm with the RBF model to deal with wireless sensor network (WSN) coverage problems and achieved great progress, Han et al. [62] introduced the RBF model to DE optimizer and proposed a surrogate assisted evolutionary algorithm with restart strategy (SAEA-RS) to tackle the constrained space component thermal layout optimization problem. Stander et al.[63] focused on the popular pressure swing adsorption system optimization problem and proposed a surrogate-assisted NSGA (SA-NSGA) for multi-objective optimization problems. Meanwhile, scholars noticed that it is not easy to construct a high-fidelity model for the whole fitness landscape. Thus, building a local surrogate model to execute more efficient exploitation has become a popular topic.

Local surrogate model assisted EAs The local surrogate model aims to improve the approximation accuracy of the surrogate model and search the promising solutions in subregions. Martinize et al. [64] proposed a surrogate-assisted local search to enhance the exploitation and embedded it to MOEA/D to deal with multi-objective EOPs. Lim et al. [65] proposed a generalized surrogate-assisted evolutionary framework by working on two major issues: (1) to mitigate the curse of uncertainty robustly, and (2) to benefit from the blessing of uncertainty. A diverse set of approximation techniques is employed to assist the memetic algorithm and improve the effectiveness of local search. Sun et al. [66] designed a fitness estimation strategy and embedded it into PSO to estimate the fitness value of particles based on their parents, ancestors, and siblings.

Ensemble surrogate model assisted EAs The ensemble surrogate model usually consists of a global and a local surrogate model, which has been proven that it can outperform a single model on most problems in practice [67]. Wang et al. [68] applied the GRNN and the RBF to assist DE. In the global surrogate assisted phase, DE is employed as the search engine to produce multiple trial vectors, and GRNN takes the responsibility to evaluate these trial vectors. In the local surrogate-assisted phase, the interior point method cooperated with the RBF is utilized to refine each individual in the population. Cai et al. [27] utilized the optima obtained from the global and local surrogate model constructed by the entire search space and the neighbor region around the personal historical best particle respectively to guide the direction of optimization of the PSO. In Wang et al.'s work [69], RBF as the global surrogate model is applied to pre-screen the offspring generated by the DE, and the local surrogate model trained with τ best samples is applied to find promising solutions in sub-regions. Furthermore, the surrogate ensemble assisted technique is also a popular approach in real-world expensive optimization problems: Yu et al. [70] proposed a dynamic surrogate-assisted evolutionary algorithm framework for expensive structural optimization, where the adaptive surrogate model selection technology can automatically select the most accurate model from the meta-model pool by the minimum root of mean square error. Chen et al. [71] proposed a surrogate assisted evolutionary algorithm with the dimensionality reduction method for water flooding production optimization, where the Sammon mapping is adopted as the dimension reduction method, and the Kriging model with lower confidence bound (LCB) is employed to estimate promising solutions. Zhou et al. [72] developed a hierarchical surrogate-assisted evolutionary algorithm with multi-fidelity modeling technology for multi-objective shale gas reservoir problems, both the net present value and cumulative gas production are regarded as objective functions, where the low-fidelity model can adopt exploration behaviors, and the high-fidelity model can generate high-quality solutions as a local search operator. Tang et al. [73] proposed an adaptive dynamic surrogate-assisted evolutionary algorithm for the aerodynamic shape design optimization of transonic airfoil and wing. The construction of the high-fidelity global surrogate model is abandoned due to high computational cost, and the local surrogate model on the sub-region is encouraged to implement exploitation behaviors. A large number of surrogate assisted techniques have been introduced to research areas and applications, which have achieved great success.

A brief survey of grouping methods

In this section, we roughly survey the development of grouping methods and classify the mechanism of grouping methods into three categories: static decomposition, random decomposition, and learning-based decomposition.

Static decomposition CCGA [8], as the pioneer of the static decomposition method, divides N-D problems into $N*$ 1-D sub-problems and optimizes them with the genetic algorithm (GA), which shows the competitiveness on separable functions but may mislead the direction of optimization on non-separable functions. To address this issue, CCPSO-S$_K$ [37] decomposes N-D problems into $k*s$-D sub-problems in order, where $N = k*s$. Particle Swarm Optimization (PSO) is adopted as the basic optimizer for each sub-component. However, a remaining problem for static grouping is that if two interacting decision variables are separated initially, then there is no opportunity for them to be assigned to a group anymore.

Random decomposition Random-based grouping methods are developed to solve the mentioned problem of static decomposition, and random grouping can be further divided into two categories: fixed sub-component size and dynamic sub-component size.

(1). *Fixed sub-component size* DECC-I [74] is the first proposed random grouping, which divides the N-D problems into $k * s$-D sub-components with random order and optimizes them with Differential Evolution (DE). Later, DECC-G [75] introduces an adaptive weighting strategy and mathematically proves that the probability of assigning two interacting variables to the same sub-component is pretty high. CCPSO-S$_K$-rg-aw [76] extends the decomposition method in DECC-G with PSO optimizer.

(2). *Dynamic sub-component size* An obvious flaw of static decomposition and random grouping with the fixed sub-component size is that we manually limit the scale of sub-components, but this specific parameter is expected to be different among various problems. For example, fully separable functions prefer small-scale sub-components, whereas a relatively large-scale sub-component size can ensure the probability to capture the interactions in non-separable functions. Therefore, DECC-II [74] was proposed to randomly select the scale of sub-components s from a predefined range in the coevolution cycle. MLCC [77] provides a scale candidate $[s_1, s_2, ..., s_n]$ and chooses the scale s_i by the recent improvement of the context vector.

Perturbation-based approaches In the early development of decomposition methods, researchers attempt to capture the interactions by probability until the appearance of Linkage Identification (LI) [78, 79]. Based on the different mechanisms of interaction identification, LI includes two kinds: Non-linearity Check and Monotonicity Detection. In this paper, we mainly focus on the Non-linearity Check, and the detailed introduction to Monotonicity Detection can be found in [78].

Non-linearity Check based methods are well-studied techniques. As a quantitative method, LINC constructs the approximate partial derivative to detect the interactions between decision variables. Mathematically,

$$\begin{aligned}&\text{If } \frac{\partial f^2(x)}{\partial x_i \partial x_j} = 0\\&\text{Then } x_i \ and \ x_j \ are \ separable.\end{aligned} \tag{20}$$

However, most of the studies focus on black-box optimization, and the value of the partial derivative cannot be obtained from the fitness landscape directly. Therefore, LINC perturbs in corresponding dimensions of samples to identify the separability:

$$\begin{aligned}&\forall s \in Pop:\\&\Delta_i = f(s_i) - f(s), \Delta_j = f\left(s_{ij}\right) - f\left(s_j\right)\\&\text{If } |\Delta_i - \Delta_j| < \varepsilon\\&\text{Then } x_i \ and \ x_j \ are \ separable\end{aligned} \tag{21}$$

Where ε is an allowable error. Once detection for two decision variables needs four FEs. Supposing the population size is k, the dimension of the problem is N, and the total FEs is $4k\frac{N(N-1)}{2}$. When N approaches 1000 and k equals 1, the necessary FEs is 1,998,000, which is completely unacceptable for LSOPs. Differential Grouping (DG) [9] extends the identical mechanism of LINC to LSOPs. Considering the high computational cost, DG only calculates the perturbation from the lower bound of search space to the upper bound. The neglect of indirect interaction reduces the FEs to $\frac{N}{2m}(m+N-2)$, where m is the size of the sub-components. In the extreme case that when the 1000-D problem is a fully separable function, the consumed FEs is 1,001,000. A sensitivity test for ε is also provided in [9]. Global DG (GDG) [11] introduces the pairwise interactions to improve the decomposition accuracy. DG2 [80], a faster and more accurate decomposition method, reduces the FEs to $\frac{N(N+1)}{2} + 1$ by sharing the FEs information. Furthermore, given the conventional DG is sensitive to ε and the absolute difference between $|\Delta_i - \Delta_j|$ in Eq. (21) may come from two aspects: interaction and rounding error from a real number to a floating-point number. Thus, DG2 proposes an adaptive ε based on IEEE 754 Standard. However, DG2 still costs 500,501 FEs for 1000-D problem decomposition, which limits the scalability of DG2 to higher-dimensional problems.

However, the high-computational budget of DG, DG2, and its variants still limits their scalability in LSOPs with more decision variables such as 5000-D, but the promotion of RDG makes this extension possible. RDG [15] detects the interactions between subset-to-subset with binary search fashion, which extremely decreases the computational complexity from $O(N^2)$ to $O(N\log(N))$. Moreover, ERDG [16] and merged DG (MDG) [17] avoid the redundant computation of RDG in interaction identification and further save the computational budget practically. To the best of our knowledge, MDG is the lightest DG-based decomposition method so far.

The above DG-based methods concentrate on additive interaction identification. However, other kinds of separability also exist. Dual DG (DDG) [36] applies the logarithmic operation to transform the multiplicative separability into additive separability and detect the interactions with DG. More specifically, in a multiplicative separable function $g(\mathbf{x})$ with a minimum larger than 0, $g_i(\mathbf{x}_i)$ are multiplicative separable sub-components. We define $f(\mathbf{x}) = \ln g(\mathbf{x})$, and $f(\mathbf{x})$ can be rewritten to

$$\begin{aligned} f(\mathbf{x}) &= \ln g(\mathbf{x}) \\ &= \ln \prod_{i=1}^{k} g_i(\mathbf{x}_i) \\ &= \sum_{i=1}^{k} \ln g_i(\mathbf{x}_i),\ 1 < k \le D \end{aligned} \tag{22}$$

Therefore, the identification condition of DDG to detect the multiplicative separability can be formulated by Eq. (23)

$$\begin{aligned} &\text{If } f(s), f(s_i), f(s_j), f(s_{ij}) \text{ arepositiveand} \\ &|\ln(f(s_{ij}) - \ln f(s_j)) - \ln(f(s_i) - \ln f(s))| < \varepsilon \\ &\text{Then } x_i \ and\ x_j \text{aremultiplicativelyseparable.} \end{aligned} \tag{23}$$

Besides, GSG[81], a general separability grouping method, conducts a comprehensive theoretical investigation that extends the study from the existing additive separability to general separability. The core of the theoretical research is the minimum points shift principle, which can effectively identify general separability.

Our proposal: SEADECC-EDDG

This section introduces our proposal: SEADECC-EDDG in detail. Two parts in our proposal need to be explained: EDDG for decomposition and SEADE for optimization of sub-components.

Decomposition: EDDG

As we mentioned before, the motivation for our proposed EDDG is that we want to develop an advanced decomposition technique that contains the efficiency of ERDG and the advantage of DDG which can detect additive and multiplicative interactions simultaneously. Therefore, we adopt the decomposition framework of ERDG and embed the determination condition of DDG to ERDG, and propose EDDG. Moreover, we design an adaptive threshold for multiplicative interaction identification inspired by RDG2. Considering this research focuses on solving LSEOPs, and the relatively large-scale sub-components still consume a large number of FEs to search for acceptable solutions due to the curse of dimensionality, thus, inspired by RDG3, we decompose the sub-component whose scale is large than the pre-defined maximum scale. In summary, the pseudocode of EDDG is shown in Algorithm 2.

Algorithm 2 EDDG

```
Require: Objective function: $f(\cdot)$; Lower and Upper bound of search
    space: lb, ub; Maximum scale of sub-components: $MS$
Ensure: Separable groups: $sep$; Nonseparable groups: $nonsep$
1: $sep, nonsep \leftarrow \emptyset$
2: $\mathbf{x}_{l,l} \leftarrow \mathbf{lb}$; $y_{l,l} \leftarrow f(\mathbf{x}_{l,l})$
3: $X_1 \leftarrow \{x_1\}$; $X_2 \leftarrow \{x_2, x_3, ..., x_N\}$
4: while $X_2$ is not empty do
5:     $\mathbf{x}_{u,l} \leftarrow \mathbf{x}_{l,l}$; $\mathbf{x}_{u,l}(X_1) \leftarrow \mathbf{ub}(X_1)$
6:     $y_{u,l} \leftarrow f(\mathbf{x}_{u,l})$
7:     $FA \leftarrow \{y_{l,l}, y_{u,l}, NaN, NaN\}$
8:     $(X_1^*, \hat{\beta}_{add}, \hat{\beta}_{multi}) \leftarrow$ Interact$(X_1, X_2, \mathbf{x}_{l,l}, \mathbf{x}_{u,l}, \mathbf{lb}, \mathbf{ub}, F)$
9:     if len$(X_1^*)$ = len$(X_1)$ then
10:        if len$(X_1^*) > 1$ then
11:            $nonsep \leftarrow \{nonsep, X_1^*\}$
12:        else
13:            $sep \leftarrow \{sep, X_1^*\}$
14:        end if
15:        $X_1 \leftarrow$ pop$(X_2)$
16:    else
17:        $X_1 \leftarrow X_1^*$; $X_2 \leftarrow X_2 - X_1$
18:    end if
19:    if $X_2 = \emptyset$ then
20:        if len$(X_1) >$ MS then
21:            Separate $X_1$ to $X_{11}, X_{12}$,...
22:            $nonsep \leftarrow \{nonsep, X_{11}, X_{11}, ...\}$
23:        else if len$(X_1) > 1$ then
24:            $nonsep \leftarrow \{nonsep, X_1\}$
25:        else
26:            $sep \leftarrow \{sep, X_1\}$
27:        end if
28:    end if
29: end while
30: return $sep, nonsep$
```

NaN is a non-numeric value. In Algorithm 2, EDDG first detects the interactions between x_1 and the rest decision variables (i.e., X_1 and X_2) (from line 5 to line 8). If these two subsets are separable (see line 9), EDDG further identifies the length of the subset X_1. And if more than one decision variable in X_1, EDDG assigns the X_1 as a non-separable sub-component (see line 11), otherwise, it is allocated to a separable decision variable (see line 13). If there exist interactions between X_1 and X_2, EDDG separates the interrelated decision variables in X_2 and assigns them into X_1 (see line 17). EDDG repeats the above process until all decision variables are well placed. Besides, the main difference between EDDG and ERDG in Algorithm 2 is in line 20 that if the scale of X_1 is larger than the pre-defined maximum scale of sub-component MS, X_1 is further divided into multiple sub-components randomly. Especially for fully non-separable and overlapping functions, the accurate decomposition to these two kinds of problems is that all decision variables are divided into a sub-component, but the optimization by this decomposition cannot be accelerated based on the CC framework. Thus, EDDG sacrifices the accuracy of decomposition and randomly decomposes the sub-component which has a larger scale than MS, and the accuracy of this decomposition will reduce to

$$Acc = \frac{MS}{D} \times 100\%, \tag{24}$$

where D is the dimension size of the original problem. Assuming the $MS = 100$ and $D = 1000$, the decomposition to the fully non-separable and overlapping function will extremely decrease to 10%. However, this process can alleviate the curse of dimensionality and accelerate the convergence of optimization at the cost of decomposition accuracy.

Algorithm 2 shows the framework of EDDG which mainly inherits the skeleton of ERDG, and Algorithm 3 illustrates the interaction identification between X_1 and X_2.

β_{multi}^{MAX} is larger than ε_{multi} greatly to ensure that when a negative value exists in FA, the $\ln(\cdot)$ operator is invalid, and in this case X_1 and X_2 are multiplicatively non-separable. In Algorithm 3, EDDG first calculates the additive difference (see line 8) and multiplicative difference (see line 13), and If X_1 and X_2 are additively and multiplicatively non-separable, X_2 is divided into two equal-sized and mutually exclusive subsets $X_2^{'}$ and $X_2^{''}$ (see line 21), and the interaction identification between X_1 and these two subsets is further executed, and this process is continued until EDDG finds all variables that interact with X_1.

After X_2 is separated into $X_2^{'}$ and $X_2^{''}$, if X_1 does not interact with $X_2^{'}$, that we can reasonably infer that X_1 interacts with $X_2^{''}$ (see line 26), thus, the interaction identification between X_1 and $X_2^{''}$ can be saved. Similarly, if X_1 interacts with $X_2^{'}$ and $\beta_{add} = \hat{\beta}_{add}$ or $\beta_{multi} = \hat{\beta}_{multi}$, EDDG can infer that X_1 does not interact with $X_2^{''}$ (see line 33), which can save the FEs for decomposition.

Algorithm 3 Interact

```
Require: Variable subset: $X_1$, $X_2$; Solution vector: $\mathbf{x}_{l,l}$, $\mathbf{x}_{u,l}$; Lower
    and Upper bound: lb, ub; Fitness archive: $FA$
Ensure: Updated $X_1$, Additive fitness difference: $\beta_{add}$, Multiplicative
    fitness difference: $\beta_{multi}$,
1: $nonSep \leftarrow 1$
2: if $FA[2] = NaN$ then
3:     $\mathbf{x}_{m,l} \leftarrow \mathbf{x}_{l,l}$; $\mathbf{x}_{u,m} \leftarrow \mathbf{x}_{u,l}$
4:     $\mathbf{x}_{m,l}(X_2) \leftarrow (\mathbf{lb}(X_2) + \mathbf{ub}(X_2))/2$
5:     $\mathbf{x}_{u,m}(X_2) \leftarrow (\mathbf{lb}(X_2) + \mathbf{ub}(X_2))/2$
6:     $FA[2] \leftarrow f(\mathbf{x}_{m,l})$; $FA[3] \leftarrow f(\mathbf{x}_{u,m})$;
7:     $\Delta_1 \leftarrow FA[0] - FA[1]$; $\Delta_2 \leftarrow FA[2] - FA[3]$
8:     $\beta_{add} \leftarrow \mathbf{abs}(\Delta_1 - \Delta_2)$
9:     $\beta_{multi} \leftarrow \beta_{multi}^{MAX}$
10:    if elements in $FA$ are all positive then
11:        $\Delta_3 \leftarrow \ln(FA[0]) - \ln(FA[1])$
12:        $\Delta_4 \leftarrow \ln(FA[2]) - \ln(FA[3])$
13:        $\beta_{multi} \leftarrow \mathbf{abs}(\Delta_3 - \Delta_4)$
14:    end if
15:    if $\beta_{add} < \varepsilon_{add}$ or $\beta_{multi} < \varepsilon_{multi}$ then
16:        $nonSep \leftarrow 0$
17:    end if
18: end if
19: if $nonSep = 1$ then
20:    if len($X_2$) > 1 then
21:        Divide $X_2$ into two equal-sized and mutually exclusive sub-
    sets $X_2^{'}$ and $X_2^{''}$
22:        $tFA \leftarrow \{FA[0], FA[1], NaN, NaN\}$
23:        $(X_1^{'}, \hat{\beta}_{add}, \hat{\beta}_{multi}) \leftarrow$ Interact($X_1, X_2^{'}, \mathbf{x}_{l,l}, \mathbf{x}_{u,l}$, lb, ub, $tFA$)
24:        if $\beta_{add} \neq \hat{\beta}_{add}$ and $\beta_{multi} \neq \hat{\beta}_{multi}$ then
25:            if len($X_1^{'}$) = len($X_1$) then
26:                $(X_1^{''}, \beta_{add}^{'}, \beta_{multi}^{'}) \leftarrow$
    Interact($X_1, X_2^{''}, \mathbf{x}_{l,l}, \mathbf{x}_{u,l}$, lb, ub, $F$)
27:            else
28:                $tFA \leftarrow \{FA[0], FA[1], NaN, NaN\}$
29:                $(X_1^{''}, \beta_{add}^{'}, \beta_{multi}^{'}) \leftarrow$
    Interact($X_1, X_2^{''}, \mathbf{x}_{l,l}, \mathbf{x}_{u,l}$, lb, ub, $tFA$)
30:            end if
31:            $X_1 \leftarrow \{X_1^{'}, X_1^{''}\}$
32:        else
33:            $X_1 \leftarrow X_1^{'}$
34:        end if
35:    else
36:        $X_1 \leftarrow \{X_1, X_2\}$
37:    end if
38: end if
39: return $X_1$, $\beta_{add}$, $\beta_{multi}$
```

Similarly, our proposed EDDG inherits the interaction identification technique of ERDG and embeds the subset-to-subset version of DDG to EDDG (i.e. from Algorithm 3 from line 9 to line 17). As a flexible decomposition method, DDG will not consume the extra FEs and only add a slight computational burden, which means it can be embedded into any existing DG-based framework easily, and the time complexity of EDDG is $O(N\log(N))$ as same as ERDG. Here, we provide the computational complexity analysis of EDDG in detail.

Four categories of the problem exist: fully separable, fully non-separable, partially separable, and the nonseparable problem with overlaps. Supposing the dimension of the problem is D, for fully separable problems, the interaction is detected $t = D - 1$ times, and Algorithm 3 is invoked by Algorithm 2 $t = D - 1$ times, thus the consumed FEs of EDDG is $2(D-1) + D - 1 + 1 = 3D - 2$. For fully non-separable problems, the decomposition process forms a binary tree, and interaction is detected $t = 2(D-1) - 1 = 2D - 3$ times, while Algorithm 3 is invoked by Algorithm 2 $t = 1$ time, thus the necessary FEs for fully non-separable problems is $2(2D-3)+1+1 = 4D-4$. For partially separable and nonseparable problems with overlaps, the computational complexity of ERDG is $O(D\log(D))$, which has been well-proven in [16], and the embedded multiplicative interaction identification does not add any extra computational complexity, thus, in summary, the computational complexity of EDDG is also $O(D\log(D))$.

Moreover, although the original DDG is combined with RDG3 in[36], it did not prove this subset-to-subset version of DDG. Here, we provide mathematical support.

Theorem 1: *Let $f(\mathbf{x})$ to be a twice continuously differentiable function and $X = \{x_1, x_2, ..., x_N\}$ is a decision variable set of N-D problem. X_1 and X_2 are two mutually exclusive subsets where $X_1 \cup X_2 = X$. U_{X_1} and U_{X_2} are two unit vector sets projected into X_1* and *X_2 respectively. If existing two unit vectors $\mathbf{u}_1^* \in U_{X_1}$* and *$\mathbf{u}_2^* \in U_{X_2}$, two positive real numbers $l_1, l_2 > 0$, and a candidate solution $\mathbf{x}$ in search space to satisfy that*

$$\begin{aligned} &\Delta_1 \neq \Delta_2 \\ &\Delta_1 = \ln\left(f\left(\mathbf{x} + l_1\mathbf{u}_1^*\right)\right) - \ln(f(\mathbf{x})) \\ &\Delta_2 = \ln\left(f\left(\mathbf{x} + l_1\mathbf{u}_1^* + l_2\mathbf{u}_2^*\right)\right) - \ln\left(f\left(\mathbf{x} + l_2\mathbf{u}_2^*\right)\right). \end{aligned} \tag{25}$$

Notice that $f(\mathbf{x}+l_1\mathbf{u}_1^*+l_2\mathbf{u}_2^*)$, $f(\mathbf{x}+l_2\mathbf{u}_2^*)$, $f(\mathbf{x}+l_1\mathbf{u}_1^*)$, $f(\mathbf{x})$ are limited to positive, then subset X_1 has multiplicative interactions with subset X_2.

Proof 1: Variable-to-variable version of DDG has been proven in[36], which can be formulated as

$$\begin{aligned} &\Delta_1 = \ln(f(s_i)) - \ln(f(s)) \\ &\Delta_2 = \ln\left(f\left(s_{ij}\right)\right) - \ln\left(f\left(s_j\right)\right) \\ &\text{If } |\Delta_1 - \Delta_2| < \varepsilon_{multi} \\ &\text{Then } x_i \textit{ and } x_j \textit{ are multiplicatively separable} \end{aligned} \tag{26}$$

where s is a trial solution, s_i and s_j perturb δ on s in the ith and jth dimension respectively, and s_{ij} perturb δ in ith and jth dimension simultaneously. Here, we define $g(s) = \ln(f(s))$. Given $f(s) > 0$ is always true, thus, this definition holds, and Eq. (26) can be transformed to Eq. (27), which is equivalent to the canonical version of DG.

$$\begin{aligned} &\Delta_1 = g(s_i) - g(s) \\ &\Delta_2 = g\left(s_{ij}\right) - g\left(s_j\right) \\ &\text{If } |\Delta_1 - \Delta_2| < \varepsilon_{multi} \\ &\text{Then } x_i \textit{ and } x_j \textit{ are multiplicatively separable} \end{aligned} \tag{27}$$

the only difference is the allowable error ε_{multi}. Therefore, we can reasonably extend Eq. (27) to the subset-to-subset version based on the theoretical fundamentals in RDG [15] and MDG [17]:

$$\begin{aligned} &\Delta_1 = g\left(\mathbf{x}^* + l_1\mathbf{u}_1 + l_2\mathbf{u}_2\right) - g\left(\mathbf{x}^* + l_2\mathbf{u}_2\right) \\ &\Delta_2 = g\left(\mathbf{x}^* + l_1\mathbf{u}_1\right) - g\left(\mathbf{x}^*\right) \\ &\text{If } |\Delta_1 - \Delta_2| < \varepsilon_{multi} \\ &\text{Then } x_i \textit{ and } x_j \textit{ are multiplicatively separable} \end{aligned} \tag{28}$$

and we replace the $g(\mathbf{x})$ to $\ln(f(\mathbf{x}))$, which is the subset-to-subset version of DDG. Thus, Eq. (25) is true, and Theorem 1 is proven.

Another important component in EDDG is the threshold for interaction identification, many works have revealed the sensitivity of the DG mechanism to the threshold. Here, we summarize the popular design of DG-based methods in Table 1.

In Table 1, DG, FII, and DDG have fixed threshold ε for interaction identification, and this strategy cannot fit fitness landscapes with various characteristics dynamically. RDG adaptively sets its threshold ε based on the magnitude of the objective space [11], where $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{10}$ are randomly sampled solutions, and $\alpha = 10^{-12}$ is suggested in [67]. RDG2 and ERDG adopt the adaptive threshold setting based on the computational round-off error, where $\gamma_k = \frac{k\mu_m}{1-k\mu_m}$ are satisfied in the floating-point number system based on the IEEE 754 Standard [83]. MDG adapts the threshold ε according to the fitness function value $f(x)$, dimension size n, and the rounding error of floating numbers, where $f_{max} = \max\{|f(\mathbf{x}_{l,l})|, |f(\mathbf{x}_{u,l})|, |f(\mathbf{x}_{l,m})|, |f(\mathbf{x}_{u,m})|\}$. Similarly, we follow the design of the adaptive threshold in RDG2 and ERDG and extend it to our proposal EDDG, and this dynamic strategy can determine the threshold ε_{add} and ε_{multi} automatically and intelligently.

Optimization: SEADE

This research focuses on optimizing the LSEOPs which have a limited computational budget, thus we introduce a surrogate ensemble assistance technique to conventional DE. Figure 3 shows the flowchart of SEADE to optimize a sub-component.

For each sub-component optimization, SEADE first initializes the population and parameters and evaluates the population by the objective function. If the termination crite-

Table 1 The design of threshold ε in popular DG-based techniques

Algorithms	Setting of the threshold
DG [9]	10^{-3}
FII [82]	10^{-2}
RDG [15]	$\alpha * \min\{\lvert f(\mathbf{x}_1)\rvert, ..., \lvert f(\mathbf{x}_{10})\rvert\}$
RDG2 [12]	$\gamma_{\sqrt{n}+2} * (\lvert f(\mathbf{x}_{l,l})\rvert + \lvert f(\mathbf{x}_{u,l})\rvert + \lvert f(\mathbf{x}_{l,m})\rvert + \lvert f(\mathbf{x}_{u,m})\rvert)$
ERDG [16]	
DDG [36]	$\varepsilon_{add} = 10^{-3}$
	$\varepsilon_{multi} = 10^{-8}$
MDG [17]	$2^{-52} * c * n * f_{\max}$
EDDG	$\varepsilon_{add} = \gamma_{\sqrt{n}+2} * (\lvert f(\mathbf{x}_{l,l})\rvert + \lvert f(\mathbf{x}_{u,l})\rvert + \lvert f(\mathbf{x}_{l,m})\rvert + \lvert f(\mathbf{x}_{u,m})\rvert)$
	$\varepsilon_{multi} = \gamma_{\sqrt{n}+2} * (\lvert \ln(f(\mathbf{x}_{l,l}))\rvert + \lvert \ln(f(\mathbf{x}_{u,l}))\rvert + \lvert \ln(f(\mathbf{x}_{l,m}))\rvert + \lvert \ln(f(\mathbf{x}_{u,m}))\rvert)$

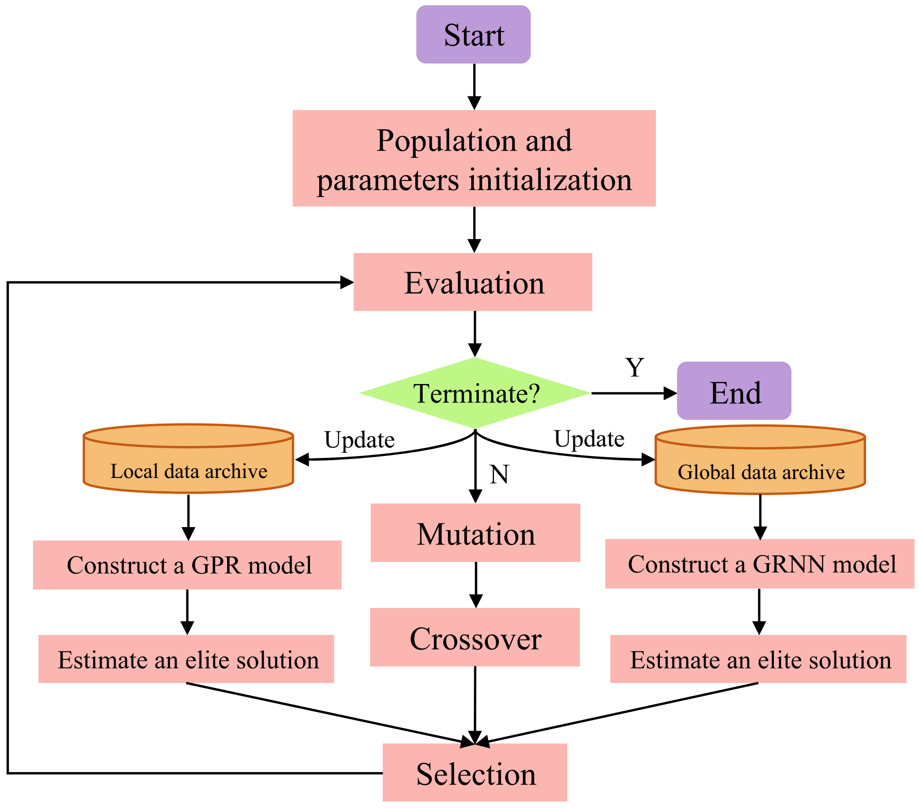

Fig. 3 The flowchart of SEADE

rion is not achieved, SEADE generates offspring individuals with three strategies: one individual estimated by the global surrogate model, one individual estimated by the local surrogate model, and $n-2$ individuals constructed by conventional schemes of mutation and crossover, where n is the population size. Then, the selection mechanism is adopted to choose the survival solutions. Next, we will details of our applied techniques.

Surrogate ensemble

Universal approximation theorems endow the neural networks (NN) with outstanding regression capacity and state that NN can represent a wide variety of functions when given appropriate weights, thus it has been extensively applied as the global surrogate model in many SAEAs [68, 84, 85]. Gaussian process regression (GPR), also known as the Kriging model, is a highly flexible model which can formulate uncertainty with great precision to build update points [86], and it has been employed as the local surrogate model in many pieces of literature [87–89]. Therefore, we combine both global and local surrogate models which are constructed by generalized regression neural network (GRNN) and GPR respectively in our proposed SEADE, and elite solutions estimated by these two surrogate models are expected to have high quality and accelerate the convergence of optimization.

Another component to construct the surrogate model is the training data. Figure 4 provides an example of two training data selection fashions for model construction. Figure 4 b selects the recent data to construct the GRP, which can approximate the local fitness landscape around the current individual. Figure 4c selects all historical data to construct the GRNN, and this scheme can learn the global information of the fitness landscape.

Besides, given the value of different samples may greatly differ, we normalized the fitness value of training data with the max-min scaler before surrogate model construction.

Infill sampling criterion

The infill sampling criterion is also known as the acquisition function [90], which is adopted to estimate elite solutions in the surrogate model. Here, we use the excepted improvement (EI) as the infill sampling criterion, which is formulated in Eq. (29)

$$\mathrm{EI}(x) = \max\left(f\left(x^*\right) - f(x), 0\right), \tag{29}$$

where x is the initial solution, and $f(x)$ is observed expectation from surrogate models. EI determines the next sampling solution x^* by maximizing the improvements from the initial solution. To search the x^* in the surrogate model, we simply implement the random search as the search engine, and the pseudocode is shown in Algorithm 4.

Algorithm 4 Random search

```
Require: Surrogate model: $f(\cdot)$, Initial solution: $x$, Lower and Upper bound: lb, ub
Ensure: Estimated solution: $x^*$
1: $x^* \leftarrow x$
2: while not Terminate do
3:     Construct solution $x'$ within lb and ub randomly
4:     if $f(x')$ has better EI than $f(x^*)$ then
5:         $x^* \leftarrow x'$
6:     end if
7: end while
8: return $x^*$
```

In summary, the pseudocode of our proposed SEADE is shown in Algorithm 5.

Algorithm 5 SEADE

```
Require: Population size: $S$, Maximum generation: $T$, Lower and Upper bound: lb, ub
Ensure: Optimum: $x^*$
1: $P \leftarrow$ popInitial($S$, lb, ub)
2: $x^* \leftarrow$ best($P$)
3: $t \leftarrow 0$
4: while $t < T$ do
5:     Generate $S - 2$ offspring $O$ with mutation and crossover
6:     Construct GRNN with all historical samples and GPR with current generation samples
7:     Estimate elites $e_1$ and $e_2$ by random search in Algorithm 4
8:     Add elites $e_1$ and $e_2$ to offspring $O$
9:     Evaluate the offspring $O$ and select survival individuals
10:    $x^* \leftarrow$ best($P$)
11: end while
12: return $x^*$
```

Numerical experiments

In this section, we implement numerical experiments to evaluate our proposal: SEADECC-EDDG. Section "Experiment settings" introduces the experiment settings, and Section "Experimental results" provides the experimental results of our proposal with compared methods.

Experiment settings

Experimental environments and implementation

SEADECC-EDDG is programmed with Python 3.11 and implemented in Hokkaido University's high-performance intercloud supercomputer, which is equipped with a CentOS operating system, Intel Xeon Gold 6148 CPU, and 384GB RAM. GRNN and GPR are provided by pyGRNN [91] and scikit-learn [92] libraries respectively.

Benchmark functions

We adopt the CEC2013 suite as our benchmark function, which consists of 15 test functions with 5 categories:

(1) f_1 to f_3: fully separable functions;

(2) f_4 to f_7: partially separable functions with 7 none-separable parts;

(3) f_8 to f_{11}: partially separable functions with 20 none-separable parts;

(4) f_{12} to f_{14}: functions with overlapping sub-problems;

(5) f_{15}: fully non-separable function;

f_1-f_{12} and f_{15} are 1000-D functions, and the rest f_{13} and f_{14} are 905-D, 3, 000, 000 FEs for decomposition and optimization are included in the standard CEC2013 suite. Due to our research focusing on LSEOPs, we limit the maximum FEs to $100 \times D$ including the decomposition and the optimization cost, where D is the dimension size. When D equals 1000, the maximum FEs is 100, 000.

Compared methods and parameters

Two parts in our proposal need to be evaluated: decomposition and optimization. The compared algorithms are listed in Table 2.

In the decomposition, the maximum scale of sub-compo-breaknents MS is 100. In the optimization, all decomposition

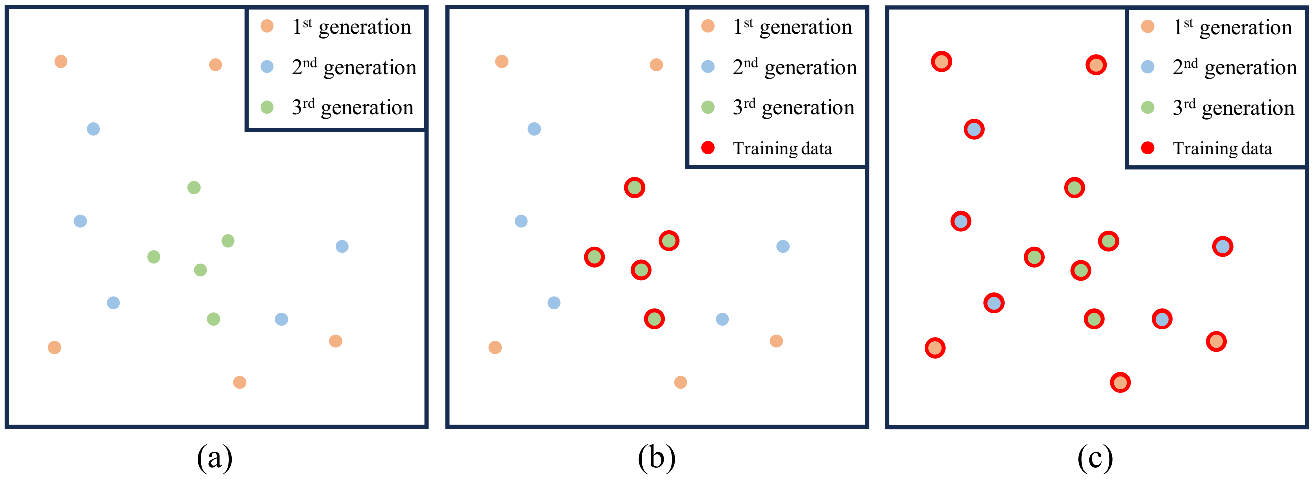

Fig. 4 Training data for constructing surrogate models, and the sample surrounded by the red circle is the selected data. **a** The distribution of individuals as the generation increases. **b** The training data for constructing the GPR. **c** The training data for constructing the GRNN

Table 2 A summary of the algorithms under comparison

Algorithms	Decomposition methods
DECC-RDG	RDG [15]
DECC-RDG2	RDG2 [12]
DECC-ERDG	ERDG [16]
DECC-MDG	MDG [17]
DECC-EDDG	EDDG
SEADECC-EDDG	

techniques are equipped with DE, where the population size is set to 10, scaling factor and crossover rate are set to 0.8 and 0.5 respectively. All compared methods are implemented in 25 trial runs to alleviate the impact of randomness. Besides, to investigate the effectiveness of the surrogate ensemble technique, we implement the ablation experiment among DECC-EDDG, SADECC-EDDG with the local surrogate model (SADECC-EDDG-i), SADECC-EDDG with the global surrogate model (SADECC-EDDG-ii), and SEADECC-EDDG to analyze the efficiency of the independent surrogate model in practice.

Performance indicators

This section introduces the metrics to evaluate the performance of EDDG and SEADE.

Metrics for EDDG: We apply three metrics to evaluate the performance of EDDG: decomposition accuracy (DA) [16], consumed FEs, and optimization with compared techniques. Here, we explain the computation of DA:

Let sep and sep' be the separable decision variables of decomposition methods and ideal decomposition respectively. $nonsep = \{g_1, g_2, ..., g_n\}$ and $nonsep' = \{g'_1, g'_2, ..., g'_n\}$ have the similar definition.

For separable decision variables:

$$DA_{sep} = \frac{|sep \cap sep'|}{|sep'|} \tag{30}$$

For non-separable decision variables:

$$DA_{nonsep} = \frac{\sum_{g' \in nonsep} |g'|}{\sum_{g' \in nonsep'} |g'|} \tag{31}$$

To evaluate the optimization performance among various decomposition techniques, we execute the optimization of all compared methods with 25 trial runs and calculate the mean and standard deviation (std). To identify the statistical significance, the Kruskal–Wallis test is employed to the fitness value at the end of optimization. If the significance exists, we further apply the Holm test whose p-value is acquired from the Mann–Whitney U test. $+$, $\approx$, and $-$ are applied to represent that our proposal is significantly better, with no significance, and significantly worse with the compared method, and the best value is in bold.

Metrics for SEADE To reveal the superiority and efficiency of the SEADE introduction, we apply the Holm test to the 25 trial runs of DECC-EDDG, SADECC-EDDG-i, SADECC-EDDG-ii, and SEADECC-EDDG. The statistical information is provided as well.

Experimental results

Comparison on decomposition

Table 3 summarizes the decomposition results including DA and consumed FEs among RDG2, ERDG, MDG, and EDDG on CEC2013 benchmark functions.

Comparison on optimization

Table 4 summarizes the optimization results among compared decomposition techniques, and Table 5 lists the optimization results between DECC-EDDG, SADECC-EDDG-i, SADECC-EDDG-ii, and SEADECC-EDDG to evaluate the performance of SEADE.

Discussion

In this section, we analyze the performance of our proposal SEADECC-EDDG and list some open topics for future research.

Performance analysis of EDDG

We evaluate the decomposition and optimization performance on four kinds of problems: f_1 to f_3 are fully separable, f_4 to f_{11} are partially separable with different characteristics, f_{12} to f_{14} are overlapping functions, and f_{15} is a fully non-separable function.

In fully separable functions f_1 to f_3, EDDG has a similar performance with the compared algorithms of RDG, RDG2, ERDG, and MDG in f_1 and f_2 that determines all decision variables as separable and achieve 100% decomposition accuracy DA, and the optimization results among compared decomposition techniques are without statistical significance. Meanwhile, we notice that all decomposition algorithms fail to detect the separability in f_3 and divide all separable decision variables into a component, we attribute this difficulty to the inherent property of Ackley's function which has a smooth and low-valued fitness landscape which directly causes the threshold ε also be strict. A similar phenomenon also appears in f_6, the shifted and rotated Ackley's function which contains 700 separable variables. Only RDG can identify a tiny part of separable variables, while the rest decomposition techniques identify this 700-D separable component as non-separable, and the optimization still suffers from the curse of dimensionality. However, our proposed EDDG can actively break the interactions and further decompose them into several sub-components, which can reduce the search space exponentially and accelerate the convergence of optimization significantly. Thus, we can observe that in the statistical analysis of optimization in f_3, our proposed DECC-EDDG has superiority to the compared DECC-RDG, DECC-RDG2, DECC-ERDG, and DECC-MDG.

$$\begin{tikzcd}
& f(x)=\sum_{i=1}^{1000}(x_i+x_{i+1})^2 \quad x_i\in[-5,5] \arrow[dl, "\mathcal{A}"'] \arrow[dr, "\mathcal{B}"] & \\
\{(x_1,x_2,\ldots,x_{1000})\} & & \{(x_1,\ldots,x_{500}),(x_{501},\ldots,x_{1000})\} \\
DA=100\% & & DA=50\%
\end{tikzcd}$$

Fig. 5 A simple example to illustrate the relationship between DA and the optimization performance for the overlapping function

In partially separable functions f_4 to f_{11}, EDDG has a similar but slight degenerating performance with RDG2, ERDG, and MDG on DA, which is due to the extra multiplicative interaction identification that may identify the additively non-separable interaction as multiplicatively separable and divide the corresponding sub-components. However, there is no statistical difference between optimization performance among DECC-RDG, DECC-RDG2, DECC-ERDG, DECC-MDG, and DECC-EDDG. Therefore, our proposed EDDG is competitive with compared decomposition algorithms in partially separable functions.

In overlapping functions f_{12} to f_{14}, RDG, RDG2, ERDG, and MDG correctly identify the interactions between decision variables and divide them into a sub-component. This decomposition degrades the optimization performance extremely due to the existence of the curse of dimensionality. Meanwhile, our proposed EDDG sacrifices the DA mostly to achieve optimization acceleration. A demonstration of the relationship between DA and the optimization performance of overlapping functions is shown in Fig. 5.

$f(x)=\sum_{i=1}^{1000}(x_i+x_{i+1})^2, x_i\in[-5,5]$ is an overlapping function. Assuming a decomposition algorithm $\mathcal{A}$ divides all decision variables into a component and DA achieves 100%, while a decomposition algorithm $\mathcal{B}$ divides the 1000-D problems into two equal-sized sub-components, and the DA of this decomposition extremely deteriorates to 50%. However, the optimization performance of decomposition $\mathcal{B}$ may perform better than $\mathcal{A}$ oppositely due to the search space of a sub-component decomposed by algorithm $\mathcal{B}$ is 10^{500}, while the search space of the component decomposed by algorithm $\mathcal{A}$ is 10^{1000}, which has 10^{500} times larger scale of search space than the sub-component in $\mathcal{B}$. Therefore, the balance between the effect of the curse of dimensionality and DA will benefit solving LSEOPs based on the CC framework, which is the main reason to explain that our proposed EDDG has lower DA than RDG, RDG2, ERDG, and MDG but the optimization performance of DECC-EDDG is higher than DECC-RDG, DECC-RDG2, DECC-ERDG, DECC-MDG, and DECC-EDDG on f_{12} to f_{14}.

Table 3 The DA and consumed FEs of RDG, RDG2, ERDG, MDG, and EDDG on CEC2013 suite

Func	RDG			RDG2			ERDG			MDG			EDDG		
	DA_{sep}	DA_{nonsep}	FEs	DA_{sep}	DA_{nonsep}	FEs	DA_{sep}	DA_{nonsep}	FEs	DA_{sep}	DA_{nonsep}	FEs	DA_{sep}	DA_{nonsep}	FEs
f_1	**100%**	–	3008	**100%**	–	2998	**100%**	–	2998	**100%**	–	**2002**	**100%**	–	2998
f_2	**100%**	–	3008	**100%**	–	2998	**100%**	–	2998	**100%**	–	**2002**	**100%**	–	2998
f_3	0%	–	6005	0%	–	5992	0%	–	3996	0%	–	**3001**	0%	–	3996
f_4	**100%**	**100%**	9842	**100%**	**100%**	9832	**100%**	**100%**	5326	**100%**	**100%**	**4072**	**100%**	**100%**	5330
f_5	**100%**	**100%**	10145	**100%**	**100%**	9895	**100%**	**100%**	5395	**100%**	**100%**	**4072**	99.6%	99.7%	5617
f_6	3.6%	91.7%	13574	0%	100%	11587	0%	91.7%	5905	0%	**100%**	7020	0%	91.7%	**5742**
f_7	27.3%	0%	11381	**100%**	**100%**	9814	**100%**	**100%**	5554	**100%**	**100%**	**4104**	**100%**	**100%**	5444
f_8	–	70%	19364	–	**80%**	19405	–	75%	8451	–	**80%**	9974	–	65.6%	**7981**
f_9	–	**100%**	19343	–	**100%**	19156	–	**100%**	**8812**	–	**100%**	11666	–	**100%**	8885
f_{10}	–	85%	19178	–	**100%**	19879	–	87.5%	8794	–	**100%**	11588	–	77.5%	**4326**
f_{11}	–	10%	10496	–	**100%**	19429	–	**100%**	**9212**	–	**100%**	11668	–	98.5%	9296
f_{12}	–	**100%**	50, 876	–	**100%**	50, 866	–	**100%**	26, 980	–	**100%**	**3001**	–	10%	29100
f_{13}	–	99.8%	8345	–	58.0%	15187	–	58.0%	7599	–	**100%**	9440	–	10%	**7583**
f_{14}	–	**100%**	9542	–	**100%**	16, 150	–	**100%**	8420	–	**100%**	9567	–	10%	**8170**
f_{15}	–	**100%**	6173	–	**100%**	5992	–	**100%**	3996	–	**100%**	**3001**	–	10%	3996

Table 4 Optimization results of DECC-RDG, DECC-RDG2, DECC-ERDG, DECC-MDG, and DECC-EDDG

Func	DECC-RDG		DECC-RDG2		DECC-ERDG		DECC-MDG		DECC-EDDG	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
f_1	**2.43e+08** ≈	**8.67e+07**	2.54e+08 ≈	1.05e+08	3.40e+08 ≈	5.29e+08	3.09e+08 ≈	5.36e+08	2.59e+08	2.17e+08
f_2	**1.04e+03** ≈	**3.61e+01**	1.06e+03 ≈	4.31e+01	1.04e+03 ≈	3.87e+01	1.06e+03 ≈	3.98e+01	1.05e+03	2.60e+01
f_3	2.15e+01 +	1.24e–02	2.15e+01 +	8.92e–03	2.15e+01 +	8.40e–03	2.15e+01 +	9.62e–03	**2.12e+01**	**9.67e–03**
f_4	1.57e+13 ≈	5.46e+12	1.82e+13 ≈	7.42e+12	**1.47e+13** ≈	**4.00e+12**	1.63e+13 ≈	6.42e+12	1.62e+13	5.31e+12
f_5	4.80e+07 ≈	5.02e+06	4.82e+07 ≈	5.58e+06	4.62e+07 ≈	3.62e+06	**4.47e+07** ≈	**4.58e+06**	4.72e+07	3.41e+06
f_6	1.07e+06 +	1.29e+03	1.07e+06 ≈	1.55e+03	1.07e+06 ≈	8.25e+02	**1.07e+06** ≈	**1.44e+03**	1.07e+06	1.67e+03
f_7	8.99e+14 +	7.90e+14	2.59e+13 ≈	1.91e+13	2.53e+13 ≈	2.01e+13	**2.08e+13** ≈	**1.37e+13**	2.48e+13	1.33e+13
f_8	6.72e+17 ≈	1.78e+17	6.71e+17 ≈	1.93e+17	6.08e+17 ≈	1.79e+17	**6.00e+17** ≈	**2.02e+17**	6.48e+17	1.89e+17
f_9	1.06e+09 ≈	6.41e+07	1.05e+09 ≈	5.99e+07	**1.01e+09** ≈	**6.85e+07**	1.02e+09 ≈	6.95e+07	1.02e+09	6.87e+07
f_{10}	9.59e+07 ≈	3.13e+05	**9.53e+07** ≈	**4.06e+05**	9.54e+07 ≈	3.08e+05	9.54e+07 ≈	2.61e+05	9.58e+07	3.28e+05
f_{11}	2.31e+13 +	2.53e+13	**5.92e+11** ≈	**2.18e+11**	6.45e+11 ≈	2.05e+11	6.26e+11 ≈	1.86e+11	6.53e+11	1.44e+11
f_{12}	9.12e+11 +	1.03e+11	8.83e+11 +	8.09e+10	6.38e+11 +	7.08e+10	4.98e+11 +	6.24e+10	**6.57e+10**	**8.30e+09**
f_{13}	4.38e+13 +	4.17e+13	4.10e+12 +	5.22e+12	3.21e+12 +	3.60e+12	1.02e+13 +	1.21e+13	**1.01e+11**	**1.53e+10**
f_{14}	3.62e+13 +	1.57e+13	4.41e+13 +	2.50e+13	3.56e+13 +	2.17e+13	3.68e+13 +	1.97e+13	**9.85e+11**	**1.72e+11**
f_{15}	3.55e+10 −	6.86e+10	1.70e+10 −	3.82e+10	**9.12e+09** −	**2.08e+10**	1.62e+10 −	3.54e+10	2.52e+12	7.08e+09
	+/≈/−: 7/7/1		+/≈/−: 4/10/1		+/≈/−: 4/10/1		+/≈/−: 4/10/1			

Table 5 Optimization results of DECC-EDDG, SADECC-EDDG-i, SADECC-EDDG-ii, and SEADECC-EDDG

Func	DECC-EDDG		SADECC-EDDG-i		SADECC-EDDG-ii		SEADECC-EDDG	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
f_1	2.59e+08 +	2.17e+08	3.53e+07 +	4.91e+07	1.83e+08 +	5.30e+07	**1.55e+07**	**9.65e+06**
f_2	1.05e+03 +	2.60e+01	6.49e+02 +	2.40e+01	1.06e+03 +	2.66e+01	**5.73e+02**	**2.14e+01**
f_3	**2.12e+01** ≈	**9.67e–03**	2.12e+01 ≈	7.75e–03	2.12e+01 ≈	1.02e–02	2.12e+01	6.64e–03
f_4	1.62e+13 ≈	5.31e+12	1.47e+13 ≈	1.67e+12	1.64e+13 ≈	2.06e+12	**1.44e+13**	**4.44e+12**
f_5	4.72e+07 +	3.41e+06	4.23e+07 ≈	5.18e+06	4.83e+07 +	3.84e+06	**3.93e+07**	**4.17e+06**
f_6	1.07e+06 ≈	1.67e+03	1.07e+06 ≈	9.80e+02	1.07e+06 ≈	8.97e+02	**1.07e+06**	**1.35e+02**
f_7	2.48e+13 +	1.33e+13	1.80e+13 +	8.86e+12	2.11e+13 +	9.19e+12	**7.88e+12**	**4.46e+12**
f_8	6.48e+17 +	1.89e+17	6.31e+17 +	1.80e+17	5.74e+17 ≈	2.02e+17	**4.40e+17**	**1.05e+17**
f_9	1.02e+09 +	6.87e+07	9.59e+08 +	6.83e+07	9.92e+08 +	6.99e+07	**8.75e+08**	**5.41e+07**
f_{10}	**9.58e+07** ≈	**3.28e+05**	9.59e+07 ≈	2.08e+05	9.60e+07 ≈	2.75e+05	9.58e+07	1.29e+05
f_{11}	6.54e+11 ≈	1.45e+11	6.80e+11 ≈	1.05e+11	6.03e+11 ≈	1.55e+11	**5.90e+11**	**1.14e+11**
f_{12}	6.57e+10 +	8.30e+09	3.22e+10 +	4.19e+09	3.63e+10 +	3.57e+09	**1.47e+10**	**1.28e+09**
f_{13}	1.01e+11 +	1.53e+10	8.84e+10 +	1.48e+10	8.83e+10 +	1.60e+10	**7.29e+10**	**1.32e+10**
f_{14}	9.85e+11 +	1.72e+11	7.79e+11 ≈	4.99e+10	8.43e+11 +	8.26e+10	**7.38e+11**	**8.07e+10**
f_{15}	**2.52e+12** −	**7.08e+09**	2.55e+12 −	6.12e+09	2.55e+12 −	5.57e+09	2.57e+12	5.05e+09
	+/≈/−: 9/5/1		+/≈/−: 7/7/1		+/≈/−: 8/6/1			

In the fully non-separable function f_{15}, EDDG also sacrifices the DA and to further decomposes the sub-component with a relatively large scale, thus the deterioration of DA can be observed from Table 3 compared with RDG, RDG2, ERDG, and MDG. However, the optimization of DECC-EDDG is significantly inferior to DECC-RDG, DECC-RDG2, DECC-ERDG, DECC-MDG, and DECC-EDDG. In this situation, the accuracy of decomposition for decision variables is more important in optimization than the alleviation of the curse of dimensionality, and we re-emphasize the importance of the balance between the DA and the effect of the curse of dimensionality in the specific large-scale optimization task, which is a challenging topic for our future research.

Performance analysis of SEADE

Ablation experiments are provided to investigate the efficiency of the surrogate ensemble technique, and the experimental and statistical results can be observed in Table 5. GPR as a non-parametric regression model is good at dealing with relatively low- and median-dimensional regression tasks, and GRNN can approximate any high-dimensional problems theoretically. From these numerical experiments and statistical summary, we conclude that the surrogate ensemble technique can accelerate the convergence of optimization significantly in most cases, and the cooperation of the global and the local surrogate model is better than the single surrogate model in practice. In the meantime, we notice an abnormal phenomenon that the introduction of surrogate ensemble assistance will significantly decelerate the optimization in f_{15}, and we infer that this deterioration is caused by the poor decomposition of EDDG in f_{15}. Observing the experimental results in Table 4, our proposal EDDG is not efficient on fully non-separable function f_{15}, and this poor decomposition leads the direction of optimization in the wrong way. Although the surrogate ensemble technique can estimate high-quality elite solutions, the poor decomposition will destroy the fitness landscape, and further influence the real quality of estimated solutions in the original fitness landscape.

Open topics for future research

The above experimental results and analysis show our proposed SEADECC-EDDG is a promising approach to dealing with LSEOPs, however, there are still some open topics for future improvements:

Estimating promising sub-regions

CC framework adopts the decomposition techniques to investigate the interrelationship between decision variables and divides the LSOPs into multiple sub-components, which can alleviate the curse of dimensionality. And another technique named search space reduction (SSR), which limits the optimization within promising sub-regions is also suitable to deal with LSOPs. Here, we give a simple mathematical explanation: Supposing that the dimension of the problem is N, and the search space of a decision variable is a. Simply, the total search space is a^N. If an SSR technique can reduce 10% search space in every dimension, then, the new search space becomes $(0.9a)^N$ and the proportion between new and original search space will be 0.9^N. When N is [10, 100, 500, 1000], this proportion is [34.86%, 2.65e-03%, 1.32e-21%, 1.74e-44%]. As the dimension increases, the proportion will decrease rapidly, and the optimization in the limited promising sub-regions will be significantly accelerated by the SSR technique. In the future, the design of the criterion to employ the SSR technique is promising to tackle LSOPs.

The trade-off between decomposition and optimization

We point out that one conflict in solving LSEOPs based on the CC framework is the computational resource allocation between decomposition and optimization. In a more severe case when the computational budget cannot afford the full decomposition of LSEOPs by RDG-based techniques, the balance of resource allocation between decomposition and optimization becomes more important. A feasible approach is to sacrifice some accuracy to realize the decomposition, and the effectiveness has been proven by our experiments and reported by other publications [35]. Specifically, we first manually separate the LSEOP in order or randomly, then RDG-based methods can be applied to decompose each sub-component in parallel. A mathematical explanation is that supposing the dimension of the problem is N, and the time complexity of RDG-based methods is $O(N \log(N))$. If we first divide this problem into k sub-components, and the total time complexity becomes $\sum_{i=1}^{k}(O((N/k)\log(N/k))) = O(N\log(N/k))$. Although the initial random grouping or in-order grouping will decrease the decomposition accuracy, the necessary FEs for decomposition can also be saved, which has stronger scalability to tackle larger-scale EOPs.

Various interaction identification

Our proposed EDDG can detect the additive and multiplicative interactions between decision variables rather than conventional DE/RDG-based techniques which only focus on additive interaction determination. However, other kinds of interactions also exist such as composite separability [81]. A simple example is that if $f(x) = U(g(x))$, where $U(\cdot)$ is monotonically increasing in its search space, $g(x)$ is a separable function, then $f(x)$ is a compositely separable function. Therefore, a remaining problem is whether can we detect composite separability based on the DG-based mechanism, which is a meaningful topic to promote the development of the CC community.

Conclusion

This paper proposes a novel algorithm named SEADECC-EDDG to solve LSEOPs based on the CC framework. In the decomposition, we design the EDDG which contains the efficiency of ERDG and the multiplicative interaction identification principle of DDG. Active interaction break operation accelerates the optimization at the cost of decomposition accuracy. In the optimization, we introduce a surrogate ensemble assisted DE as the basic optimizer. The global surrogate model constructed by generalized regression neural network (GRNN) and historical samples can describe the characteristics of the whole fitness landscape, and the local surrogate model constructed by Gaussian process regression (GPR) and current generation solutions can approximate the regularity of the fitness landscape around the recent samples. Estimated elite solutions by the combination of expected improvement (EI) and the random search from surrogate models are expected to accelerate the convergence of the optimization.

In the future, we will focus on introducing the search space reduction technique (SSR) to LSOPs, detecting various interactions with RDG-based frameworks, and solving more complex LSEOPs.

Acknowledgements This work was supported by JSPS KAKENHI Grant number JP20K11967, JST SPRING Grant number JPMJSP2119, and Interdisciplinary large-scale computer system (Supercomputing system), Information Initiative Center, Hokkaido University.

Data availability The source code of this research can be downloaded from https://github.com/RuiZhong961230/SEADECC-EDDG.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Hiba H, Ibrahim A, Rahnamayan S (2019) Large-scale optimization using center-based differential evolution with dynamic mutation scheme. In: 2019 IEEE Congress on evolutionary computation (CEC), pp 3189–3196. https://doi.org/10.1109/CEC.2019.8789992
2. Wang Z-J, Zhan Z-H, Yu W-J, Lin Y, Zhang J, Gu T-L, Zhang J (2020) Dynamic group learning distributed particle swarm optimization for large-scale optimization and its application in cloud workflow scheduling. IEEE Trans Cybern 50(6):2715–2729. https://doi.org/10.1109/TCYB.2019.2933499
3. Yi J-H, Xing L-N, Wang G-G, Dong J, Vasilakos AV, Alavi AH, Wang L (2020) Behavior of crossover operators in nsga-iii for large-scale optimization problems. Inf Sci 509:470–487. https://doi.org/10.1016/j.ins.2018.10.005
4. Gaur A, Talukder AKMK, Deb K, Tiwari S, Xu S, Jones D (2017) Finding near-optimum and diverse solutions for a large-scale engineering design problem. In: 2017 IEEE Symposium Series on computational intelligence (SSCI), pp 1–8. https://doi.org/10.1109/SSCI.2017.8285271
5. Shahrouzi M, Salehi A (2020) Design of large-scale structures by an enhanced metaheuristic utilizing opposition-based learning. In: 2020 4th Conference on swarm intelligence and evolutionary computation (CSIEC), pp 027–031. https://doi.org/10.1109/CSIEC49655.2020.9237319
6. Feng L, Shang Q, Hou Y, Tan KC, Ong Y-S (2023) Multispace evolutionary search for large-scale optimization with applications to recommender systems. IEEE Trans Artif Intell 4(1):107–120. https://doi.org/10.1109/TAI.2022.3156952
7. Köppen M (2000) The curse of dimensionality. In: 5th Online World Conference on soft computing in industrial applications (WSC5), vol 1, pp 4–8
8. Potter MA, De Jong KA (1994) A cooperative coevolutionary approach to function optimization. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 866 LNCS, 249–257
9. Omidvar MN, Li X, Mei Y, Yao X (2014) Cooperative co-evolution with differential grouping for large scale optimization. IEEE Trans Evol Comput 18(3):378–393. https://doi.org/10.1109/TEVC.2013.2281543
10. Ling Y, Li H, Cao B (2016) Cooperative co-evolution with graph-based differential grouping for large scale global optimization. In: 2016 12th International Conference on natural computation, fuzzy systems and knowledge discovery (ICNC-FSKD), pp 95–102. https://doi.org/10.1109/FSKD.2016.7603157
11. Mei Y, Li Omidvar MN., X, Yao X, (2016) A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization. ACM Trans Math Softw. https://doi.org/10.1145/2791291
12. Sun Y, Omidvar MN, Kirley M, Li X (2018) Adaptive threshold parameter estimation with recursive differential grouping for problem decomposition. In: Proceedings of the Genetic and Evolutionary Computation Conference. GECCO '18, pp 889–896. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3205455.3205483
13. Jin Y, Sendhoff B (2009) A systems approach to evolutionary multiobjective structural optimization and beyond. IEEE Comput Intell Mag 4(3):62–76. https://doi.org/10.1109/MCI.2009.933094
14. Rashidi S, Ranjitkar P (2015) Bus dwell time modeling using gene expression programming. Comput-Aided Civ Infrastr Eng. https://doi.org/10.1111/mice.12125
15. Sun Y, Kirley M, Halgamuge SK (2018) A recursive decomposition method for large scale continuous optimization. IEEE Trans Evol Comput 22(5):647–661. https://doi.org/10.1109/TEVC.2017.2778089
16. Yang M, Zhou A, Li C, Yao X (2021) An efficient recursive differential grouping for large-scale continuous problems. IEEE Trans Evol Comput 25(1):159–171. https://doi.org/10.1109/TEVC.2020.3009390
17. Ma X, Huang Z, Li X, Wang L, Qi Y, Zhu Z (2022) Merged differential grouping for large-scale global optimization. IEEE Trans Evol Comput 26(6):1439–1451. https://doi.org/10.1109/TEVC.2022.3144684
18. Evolutionary algorithms. In: Bidgoli H (ed) Encyclopedia of Information Systems, pp 259–267. Elsevier, New York (2003). https://doi.org/10.1016/B0-12-227240-4/00065-4
19. Chugh T, Sindhya K, Hakanen J, Miettinen K (2019) A survey on handling computationally expensive multiobjective optimization

problems with evolutionary algorithms. Soft Comput 23(9):3137–3166. https://doi.org/10.1007/s00500-017-2965-0
20. Jin Y (2011) Surrogate-assisted evolutionary computation: recent advances and future challenges. Swarm Evol Comput 1(2):61–70. https://doi.org/10.1016/j.swevo.2011.05.001
21. Amouzgar K, Bandaru S, Ng AHC (2018) Radial basis functions with a priori bias as surrogate models: a comparative study. Eng Appl Artif Intell 71:28–44. https://doi.org/10.1016/j.engappai.2018.02.006
22. Zhou Z, Ong YS, Nguyen MH, Lim D (2005) A study on polynomial regression and gaussian process global surrogate model in hierarchical surrogate-assisted evolutionary algorithm. In: 2005 IEEE Congress on evolutionary computation, vol 3, pp 2832–28393. https://doi.org/10.1109/CEC.2005.1555050
23. Liu B, Zhang Q, Gielen GGE (2014) A gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems. IEEE Trans Evol Comput 18(2):180–192. https://doi.org/10.1109/TEVC.2013.2248012
24. Song Z, Wang H, He C, Jin Y (2021) A kriging-assisted two-archive evolutionary algorithm for expensive many-objective optimization. IEEE Trans Evol Comput 25(6):1013–1027. https://doi.org/10.1109/TEVC.2021.3073648
25. Atashkari K, Nariman-Zadeh N, Gölcü M, Khalkhali A, Jamali A (2007) Modelling and multi-objective optimization of a variable valve-timing spark-ignition engine using polynomial neural networks and evolutionary algorithms. Energy Convers Manag 48(3):1029–1041. https://doi.org/10.1016/j.enconman.2006.07.007
26. Li F, Cai X, Gao L, Shen W (2021) A surrogate-assisted multiswarm optimization algorithm for high-dimensional computationally expensive problems. IEEE Trans Cybern 51(3):1390–1402. https://doi.org/10.1109/TCYB.2020.2967553
27. Cai X, Qiu H, Gao L, Jiang C, Shao X (2019) An efficient surrogate-assisted particle swarm optimization algorithm for high-dimensional expensive problems. Knowl-Based Syst 184:104901. https://doi.org/10.1016/j.knosys.2019.104901
28. Cai X, Gao L, Li X (2020) Efficient generalized surrogate-assisted evolutionary algorithm for high-dimensional expensive problems. IEEE Trans Evol Comput 24(2):365–379. https://doi.org/10.1109/TEVC.2019.2919762
29. De Falco I, Della Cioppa A, Trunfio GA (2019) Investigating surrogate-assisted cooperative coevolution for large-scale global optimization. Inf Sci 482:1–26. https://doi.org/10.1016/j.ins.2019.01.009
30. Ren Z, Pang B, Wang M, Feng Z, Liang Y, Chen A, Zhang Y (2019) Surrogate model assisted cooperative coevolution for large scale optimization. Appl Intell 49(2):513–531. https://doi.org/10.1007/s10489-018-1279-y
31. Sun M, Sun C, Li X, Zhang G, Akhtar F (2022) Surrogate ensemble assisted large-scale expensive optimization with random grouping. Inf Sci 615:226–237. https://doi.org/10.1016/j.ins.2022.09.063
32. Sun M, Sun C, Li X, Zhang G, Akhtar F (2022) Large-scale expensive optimization with a switching strategy. Complex Syst Model Simul 2(3):253–263. https://doi.org/10.23919/CSMS.2022.0013
33. Gu H, Wang H, Jin Y (2022) Surrogate-assisted differential evolution with adaptive multi-subspace search for large-scale expensive optimization. IEEE Trans Evol Comput. https://doi.org/10.1109/TEVC.2022.3226837
34. Tian Y, Si L, Zhang X, Cheng R, He C, Tan KC, Jin Y (2021) Evolutionary large-scale multi-objective optimization: a survey. ACM Comput Surv. https://doi.org/10.1145/3470971
35. Sun Y, Li X, Ernst A, Omidvar MN (2019) Decomposition for large-scale optimization problems with overlapping components. In: 2019 IEEE Congress on evolutionary computation (CEC), pp 326–333. https://doi.org/10.1109/CEC.2019.8790204
36. Li J-Y, Zhan Z-H, Tan KC, Zhang J (2022) Dual differential grouping: a more general decomposition method for large-scale optimization. IEEE Trans Cybern. https://doi.org/10.1109/TCYB.2022.3158391
37. van den Bergh F, Engelbrecht AP (2004) A cooperative approach to particle swarm optimization. IEEE Trans Evol Comput 8(3):225–239. https://doi.org/10.1109/TEVC.2004.826069
38. Storn R (1996) On the usage of differential evolution for function optimization. In: Proceedings of North American fuzzy information processing, pp 519–523. https://doi.org/10.1109/NAFIPS.1996.534789
39. Seyedpoor S, Shahbandeh S, Yazdanpanah O (2015) An efficient method for structural damage detection using a differential evolution algorithm based optimization approach. Civ Eng Environ Syst. https://doi.org/10.1080/10286608.2015.1046051
40. Emidio B, Gomes G, Silva W, Silva RSYRC, Bezerra L, Lopez Palechor E (2020) Differential evolution algorithm for identification of structural damage in steel beams. Frattura ed Integrità Strutturale 14:51–66. https://doi.org/10.3221/IGF-ESIS.52.05
41. Seyedpoor SM, Pahnabi N (2021) Structural damage identification using frequency domain responses and a differential evolution algorithm. Iran J Sci Technol Trans Civ Eng. https://doi.org/10.1007/s40996-020-00528-0
42. Awad N, Mallik N, Hutter F (2021) Differential evolution for neural architecture search arXiv e-prints (2020): arXiv-2012
43. Gülcü A, Kuş Z (2023) Neural architecture search using differential evolution in MAML framework for few-shot classification problems. In: Di Gaspero L, Festa P, Nakib A, Pavone M (eds) Metaheuristics. MIC 2022. Lecture Notes in Computer Science, vol 13838. Springer, Cham. https://doi.org/10.1007/978-3-031-26504-4_11
44. Mahdadi A, Meshoul S (2015) A multiobjective integer differential evolution approach for computer aided drug design. In: 2015 3rd International Conference on control, engineering & information technology (CEIT), pp 1–6. https://doi.org/10.1109/CEIT.2015.7233161
45. Shahzad W, Yawar A, Ahmed E (2016) Drug design and discovery using differential evolution. J Biol Environ Sci 6:16–26
46. Specht DF (1991) A general regression neural network. IEEE Trans Neural Netw 2(6):568–576. https://doi.org/10.1109/72.97934
47. Liang Y, Niu D, Hong W-C (2019) Short term load forecasting based on feature extraction and improved general regression neural network model. Energy 166:653–663. https://doi.org/10.1016/j.energy.2018.10.119
48. Hornik K (1991) Approximation capabilities of multilayer feedforward networks. Neural Netw 4(2):251–257. https://doi.org/10.1016/0893-6080(91)90009-T
49. Wei W, Jiang J, Liang H, Gao L, Liang B, Huang J, Zang N, Liao Y, Yu J, Lai J, Qin F, Su J, Ye L, Chen H (2016) Application of a combined model with autoregressive integrated moving average (arima) and generalized regression neural network (grnn) in forecasting hepatitis incidence in heng county, china. PLoS One 11(6):1–13. https://doi.org/10.1371/journal.pone.0156768
50. Leung M, Chen A-S, Mancha R (2009) Making trading decisions for financial engineered derivatives: a novel ensemble of neural networks using information content. Int Syst Acc Finance Manag 16:257–277. https://doi.org/10.1002/isaf.308
51. Cheng R, Jin Y, Narukawa K, Sendhoff B (2015) A multiobjective evolutionary algorithm using gaussian process-based inverse modeling. IEEE Trans Evol Comput 19(6):838–856. https://doi.org/10.1109/TEVC.2015.2395073
52. Buche D, Schraudolph NN, Koumoutsakos P (2005) Accelerating evolutionary algorithms with gaussian process fitness function models. IEEE Trans Syst Man Cybern Part C (Applications and Reviews) 35(2):183–194. https://doi.org/10.1109/TSMCC.2004.841917

53. Zhang Q, Liu W, Tsang E, Virginas B (2010) Expensive multi-objective optimization by moea/d with gaussian process model. IEEE Trans Evol Comput 14(3):456–474. https://doi.org/10.1109/TEVC.2009.2033671
54. Liu K, Hu X, Wei Z, Li Y, Jiang Y (2019) Modified gaussian process regression models for cyclic capacity prediction of lithium-ion batteries. IEEE Trans Transp Electr 5(4):1225–1236. https://doi.org/10.1109/TTE.2019.2944802
55. Yang D, Zhang X, Pan R, Wang Y, Chen Z (2018) A novel gaussian process regression model for state-of-health estimation of lithium-ion battery using charging curve. J Power Sources 384:387–395. https://doi.org/10.1016/j.jpowsour.2018.03.015
56. Liu D, Pang J, Zhou J, Peng Y, Pecht M (2013) Prognostics for state of health estimation of lithium-ion batteries based on combination gaussian process functional regression. Microelectron Reliab 53(6):832–839. https://doi.org/10.1016/j.microrel.2013.03.010
57. Rasmussen CE, Nickisch H (2010) Gaussian processes for machine learning (gpml) toolbox. J Mach Learn Res 11:3011–3015
58. Yu H, Tan Y, Sun C, Zeng J (2019) A generation-based optimal restart strategy for surrogate-assisted social learning particle swarm optimization. Knowl-Based Syst 163:14–25. https://doi.org/10.1016/j.knosys.2018.08.010
59. Di Nuovo AG, Ascia G, Catania V (2012) A study on evolutionary multi-objective optimization with fuzzy approximation for computational expensive problems. In: Parallel Problem Solving from Nature—PPSN XII, pp 102–111. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-32964-7_11
60. Hansen N (2019) A global surrogate assisted cma-es. In: Proceedings of the Genetic and Evolutionary Computation Conference. GECCO ’19, pp 664–672. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3321707.3321842
61. Pan J-S, Zhang L-G, Chu S-C, Shieh C-S, Watada J (2023) Surrogate-assisted hybrid meta-heuristic algorithm with an add-point strategy for a wireless sensor network. Entropy. https://doi.org/10.3390/e25020317
62. Han L, Wang H, Wang S (2022) A surrogate-assisted evolutionary algorithm for space component thermal layout optimization. Sp Sci Technol 15:10. https://doi.org/10.34133/2022/9856362
63. Liezl Stander TLVZ, Woolway M (2022) Surrogate assisted evolutionary multi-objective optimisation applied toa pressure swing adsorption system. Neural Comput Appl. https://doi.org/10.1007/s00521-022-07295-1
64. Martínez SZ, Coello CAC (2013) Combining surrogate models and local search for dealing with expensive multi-objective optimization problems. In: 2013 IEEE Congress on evolutionary computation, pp 2572–2579. https://doi.org/10.1109/CEC.2013.6557879
65. Lim D, Jin Y, Ong Y-S, Sendhoff B (2010) Generalizing surrogate-assisted evolutionary computation. IEEE Trans Evol Comput 14(3):329–355. https://doi.org/10.1109/TEVC.2009.2027359
66. Sun C, Zeng J, Pan J, Xue S, Jin Y (2013) A new fitness estimation strategy for particle swarm optimization. Inf Sci 221:355–370. https://doi.org/10.1016/j.ins.2012.09.030
67. Sun C, Jin Y, Cheng R, Ding J, Zeng J (2017) Surrogate-assisted cooperative swarm optimization of high-dimensional expensive problems. IEEE Trans Evol Comput 21(4):644–660. https://doi.org/10.1109/TEVC.2017.2675628
68. Wang Y, Yin D-Q, Yang S, Sun G (2019) Global and local surrogate-assisted differential evolution for expensive constrained optimization problems with inequality constraints. IEEE Trans Cybern 49(5):1642–1656. https://doi.org/10.1109/TCYB.2018.2809430
69. Wang X, Wang GG, Song B, Wang P, Wang Y (2019) A novel evolutionary sampling assisted optimization method for high-dimensional expensive problems. IEEE Trans Evol Comput 23(5):815–827. https://doi.org/10.1109/TEVC.2019.2890818
70. Yu M, Li X, Liang J (2020) A dynamic surrogate-assisted evolutionary algorithm framework for expensive structural optimization. Struct Multidisc Optim 61:711–729. https://doi.org/10.1007/s00158-019-02391-8
71. Chen G, Zhang K, Xue X, Zhang L, Yao J, Sun H, Fan L, Yang Y (2020) Surrogate-assisted evolutionary algorithm with dimensionality reduction method for water flooding production optimization. J Pe Sci Eng 185:106633. https://doi.org/10.1016/j.petrol.2019.106633
72. Zhou J, Wang H, Xiao C, Zhang S (2023) Hierarchical surrogate-assisted evolutionary algorithm for integrated multi-objective optimization of well placement and hydraulic fracture parameters in unconventional shale gas reservoir. Energies. https://doi.org/10.3390/en16010303
73. Tang Z, Xu L, Luo S (2022) Adaptive dynamic surrogate-assisted evolutionary computation for high-fidelity optimization in engineering. Appl Soft Comput 127:109333. https://doi.org/10.1016/j.asoc.2022.109333
74. Yang Z, Tang K, Yao X (2007) Differential evolution for high-dimensional function optimization. In: 2007 IEEE Congress on evolutionary computation, pp 3523–3530. https://doi.org/10.1109/CEC.2007.4424929
75. Yang Z, Tang K, Yao X (2008) Large scale evolutionary optimization using cooperative coevolution. Inf Sci 178(15):2985–2999. https://doi.org/10.1016/j.ins.2008.02.017
76. Li X, Yao X (2009) Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms. In: 2009 IEEE Congress on evolutionary computation, pp 1546–1553. https://doi.org/10.1109/CEC.2009.4983126
77. Yang Z, Tang K, Yao X (2008) Multilevel cooperative coevolution for large scale optimization. In: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), pp 1663–1670. https://doi.org/10.1109/CEC.2008.4631014
78. Munetomo M, Goldberg DE (1999) Linkage identification by non-monotonicity detection for overlapping functions. Evol Comput 7(4):377–398. https://doi.org/10.1162/evco.1999.7.4.377
79. Tezuka M, Munetomo M, Akama K (2004) Linkage identification by nonlinearity check for real-coded genetic algorithms. Lecture Notes in Computer Science (including subseries In: Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 3103:222–233
80. Omidvar MN, Yang M, Mei Y, Li X, Yao X (2017) DG2: a faster and more accurate differential grouping for large-scale black-box optimization. IEEE Trans Evol Comput 21(6):929–942. https://doi.org/10.1109/TEVC.2017.2694221
81. Chen M, Du W, Tang Y, Jin Y, Yen GG (2022) A decomposition method for both additively and non-additively separable problems. IEEE Trans Evol Comput. https://doi.org/10.1109/TEVC.2022.3218375
82. Hu X-M, He F-L, Chen W-N, Zhang J (2017) Cooperation coevolution with fast interdependency identification for large scale optimization. Inf Sci 381:142–160. https://doi.org/10.1016/j.ins.2016.11.013
83. Ieee standard for floating-point arithmetic. In: IEEE Std 754-2019 (Revision of IEEE 754-2008), 1–84 (2019). https://doi.org/10.1109/IEEESTD.2019.8766229
84. Dushatskiy A, Mendrik AM, Alderliesten T, Bosman PAN (2019) Convolutional neural network surrogate-assisted gomea. In: Proceedings of the Genetic and Evolutionary Computation Conference. GECCO ’19, pp 753–761. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3321707.3321760
85. Sarkari Khorrami M, Mianroodi J, Siboni N, Goyal P, Svendsen B, Benner P, Raabe D (2023) An artificial neural network for surrogate

modeling of stress fields in viscoplastic polycrystalline materials. npj Comput Mater. https://doi.org/10.1038/s41524-023-00991-z

86. Joy EJ, Menon AS, Biju N (2018) Implementation of kriging surrogate models for delamination detection in composite structures. Adv Compos Lett 27(6):096369351802700604. https://doi.org/10.1177/096369351802700604
87. Wang Y, Lin J, Liu J, Sun G, Pang T (2022) Surrogate-assisted differential evolution with region division for expensive optimization problems with discontinuous responses. IEEE Trans Evol Comput 26(4):780–792. https://doi.org/10.1109/TEVC.2021.3117990
88. Ren C, Aoues Y, Lemosse D, Souza De Cursi E (2022) Ensemble of surrogates combining kriging and artificial neural networks for reliability analysis with local goodness measurement. Struct Saf 96:102186. https://doi.org/10.1016/j.strusafe.2022.102186
89. Santos LF, Costa CBB, Caballero JA, Ravagnani MASS (2023) Multi-objective simulation-optimization via kriging surrogate models applied to natural gas liquefaction process design. Energy 262:125271. https://doi.org/10.1016/j.energy.2022.125271
90. Chen C, Liu J, Xu P (2022) Comparison of parallel infill sampling criteria based on kriging surrogate model. Sci Rep. https://doi.org/10.1038/s41598-021-04553-5
91. Amato F pyGRNN. (2021) https://github.com/federhub/pyGRNN
92. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830