Research on Evolutionary Large-scale Optimization Algorithm

進化的大規模最適化アルゴリズムに関する研究

Rui Zhong

A Dissertation

submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Information Science and Technology

in the field of

Computer Science and Information Technology

at the

Graduate School of Information Science and Technology

Hokkaido University, Japan

2024

Abstract

Evolutionary computation (EC), a sub-field of artificial intelligence (AI) and soft computing, constitutes a family of global and stochastic optimization techniques inspired by natural phenomena and organism behaviors. Thanks to its superior characteristics of robustness, scalability, gradients-free, and easy implementation, it has found wide application in academic research and real-world scenarios. However, as the scale of the optimization problem increases, an inevitable issue known as the curse of dimensionality arises. In this phenomenon, the search domain of the problem also expands exponentially and simultaneously. This phenomenon significantly deteriorates the performance of EC techniques and exacerbates the complexity and difficulty of problems. When the scale of the problem reaches a specific dimension (e.g., 1000-D), this type of problem is referred to as a large-scale optimization problem (LSOP).

This dissertation proposes two directions to deal with LSOPs with EC techniques: (1). Introducing the cooperative coevolution (CC) framework and (2). Enhancing the performance of EC techniques. Motivated by divide-and-conquer, the CC framework decomposes the LSOP into multiple sub-components and optimizes them independently and alternatively, which can alleviate the negative influence of the curse of dimensionality.

However, few studies focus on the robustness of decomposition methods such as real-coded linkage identification by nonlinearity check (LINC-R). These methods cannot detect separability between decision variables in noisy environments. To overcome this problem, we propose linkage measurement minimization (LMM). Although it is difficult for conventional LINC-R and differential grouping (DG) based methods to identify the interactions through the absolute strength of interaction with thresholds, we hypothesize the strength between originally separable and non-separable decision variables is distinct in noisy environments, and this relative strength can help algorithm identify the separability and form the sub-components automatically. Additionally, most studies focus on identifying addi-

tive separability between decision variables and improving decomposition accuracy while reducing computational complexity, and other kinds of separability are rarely investigated, such as multiplicative separability and composite separability. Therefore, we propose Efficient Dual Differential Grouping (EDDG), which incorporates the principle of identifying multiplicative interactions into the framework of Efficient Recursive Differential Grouping (ERDG). Our proposed EDDG is expected to detect both additive and multiplicative interactions simultaneously while consuming minimal computational resources.

Another direction to achieve the objective of this dissertation is to enhance the performance of optimizers. Furthermore, the EC technique can mainly be divided into meta-heuristic algorithm (MA) and hyper-heuristic algorithm (HHA). For MAs, we pay attention to the two MAs: bio-inspired vegetation evolution (VEGE) and physics-based RIME optimization. The dynamic maturity strategy and the diverse mutation strategy are introduced to VEGE, while the Latin hypercube sampling and embedded distance-based selection are employed in the RIME. Our proposed improved versions of VEGE and RIME are significantly better than the original version and competitive with the state-of-the-art MAs. For HHAs, we propose a novel evolutionary multi-mode slime mold optimization (EMSMO), where the low-level heuristics (LLHs) are inspired by the foraging behaviors of the slime mold, and the high-level component adopts the improvement-based selection function. Moreover, we introduce the surrogate-assisted estimation operator to the hyper-heuristic framework and propose a surrogate ensemble-assisted hyper-heuristic algorithm (SEA-HHA). The experimental results and statistical analyses confirm the efficiency and effectiveness of our proposals.

Acknowledgments

I am deeply grateful to the numerous individuals and institutions that have supported me throughout my doctoral program, enabling the completion of this dissertation.

First and foremost, I extend my heartfelt appreciation to my supervisor Professor Masaharu Munetomo, whose guidance, expertise, and unwavering support have been invaluable. Their mentorship has not only shaped the direction of this research but also broadened my understanding of the subject matter.

My sincere thanks go to cooperators Assistant Professor Jun Yu from Niigata University, Specially Appointed Professor Chao Zhang from the University of Toyama, and Emeritus Professor Hideyuki Takagi who generously contributed their time and insights, making this study possible. Additionally, I am indebted to Enzhi Zhang, Yang Cao, Yuefeng Xu, Fei Peng, and Shilong Zhang for their collaborative spirit, which has enhanced the depth and breadth of my research.

I am thankful to my colleagues and friends who have provided continuous support, engaging discussions, and moments of respite during the challenging phases of this journey. Your encouragement and camaraderie have been a constant source of inspiration.

My appreciation extends to Hokkaido University and its departments for providing the necessary resources and environment conducive to research. The assistance of the librarians, research assistants, and administrative staff has been indispensable.

Lastly, I am indebted to my family and loved ones for their unwavering belief in me. Your encouragement, patience, and sacrifices have been my foundation throughout this endeavor.

In conclusion, this dissertation would not have been possible without the collective efforts and contributions of these remarkable individuals and entities. While their names may not be mentioned here, their impact on my academic and personal growth is immeasurable.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the realm of optimization, the quest for efficiency and effectiveness of advanced optimizers remains a perpetual pursuit, particularly in the face of increasingly complex and large-scale problems across diverse domains. Evolutionary algorithms (EAs) have emerged as a promising paradigm for addressing such challenges. EA draws inspiration from the principles of nature (e.g. the movement of organisms in a bird flock or fish school) to iteratively refine candidate solutions and approach the optimal solutions. However, as the dimension of the problem increases, an unavoidable phenomenon is the curse of dimensionality, where the search space is amplified exponentially as the dimension of the problem increases linearly. This phenomenon not only significantly deteriorates the performance of EC techniques but also amplifies the complexity and arduousness of problem-solving endeavors. As the scale of the problem continuously increases and reaches a specific dimensionality threshold (e.g., 1000 dimensions), the challenges become formidable and are termed large-scale optimization problems (LSOPs). In LSOPs, the numerous decision variables extremely increase the computational burden, rendering traditional optimization approaches unaffordable and underscoring the pressing need for innovative methodologies tailored to solve this kind of problem.

Therefore, the objective of this dissertation is to solve LSOPs through two directions:(1).

Using the CC framework and (2). Enhancing the performance of the EC techniques. For the objective (1), our proposals are listed as follows:

- We propose linkage measurement minimization (LMM) for decomposing LSOPs in noisy environments.
- We propose efficient dual differential grouping (EDDG) to detect the additive and the multiplicative separability simultaneously with an affordable computational budget.

For the objective (2), our proposals are listed as follows:

- We propose an improved vegetation evolution (VEGE) with a dynamic maturity strategy and diverse mutation strategy for continuous optimization.
- We propose a strengthened RIME algorithm (SRIME) with the Latin hypercube sampling and embedded distance-based selection for continuous optimization.
- We propose a novel HHA named evolutionary multi-mode slime mold optimization. The LLHs are inspired by the foraging behaviors of the slime mold, and the high-level component adopts the improvement-based selection function.
- We introduce the surrogate-assisted estimation operator to the hyper-heuristic framework and propose a novel HHA named surrogate ensemble-assisted hyper-heuristic algorithm (SEA-HHA)

The remainder of this dissertation is organized as follows. Chapter 2 introduces the background of our research. Chapter 3 introduces the proposed LMM. Chapter 4 introduces the proposed surrogate ensemble assisted DE (SEADE) and EDDG to deal with LSEOPs. Chapter 5 introduces the proposed vegetation evolution with a dynamic maturity strategy and diverse mutation strategy. Chapter 6 introduces the proposed SRIME algorithm. Chapters 7 and 8 introduce the proposed hyper-heuristics algorithms inspired by the foraging behaviors of slime mold and with surrogate assistance, respectively. Finally, Chapter 9 concludes this dissertation. Figure 1.1 demonstrates the skeleton of this dissertation.

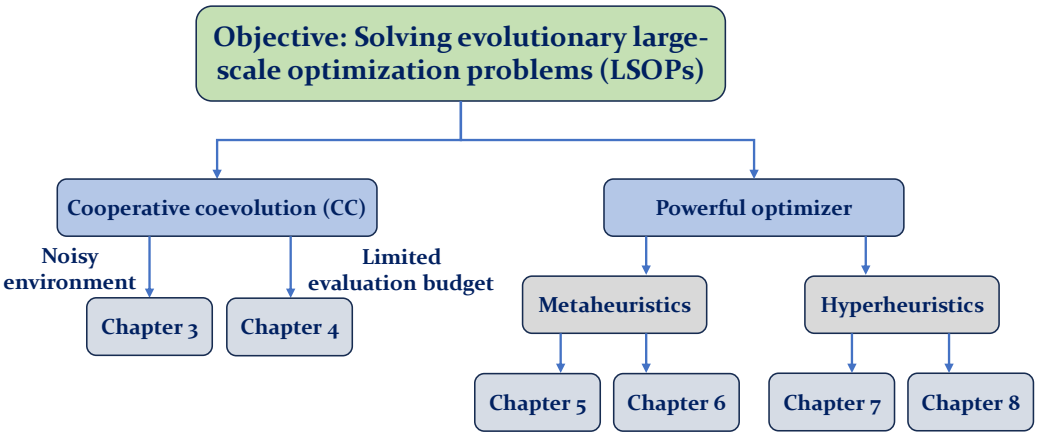


Figure 1.1: The skeleton of this dissertation.

Chapter 2

Background

2.1 Large-scale Optimization Problems (LSOPs)

Without loss of generality, a minimization problem without constraints can be defined as follows:

$$\begin{aligned} &\min f(\mathrm{X}) \\ &s.t. : \mathrm{X} \in \mathbb{R}^n \end{aligned} \tag{2.1}$$

Where $\mathrm{X} = \{x_1, x_2, ..., x_n\}$ is an n-dimensional decision vector, and each x_i $(i \in [1, n])$ is a decision variable, and $f(\mathrm{X})$ is the objective function needed to be minimized. Although there is no clear boundary between small-scale, median-scale, and large-scale problems, the academic community commonly considers when the scale of binary and integer problems reaches 1 billion,[6–8] and the dimension of the real-coded problem reaches 1000,[9] which can be regarded as LSOPs. In our work, the LSOP is a special case of black-box optimization, which denotes the information of the fitness landscape such as the gradient and the topology is difficult to obtain. A visual demonstration between black-box optimization and white-box optimization is provided in Figure 2.1.

In black-box optimization problems, the inherent structure remains concealed, while the internal structure of white-box optimization problems such as linkage information, gra-

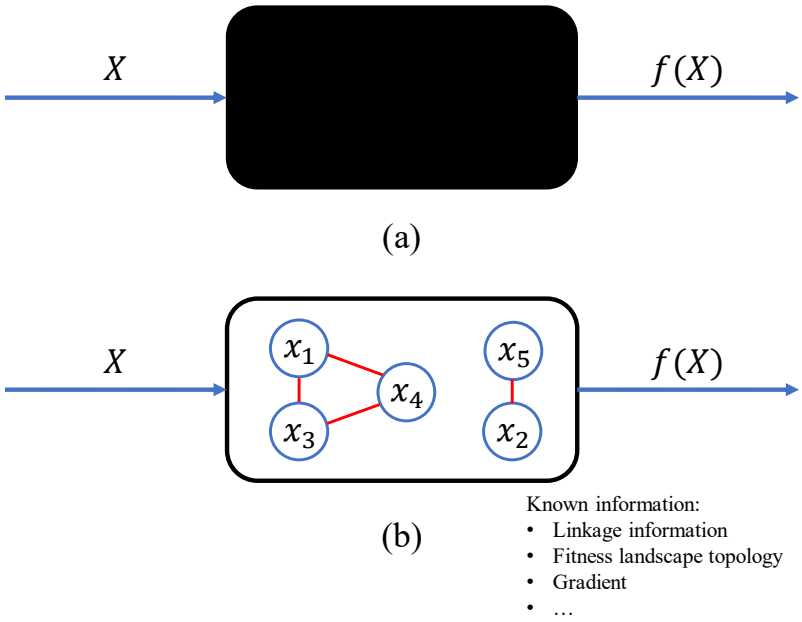


Figure 2.1: (a). The black-box optimization problem. (b). The white-box optimization problem.

dient details, topology, and feasible regions are available. These types of knowledge can support us in designing problem-specific search operators. Therefore, addressing black-box optimization presents a challenging task.

2.2 Evolutionary Computation (EC)

A general flowchart of EC techniques consists of five steps, which can be described as follows:

(1). Initialize the population and parameters.

(2). Determine the position of the offspring individual with specific search operators.

(3). Evaluate the offspring individuals.

(4). Select the surviving individuals by selection mechanism.

(5). Repeat step (2) to step (4) until the global optimum is found or the computational resources are exhausted.

Among these steps, the design of specific search operators in step (2) is the kernel in the EC technique, which can be realized by imitating the social behaviors of animals[10–12] or simulating the laws of science.[4,13,14]

2.3 Hyper-heuristic algorithm

Hyper-heuristic (HH) algorithm is an optimization technique that operates at a higher level, focusing on the selection or generation of LLHs to solve computational problems, rather than directly solving the problems themselves. In essence, the HH algorithm aims to find or generate sequences of heuristics that demonstrate superior robustness and scalability across a range of problems. Figure 1.1 illustrates a representative architecture of the HH framework.

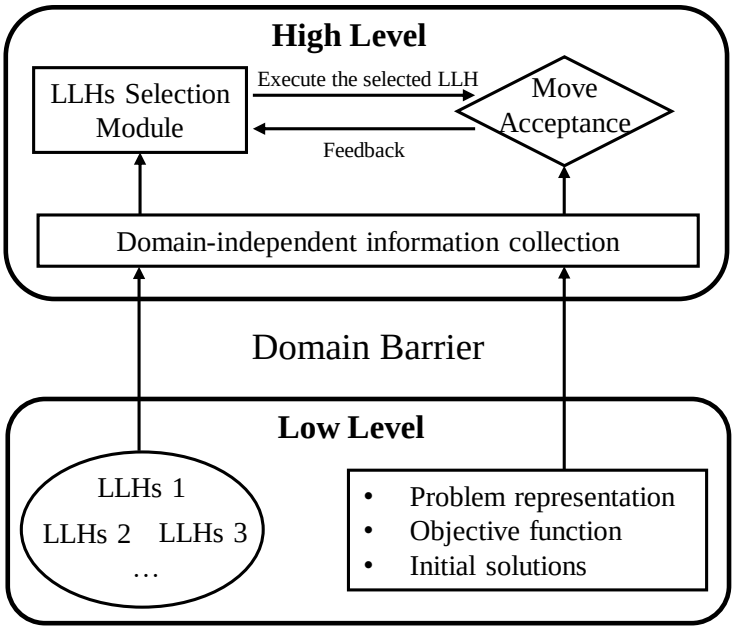


Figure 2.2: A representative architecture of the HH framework.[1]

A complete HH algorithm is comprised of two main components: the high-level component and the low-level component. The high-level component encompasses overarching strategies or mechanisms that guide the selection or generation of LLHs. It provides a

methodology for decision-making and adaptation throughout the optimization process. On the other hand, the low-level component comprises the actual LLHs that are applied to problem instances to construct solutions. These LLHs serve as the fundamental building blocks used by the HH algorithm to explore and exploit the problem space.

2.4 Cooperative Coevolution (CC)

Inspired by the divide-and-conquer strategy, the cooperative coevolution (CC)[15] framework decomposes LSOPs into multiple sub-components and optimizes them alternately, which is an efficient and flexible approach to dealing with LSOPs. Generally, a standard 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[16] will participate in the evaluation of sub-solutions.

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

In summary, the pseudocode of the CC framework is shown in Figure 2.3.

2.5 Separability between decision variables

Three types of separability commonly exist between decision variables: additively separable, multiplicatively separable, and compositely separable. The relationship between various categories of separability is demonstrated in Figure 2.4.

Additively separable: Additive separability is the most well-studied category of separability in recent decades. If an n-D function can be divided into $m(m > 1)$ additively

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
```

Figure 2.3: The pseudocode of the CC framework.

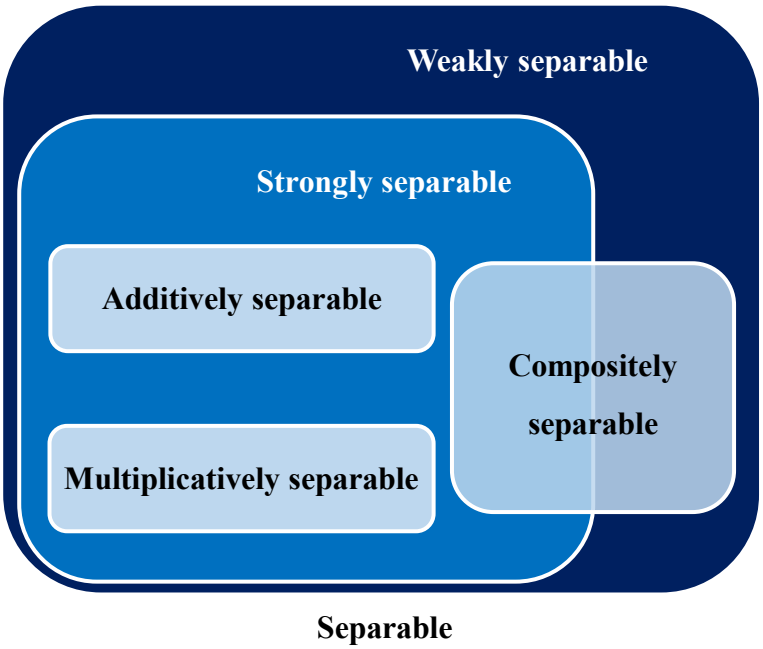


Figure 2.4: The relationship between various categories of separability.[2]

independent parts such that

$$\min f(x) = \min \sum_{i=1}^{m} f_i(x_i) \tag{2.2}$$

then this function is additively separable. From mathematical perspective, if two decision variables x_i and x_j satisfy $\frac{\partial^2 f}{\partial x_i \partial x_j} = 0$, then x_i and x_j are additively separable. Therefore, the finite differences principle[17] can be adopted in DG to identify additive separability. An example to illustrate additively separable is $f(x) = x_1^2 + x_2^2,\ x_1, x_2 \in [-5, 5]$. Figure 2.5 provides the contour graph of this function. Red lines denote when x_2 is fixed, the fitness changes as x_1 changes. The black points are the optima when x_2 is fixed to -2, 0, and 2. We can simply find that the change of x_2 will not affect the optimal position of x_1, thus, x_1 and x_2 can be optimized independently.

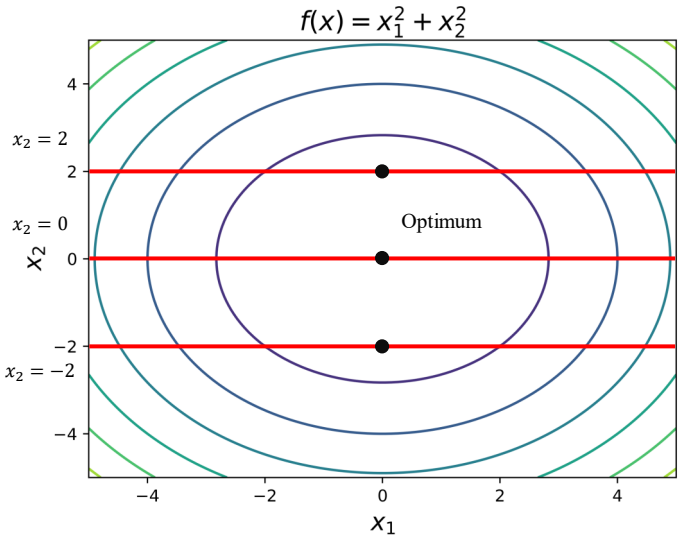


Figure 2.5: The contour graph of the function $f(x) = x_1^2 + x_2^2,\ x_1, x_2 \in [-5, 5]$.

Multiplicatively separable: Multiplicative separability has a similar definition to the additive separability that, if an n-D function can be divided into $m(m > 1)$ multiplicatively

independent parts such that

$$\begin{aligned} \min f(x) &= \min \prod_{i=1}^{m} f_i(x_i) \\ s.t.\ \forall i \in [1, m], f_i(x_i) &\leq 0 \ or \ f_i(x_i) \geq 0 \end{aligned} \tag{2.3}$$

then this function is multiplicatively separable. An example is $f(x) = (x_1^2 + 1) \cdot (x_2^2 + 1),\ x_1, x_2 \in [-5, 5]$ and corresponding contour graph is shown in Figure 2.6. When x_2 is

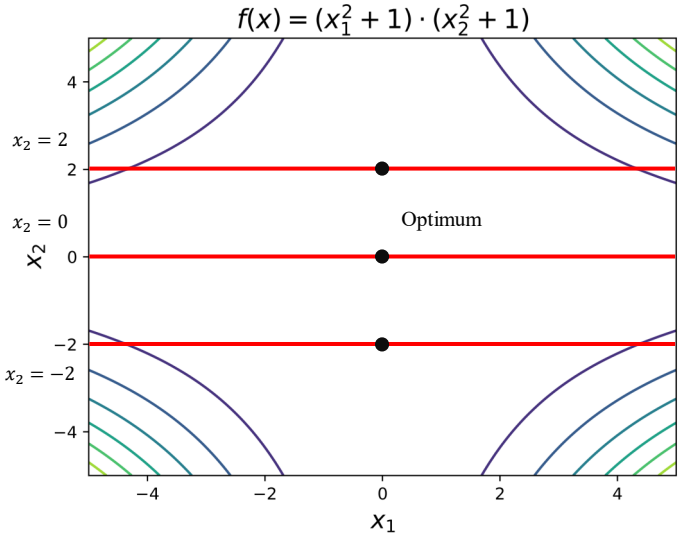


Figure 2.6: The contour graph of the function $f(x) = (x_1^2 + 1) \cdot (x_2^2 + 1),\ x_1, x_2 \in [-5, 5]$.

fixed to -2, 0, and 2, this function can be simplified to

$$\begin{aligned} f(x) &= 5(x_1^2 + 1),\ x_2 = -2 \ or \ 2 \\ f(x) &= x_1^2 + 1,\ x_2 = 0 \end{aligned} \tag{2.4}$$

and the optimal position of $x_1 = 0$, which is denoted in Figure 2.6. In this situation, x_1 and x_2 are multiplicatively separable.

Compositely separable: If $f(x)$ is a composite function (i.e., $f(x) = U(g(x))$ and (1). the inner function $g(x)$ is a separable function and (2). the outer function $U(x)$ is monotonically increasing in its feasible regions are satisfied simultaneously, then the function $f(x)$

is compositely separable.[18] An example is $f(x) = \log_{10}(x_1^2 + x_2^2 + 1),\ x_1, x_2 \in [-5, 5]$, which is visualized in Figure 2.7. We notice that the fitness landscape topology of this example is similar to the additively separable example $f(x) = x_1^2 + x_2^2$, and even if the x_2 changes, the optimal position of x_1 is irrelevant to the position of x_2.

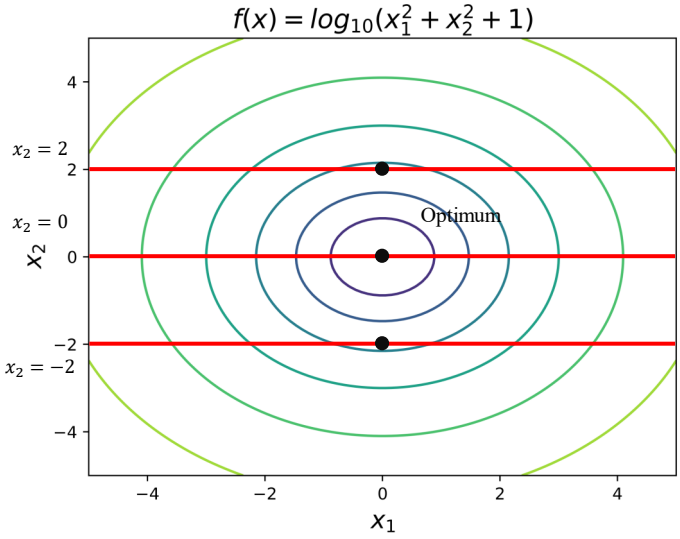


Figure 2.7: The contour graph of the function $f(x) = \log_{10}(x_1^2 + x_2^2 + 1),\ x_1, x_2 \in [-5, 5]$.

2.6 Review of decomposition methods

Since Potter et al.[15] first adopted the CC framework to deal with LSOPs, many cooperative coevolutionary algorithms (CCEAs) have been proposed and successfully applied to solving various complex optimization problems. In this section, we review the decomposition methods in CCEAs, which can be classified into three categories: static decomposition, random decomposition, and learning-based decomposition.

Static decomposition: CCGA,[15] 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$_K$[16]

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[19] 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[20] 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[21] 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 of capturing the interactions in non-separable functions. Therefore, DECC-II[19] was proposed to randomly select the scale of sub-components s from a predefined range in the coevolution cycle. MLCC[22] provides a scale candidate $[s_1, s_2, ..., s_n]$ and chooses the scale s_i by the recent improvement of the context vector. However, the determination of the proper scale of the subcomponent for a specific optimizer is a difficult task, and MLSoft[23] regarded this issue as an optimization problem and the widely used reinforcement learning technique was applied to select the suitable scale of subcomponents dynamically. Moreover, Cooperative coevolution with adaptive subcomponents (CCAS)[24,25] further investigated the effects of both the size of the

population and the size of subcomponents on the performance of the CC framework and proposed an adaptive CC mechanism, which allows parts of the available computational budget is spent for adapting both the scale of subcomponents and the number of evolved individuals during the optimization process.

Thanks to the superior characteristics of easy implementation, computationally cheap, and scalability of the random decomposition technique, it is still a popular technique in recent advances in dealing with LSEOPs: Sun et al.[26] 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.[27] 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.[28] 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.

Learning-based approaches: In the early development of decomposition methods, researchers attempted to capture the interactions by probability until the appearance of Linkage Identification (LI). Based on the different mechanisms of interaction identification, LI includes two kinds: Linkage Identification by Non-linearity Check (LINC)[29] and Linkage Identification by Monotonicity Detection (LIMD).[30]

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

variables. Mathematically,

$$\begin{gathered} \text{If } \frac{\partial f^2(x)}{\partial x_i \partial x_j} = 0 \\ \text{Then } x_i \; and \; x_j \; are \; separable \end{gathered} \tag{2.5}$$

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:

$$\begin{gathered} \forall s \in Pop: \\ \Delta_i = f(s_i) - f(s), \Delta_j = f(s_{ij}) - f(s_j) \\ \text{If } |\Delta_i - \Delta_j| < \varepsilon \\ \text{Then } x_i \; and \; x_j \; are \; separable \end{gathered} \tag{2.6}$$

Where ε is an allowable error. Figure 2.8 demonstrates the mechanism of LINC in the 2-D search space.

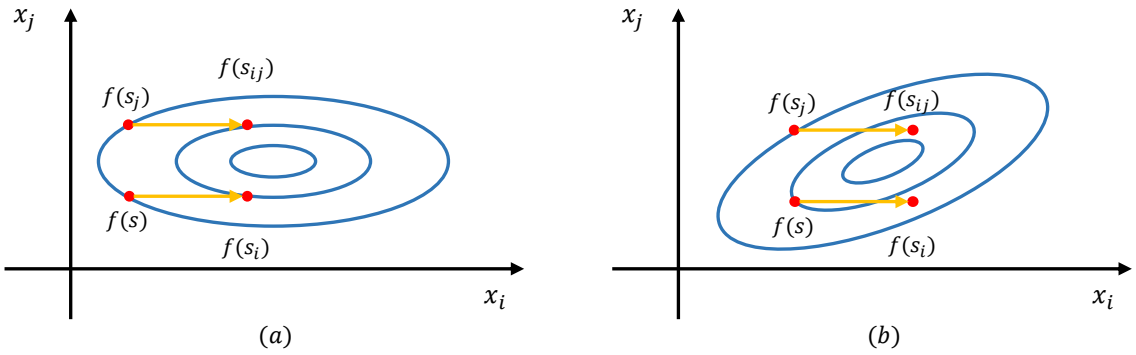


Figure 2.8: The visualized demonstration of the LINC in the 2-D domain. (a). LINC in separable decision variables. (b). LINC in non-separable decision variables.

Once detection for two decision variables needs 4 fitness evaluations (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 com-

pletely unacceptable for LSOPs. Differential Grouping (DG)[17] 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.[17] Global DG (GDG)[31] introduces the pairwise interactions to improve the decomposition accuracy. Graph DG[32,33] utilized the data structure of the graph to support the interaction identification between decision variables and realized the optimal decomposition in the whole CEC2010 benchmark functions. DG2,[34] 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. (2.6) 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 the 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[35] 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

$$f(\mathbf{x}^* + l_1\mathbf{u}_1 + l_2\mathbf{u}_2) - f(\mathbf{x}^* + l_2\mathbf{u}_2) \neq f(\mathbf{x}^* + l_1\mathbf{u}_1) - f(\mathbf{x}^*) \tag{2.7}$$

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. (2.7), the element in X_1 is determined as a separable variable, otherwise, X_2 is separated into two mutually exclusive subsets with equal size and recursively execute the identification. Figure 2.9 shows a demonstration of RDG. This binary search fashion strategy extremely reduces the

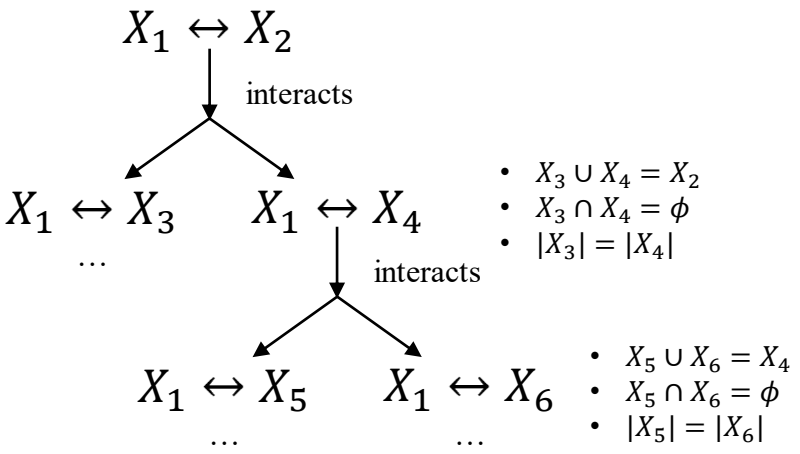


Figure 2.9: The decomposition process of RDG.

time complexity for decomposition from $O(N^2)$ to $O(N\log(N))$. Furthermore, Efficient RDG (ERDG)[36] notices the redundant interaction examinations exist in the RDG, which can be avoided by historical information. From Figure 2.9, variable subset X_3 and X_4 are mutually exclusive and $X_3 \cup X_4 = X_2$, thus, Eq. (2.7) can be transformed to Eq. (2.8)

$$f(\mathbf{x}^* + l_1\mathbf{u}_1 + l_2(\mathbf{u}_3 + \mathbf{u}_4)) - f(\mathbf{x}^* + l_2(\mathbf{u}_3 + \mathbf{u}_4)) \neq f(\mathbf{x}^* + l_1\mathbf{u}_1) - f(\mathbf{x}^*) \tag{2.8}$$

Here, we define

$$\begin{aligned} \Delta_1 =& f(\mathbf{x}^* + l_1\mathbf{u}_1 + l_2(\mathbf{u}_3 + \mathbf{u}_4)) - f(\mathbf{x}^* + l_2(\mathbf{u}_3 + \mathbf{u}_4)) \\ \Delta_2 =& f(\mathbf{x}^* + l_1\mathbf{u}_1) - f(\mathbf{x}^*) \end{aligned} \tag{2.9}$$

If X_1 interacts with X_2, X_2 will be separated into X_3 and X_4, and the interaction identifi-

cation of $X_1 \leftrightarrow X_3$ can be computed as

$$\begin{aligned} \Delta_1^{'} =& f(\mathbf{x}^* + l_1\mathbf{u}_1 + l_2\mathbf{u}_3) - f(\mathbf{x}^* + l_2\mathbf{u}_3) \\ \Delta_2^{'} =& f(\mathbf{x}^* + l_1\mathbf{u}_1) - f(\mathbf{x}^*) = \Delta_2 \end{aligned} \tag{2.10}$$

Similarly, Eq. (2.11) can be applied to identify the interaction of $X_1 \leftrightarrow X_4$.

$$\begin{aligned} \Delta_1^{''} =& f(\mathbf{x}^* + l_1\mathbf{u}_1 + l_2\mathbf{u}_4) - f(\mathbf{x}^* + l_2\mathbf{u}_4) \\ \Delta_2^{''} =& \Delta_2 \end{aligned} \tag{2.11}$$

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. (2.11) can be avoided to save the FEs. Although the time 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 large-scale global optimization (LSGO) benchmark functions.

Similarly, merged DG (MDG)[37] also used a binary-tree-based iterative merging strategy and further saves 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)[38] 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

$$\begin{aligned} f(\mathbf{x}) &= \ln g(\mathbf{x}) \\ &= \ln \prod_{i=1}^{k} g_i(\mathbf{x}_i) \\ &= \sum_{i=1}^{k} \ln g_i(\mathbf{x}_i), 1 < k \leq D \end{aligned} \tag{2.12}$$

Therefore, the identification condition of DDG to detect the multiplicative separability can be formulated by Eq. (2.13)

$$\begin{gathered} \text{If } f(s), f(s_i), f(s_j), f(s_{ij}) \; are \; positive \; and \\ |\ln(f(s_{ij}) - \ln f(s_j)) - \ln(f(s_i) - \ln f(s))| < \varepsilon \\ \text{Then } x_i \; and \; x_j \; are \; multiplicatively \; separable \end{gathered} \tag{2.13}$$

Besides, GSG,[18] 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.

2.7 Review of meta-heuristics

Since more and more complex and high-dimensional optimization problems occur in academic research and industrial scenarios, traditional techniques such as linear/nonlinear programming,[39] Newton's method,[40] conjugate gradient method,[41] and gradient descent method[42] are not suitable to deal with these complex optimization problems since these methods may require that the feasible domain is convex[43] and the objective function is continuously differentiable.[44] Therefore, conventional optimization methods face a severe challenge and it is urgent to develop an efficient and robust optimization algorithm.[45,46] In

the meantime, the metaheuristic algorithm (MA) provides a feasible methodology to address complex optimization problems. MA is a class of stochastic optimization techniques, which are commonly inspired by natural phenomena or organism behaviors and iteratively search for acceptable solutions.[47] Here, we list some representative MAs and corresponding inspirations in Table 2.1.

Table 2.1: Representative MAs in literature.

Category	Algorithm	Inspiration
evolution-based	genetic algorithm (GA)[48]	evolutionary concepts
	differential evolution (DE)[49]	Darwin's theory of evolution
	genetic programming (GP)[50]	biological evolution
	evolutionary programming (EP)[51]	finite state machine
swarm intelligence	particle swarm optimization (PSO)[52]	social behaviors of bird flock
	salp swarm algorithm (SSA)[53]	the navigating and foraging behaviors of salp
	yellow saddle goatfish algorithm (YSGA)[54]	collaborative behaviors of yellow saddle goatfish
	sailfish optimizer (SFO)[55]	a group of hunting sailfish
	bald eagle search (BES)[56]	the hunting strategies of bald eagles
	marine predators algorithm (MPA)[57]	the widespread foraging strategy of ocean predators
	black widow optimization (BWO)[58]	the unique mating behavior of black widow spiders
	red deer algorithm (RDA)[59]	the behavior of Scottish red deer
	capuchin search algorithm (CapSA)[60]	dynamic behavior of capuchin monkeys
	tuna swarm optimization (TSO)[61]	the cooperative foraging behavior of tuna swarm
	archerfish hunting optimizer (AHO)[11]	the shooting and jumping behaviors of the archerfish
	snake optimizer (SO)[62]	special mating behavior of snakes
	vegetation evolution (VEGE)[63]	growth mechanism of tomatoes
	fire hawk optimizer (FHO)[64]	the foraging behavior of fire hawks
	honey badger algorithm (HBA)[65]	intelligent foraging behavior of honey badgers
	sand cat swarm optimization (SCSO)[66]	the sand cat behavior that tries to survive in nature
	coati optimization algorithm (COA)[67]	natural behaviors of coatis
	crayfish optimization algorithm (COA)[68]	behaviors of crayfish
human-based	brain storm optimization (BSO)[69]	the human brainstorming process
	teaching learning based optimization (TLBO)[70]	the influence of a teacher on learners
	adolescent identity search algorithm (AISA)[71]	the identity development/search of adolescents
	political optimizer (PO)[72]	the multi-phased process of politics
	group teaching optimization algorithm (GTOA)[73]	the group teaching mechanism
	search and rescue optimization (SARO)[74]	human behaviors during search and rescue operations
	cooperation search algorithm (CSA)[75]	team cooperation behaviors in modern enterprise
	child drawing development optimization (CDDO)[76]	child's learning behavior and cognitive development
	human conception optimizer (HCO)[77]	biological principles of the human conception process
	sewing training-based optimization (STBO)[78]	process of sewing to beginner tailors
	mountaineering team-based optimization (MTBO)[79]	cooperative human behaviors in mountaineering
	mother optimization algorithm (MOA)[80]	the interaction between a mother and her children
law-based	nuclear reaction optimization (NRO)[81]	nuclear reaction process
	Lévy flight distribution (LFD)[82]	Lévy flight random walk
	gradient-based optimizer (GBO)[83]	gradient-based Newton's search method
	momentum search algorithm (MSA)[84]	Newton's laws
	material generation algorithm (MGA)[85]	chemical compounds and chemical reactions
	RUN[86]	Runge Kutta method
	Pareto-like sequential sampling (PSS)[87]	Pareto's principle
	INFO[88]	weighted mean of the vector
	special relativity search (SRS)[89]	the interaction of particles in an electromagnetic field
	RIME[4]	the soft-rime and hard-rime growth process
	snow ablation optimizer (SAO)[90]	the sublimation and melting behavior of snow
	optical microscope algorithm (OMA)[91]	microscope magnification
	Chernobyl disaster optimizer (CDO)[92]	the nuclear reactor core explosion of Chernobyl

Chapter 3

Linkage identification in noisy environments

This chapter introduces the linkage identification in noisy environments[1]. Section 3.1 presents the motivation of this research, Section 3.2 details our proposal: linkage measurement minimization (LMM), Section 3.3 provides the mathematical support for our proposal, Section 3.4 describes the experiments on the CEC2013 LSGO suite, and Section 3.5 discusses the direction of our research in the future. Finally, Section 3.6 concludes this chapter.

3.1 Motivation

Optimization problems often encounter noise, particularly in conjunction with large-scale attributes, leading to a significant increase in problem complexity. Numerous algorithms have been developed to address these challenges. These include custom optimization operators designed to adapt to large-scale attributes, the creation of surrogate models, and

[1]This chapter was previously published as Zhong, R., Zhang, E. & Munetomo, M. Cooperative coevolutionary differential evolution with linkage measurement minimization for large-scale optimization problems in noisy environments. Complex Intell. Syst. 9, 4439–4456 (2023) https://doi.org/10.1007/s40747-022-00957-6.[93]

problem decomposition strategies, known collectively as the CC framework. This chapter introduces our approach, which leverages the CC framework to tackle LSOPs in noisy environments.

The CC framework draws inspiration from the divide and conquer strategy, which has demonstrated remarkable success in addressing large-scale continuous,[94] combinatorial,[95] and constrained[96] problems. Effective decomposition of LSOPs is pivotal for the successful implementation of the CC framework. Numerous studies[97,98] highlight the sensitivity of the CC framework to various problem decomposition strategies. Pioneered by LINC-R,[30] many decomposition methods have since been proposed. Differential Grouping (DG)[17] was the first to extend LINC-R's identical mechanism to address 1000-dimensional problems. Subsequently, Extend DG (XDG)[99] enhanced the capabilities of DG to handle overlaps more effectively. DG2[34] addressed the computational cost challenges of DG by leveraging the transmissibility of separability. Global DG (GDG)[31] treated the variable interactions matrix as a graph's adjacency matrix, utilizing depth-first or breadth-first search to identify interactions and form sub-problems. RDG[35] further optimized computational efficiency by examining interactions between sets of variables rather than individual variables, forming sub-problems recursively. ERDG[36] leveraged historical information on interaction identification to reduce redundant examinations, enhancing computational efficiency compared to RDG.

While these decomposition methods are regarded as high-accuracy approaches, they rely on determining the difference between fitness values and a predefined threshold to detect interactions. However, they exhibit extreme sensitivity to the fidelity of observed objective values and may fail to detect interactions in noisy environments. Here, we take LINC-R as an example and illustrate the reason mathematically.

We first present a demonstration of LINC-R works in noisy environments in Figure 3.1. From the intuition, LINC-R cannot detect the separability in noisy environments since the fitness landscape will be affected by the noise though Figure 3.1 (b). The complete

mathematical analysis is explained as follows:

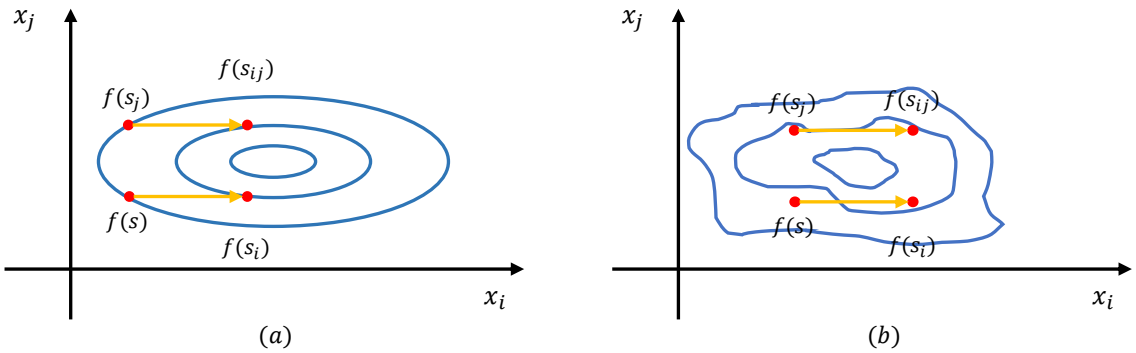


Figure 3.1: A demonstration of LINC-R works in noisy environments. (a). LINC-R works on noiseless separable decision variables. (b). LINC-R works on separable decision variables in noisy environments.

Additive noise and multiplicative noise represent two distinct types of noise commonly encountered in optimization problems. Additive noise is typically independent of the underlying fitness landscape, allowing for parameter adjustments to mitigate its effects in the decomposition process, albeit with considerable difficulty.[100] On the other hand, multiplicative noise is intricately linked to the fitness landscape, potentially amplifying the noise as fitness values increase. Consequently, addressing multiplicative noise poses greater challenges during the decomposition stage compared to additive noise. Therefore, the following explanation focuses on LINC-R in multiplicative noisy environments, as expressed in Eq. (3.1).

$$\begin{aligned} &\Delta_1^N = f^N(s_i) - f^N(s) = f(s_i)(1+\beta_1) - f(s)(1+\beta_2) \\ &\Delta_2^N = f^N(s_{ij}) - f^N(s_j) = f(s_{ij})(1+\beta_3) - f(s_j)(1+\beta_4) \\ &if\ |\Delta_2^N - \Delta_1^N| > \varepsilon \\ &then\ x_i\ and\ x_j\ are\ nonseparable \end{aligned} \tag{3.1}$$

where β_i is the noise following a Gaussian distribution $N(0, \sigma^2)$. Furthermore, the criterion of LINC-R can be simplified to Eq. (3.2).

$$|\Delta_2^N - \Delta_1^N| = |\Delta_2 - \Delta_1 + f(s_{ij})\beta_3 - f(s_j)\beta_4 - f(s_i)\beta_1 + f(s)\beta_2| \tag{3.2}$$

Here, we define the noise term in Eq. (3.3).

$$\phi_{ij} = f(s_{ij})\beta_3 - f(s_j)\beta_4 - f(s_i)\beta_1 + f(s)\beta_2 \tag{3.3}$$

This term follows a Gaussian distribution in Eq. (3.4).

$$\phi_{ij} \sim N(0, (f^2(s_{ij}) + f^2(s_j) + f^2(s_j) + f^2(s))\sigma^2) \tag{3.4}$$

In noisy environments with multiplicative noise, LINC-R cannot identify that the fitness difference is caused by interaction or noise, and the probability of $\phi_{ij} = 0$ being satisfied is almost 0.[101] In practice, the decomposition methods developed on the LINC-R, such as DG, DG2, RDG, etc. will fail in environments with multiplicative noise. Therefore, grouping methods that detect interactions by perturbation face severe challenges.

3.2 Proposal: MDE-DSCC-LMM

In this section, we will delve into the specifics of our proposed methodology, which comprises two key components: decomposition and optimization. Firstly, in the decomposition phase, we employ our proposed approach called linkage measurement minimization (LMM) to partition the decision variables into sub-problems. Secondly, in the optimization phase, we utilize a modified differential evolution with distance-based selection (MDE-DS) as the fundamental optimizer to optimize these sub-problems. We will now provide a detailed explanation of the procedures involved in both the decomposition and optimization components.

3.2.1 linkage measurement minimization (LMM)

First, we provide the flowchart of our proposal in decomposition: LMM. The flowchart is shown in Figure 3.2.

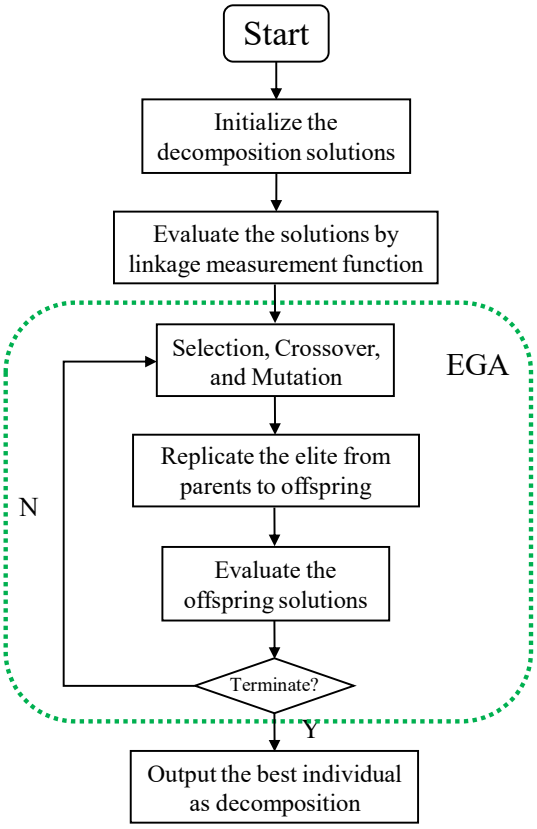


Figure 3.2: The flowchart of decomposition method.

The basic idea is that we regard the decomposition problem as a combinatorial optimization problem and design the LMF based on LINC-R to lead the direction of searching for a better decomposition solution. The specific derivation of LMF is as follows.

The original LINC-R is defined as Eq. (3.5)

$$\begin{aligned} &\exists s \in Pop : \\ &if\ |(f(s_{ij}) - f(s_j)) - (f(s_i) - f(s))| > \varepsilon \\ &then\ x_i\ and\ x_j\ are\ nonseparable \end{aligned} \tag{3.5}$$

where the size of Pop is m, FEs consumed in a pair of variables based on Pop is $4m$, and the interaction between every pair of variables is identified in LINC-R. Thus, in the

$n-$D problem, the necessary FEs is $2mn(n+1)$, which is unaffordable for LSOPs. Many studies [17,102] only detect the interactions by calculating the fitness difference from the lower bound of search space to the upper bound to save the FEs in decomposition, and we adopt the same strategy in our proposal, although it is not so robust and may fail to detect the interactions in trap functions. [103]

We also notice that the original LINC-R can be transformed into the vector addition form. Eq. (3.6) demonstrates this variant LINC-R:

$$\begin{aligned} if\ |(f(s_{ij}) - f(s)) - ((f(s_i) - f(s)) + (f(s_j) - f(s)))| < \varepsilon \\ then\ x_i\ and\ x_j\ are\ separable \end{aligned} \tag{3.6}$$

Figure 3.3 shows how LINC-R and the variant LINC-R work on the separable variables x_i and x_j. Although the form is different, the mechanism of LINC-R and variant LINC-R are identical.

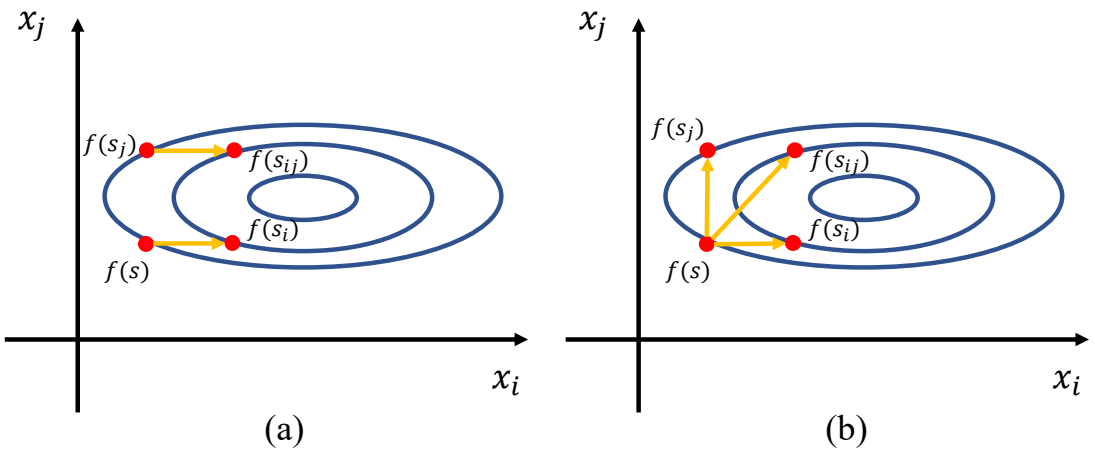


Figure 3.3: (a). LINC-R works on the separable variables. (b). variant LINC-R works on the separable variables.

Based on this interesting finding, we derive LINC-R to $3-$D and higher dimensions. In 3-D space, the schematic diagram is shown in Figure 3.4.

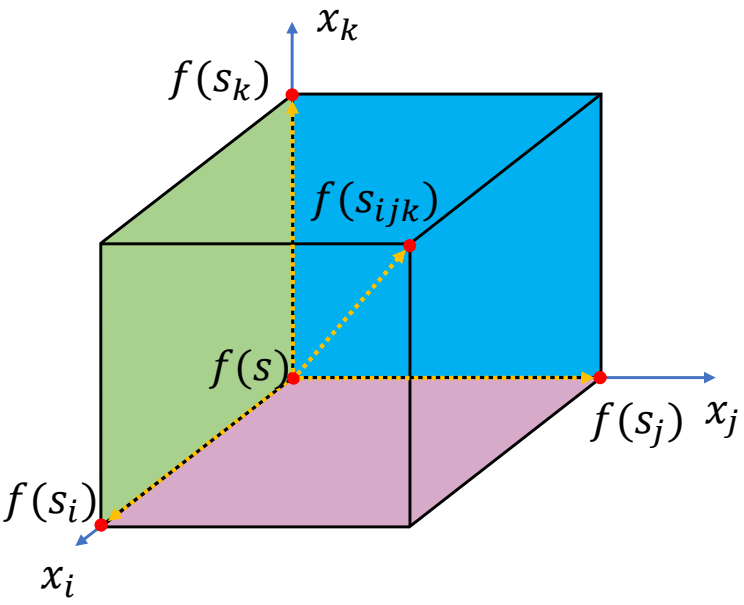


Figure 3.4: The variant LINC-R works on 3-D space.

Here, we define the fitness difference in 3-D:

$$\begin{aligned} \Delta_i &= f(s_i) - f(s) \\ \Delta_j &= f(s_j) - f(s) \\ \Delta_k &= f(s_k) - f(s) \\ \Delta_{ijk} &= f(s_{ijk}) - f(s) \end{aligned} \tag{3.7}$$

When the variant LINC-R is applied simultaneously to determine the interactions between x_i, x_j, and x_k:

$$\begin{aligned} &if\ |\Delta_{ijk} - (\Delta_i + \Delta_j + \Delta_k)| < \varepsilon \\ &then\ x_i, x_j, and\ x_k\ are\ separable \end{aligned} \tag{3.8}$$

Therefore, we can reasonably infer that when the dimension reaches n:

$$\begin{aligned} if\ |\Delta_{1,2,...,n} - (\Delta_1 + \Delta_2 + ... + \Delta_n)| < \varepsilon \\ then\ x_1, x_2, ..., .x_n\ are\ separable \end{aligned} \tag{3.9}$$

However, when Eq. (3.9) is not satisfied, we only know that interactions exist in some variables, but we cannot know in which pairs of variables. Taking 3-D space as an example:

$$\begin{aligned} if\ |\Delta_{ijk} - (\Delta_i + \Delta_j + \Delta_k)| > \varepsilon\ and\ |\Delta_{ijk} - (\Delta_{ij} + \Delta_k)| < \varepsilon \\ then\ x_i, x_j\ are\ non-separabale\ and\ x_k\ is\ separable\ from\ x_i, x_j \end{aligned} \tag{3.10}$$

Therefore, in the n-dimensional space, although it is difficult to detect the interactions between multiple variables through high-dimensional LINC-R directly, we can actively search for the interactions between variables through heuristic algorithms. According to the above description, in the n-dimensional problem, the linkage measurement function (LMF) is defined in Eq. (3.11)

$$\mathrm{LMF}(s) = (\Delta_{1,2,...,n} - \sum_{i,j,...}^{m} \Delta_{i,j,...})^2 \tag{3.11}$$

m is the number of sub-problems. LMF in noisy environments is defined in Eq. (3.12)

$$\mathrm{LMF^N}(s) = (\Delta^N_{1,2,...,n} - \sum_{i,j,...}^{m} \Delta^N_{i,j,...})^2 \tag{3.12}$$

Where $\Delta^N_{1,2,...,n} = f^N(s_{1,2,...,n}) - f^N(s) = f(s_{1,2,...,n})(1+\beta_i) - f(s)(1+\beta_j)$. To optimize the $\mathrm{LMF^N}(s)$, EGA is employed as the basic optimizer. Figure 3.5 demonstrates how to decode from genotype to decomposition. the length of a chromosome is LD, L is the genome length, and D is the dimension of the problem.

We decode the binary chromosome to decimal phenotype level and divide the decision variables into corresponding sub-problems, the decision variables assigned to sub-problem

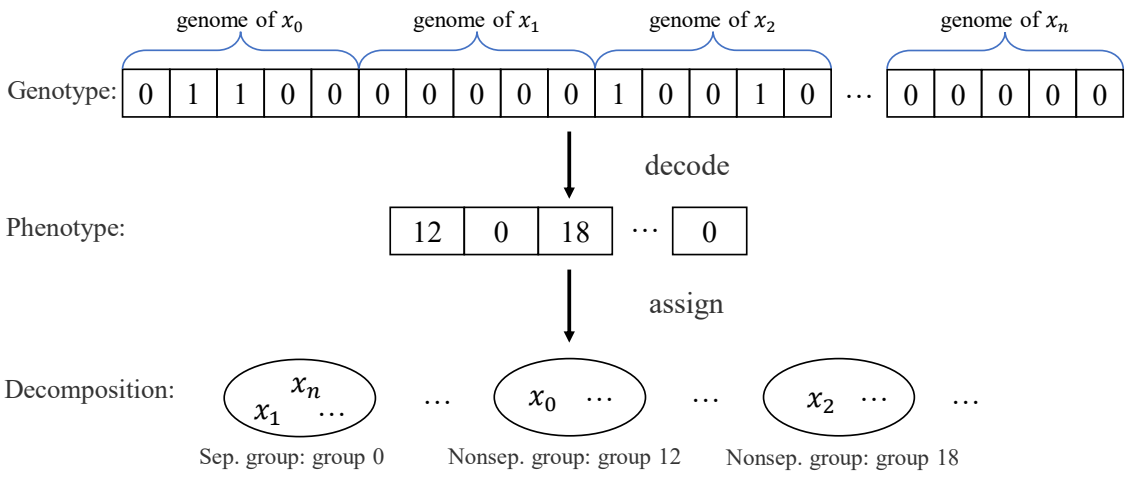


Figure 3.5: A demonstration of decoding from genotype to decomposition

0 are regarded as separable variables. This procedure of optimization guided by LMF is named linkage measurement minimization (LMM), and the pseudocode of the decomposition is shown in Figure 3.6

Algorithm LMM

Require: Population size : s; Genome length : L; Maximum iteration : T
Dimension : D
Ensure: The best decomposition : E

▶ (Decomposition solution initialization)
for $i = 0$ *to* s_1 **do**
  for $j = 0$ *to* D **do**
    $n \leftarrow$ **randint**$(0, 2^{L-1} - 1)$
    $bn \leftarrow$ **binary**(n)
    $P^t_{i,j} \leftarrow bn$
  end for
end for
$F^t \leftarrow$ **evaluate**(P^t)
$E \leftarrow$ **bestIndi**(P^t, F^t) # save the best solution
▶ (Optimization by EGA)
while $t = 0$ *and* $t < T$ **do**
  $P^{t+1} \leftarrow$ **selection**(P^t, F^t, s)
  $P^{t+1} \leftarrow$ **crossover**(P^{t+1})
  $P^{t+1} \leftarrow$ **mutation**(P^{t+1})
  $F^{t+1} \leftarrow$ **evaluate**(P^{t+1})
  $P^{t+1} \leftarrow$ **replace**(P^{t+1}, E) # elite is inherited to next generation
  $E \leftarrow$ **bestIndi**(P^{t+1}, F_{t+1}) # update the current best individual
  $t \leftarrow t + 1$
end while
return E

Figure 3.6: The pseudocode of LMM.

As the general process of GA, we first initialize the decomposition solutions randomly in Algorithm 3.6. The object E saves the best decomposition solution. Then, we repeat the procedure of selection, crossover, mutation, evaluation, and inheritance until the iteration reaches the stop criterion from Line 12 to 19. The elitist strategy[104] directly replicates the best individual to the next generation, which can prevent the elite individual from destroying the superior gene and chromosome structure during optimization.

3.2.2 Modified differential evolution with distance-based selection (MDE-DS)

Differential evolution (DE)[49] was first proposed in 1995 and has been wildly applied in data mining, pattern recognition, artificial neural networks, and other fields due to its characteristics such as easy implementation, fast convergence speed, and strong robustness. MDE-DS[105] is designed for continuous optimization problems in the presence of noise with the modification in mutation, crossover, and selection, and the detailed description of MDE-DS is as follows.

Parameter control: The constant F and Cr are unnecessary, as F is randomly sampled from 0.5 to 2 for each mutation operation and Cr is randomly switched between 0.3 to 1 for each target vector. Switching F between two extreme corners of the feasible range is conducive to attaining a balance between exploration and exploitation of the search. There is a new parameter b (the blending rate) in blending crossover, whose value is also randomly chosen from among three candidates: a low value of 0.1, a medium value of 0.5, and a high value of 0.9. The utility of such a switching scheme has been discussed in[106] for solving LSOPs.

Mutation: MDE-DS includes two different mutation strategies and switches them randomly with 50% probability.

In the population centrality-based mutation, the elite sub-population (top 50%) is se-

lected and $\widetilde{\vec{X}}_{best,G}$ is calculated by the arithmetic mean (centroid) of the sub-population individuals. Eq. (3.13) is adopted to mutate the i^{th} individual.

$$\vec{V}_{i,G} = \vec{X}_{r1,G} + F\left(\widetilde{\vec{X}}_{best,G} - \vec{X}_{r2,G}\right) \tag{3.13}$$

where $\vec{X}_{r1,G}$ and $\vec{X}_{r2,G}$ are two different individuals corresponding to randomly chosen indices $r1$ and $r2$. $\vec{V}_{i,G}$ is the newly generated mutant vector corresponding to the current target vector for present generation G.

In the DMP-based mutation scheme, the best individual $\vec{X}_{best,G}$ in each generation is selected and the dimension-wise average is implemented for both $\vec{X}_{best,G}$ and the current target individual $\vec{X}_{i,G}$. The mutation is generated in the following way:

$$\vec{V}_{i,G} = \vec{X}_{i,G} + \Delta_m \cdot \left(\frac{\vec{M}_{i,G}}{\|\vec{M}_{i,G}\|}\right) \tag{3.14}$$

where $\Delta_m = (X_{best_{dim},G} - X_{i_{dim},G})$, with $X_{best_{dim},G} = \frac{1}{D}\sum_{k=1}^{D} x_{best_k,G}$ and $X_{i_{dim},G} = \frac{1}{D}\sum_{k=1}^{D} x_{i_k,G}$. $\frac{\vec{M}_{i,G}}{\|\vec{M}_{i,G}\|}$ is a unit vector with random direction.

The significance of the population centrality-based mutation scheme is that it balances greediness while still maintaining a certain extent of diversity. For example, it is less greedy than the DE/best/1 scheme and hence the probability of the optimization trapped in local optima is less. On the other hand, the DMP-based mutation scheme prefers exploration,[107] and thus, in the absence of any feedback about the nature of the function, an unbiased combination of these two methods is applied.

Crossover: Crossover plays an important role in generating promising offspring from two or more existing individuals within the function landscape. blending crossover is em-

ployed in MDE-DS and described in Eq. (3.15)

$$u_{j,i,G} = \begin{cases} b \cdot x_{j,i,G} + (1-b) \cdot v_{j,i,G}, \ if \ rand_{i,j} \leq 0.5 \ or \ j = j_r \\ x_{j,i,G}, \ otherwise \end{cases} \tag{3.15}$$

where $u_{j,i,G}$ and $v_{j,i,G}$ are the j^{th} dimensions of the trial and donor vectors respectively corresponding to the current index i in generation G and $x_{j,i,G}$ is the j^{th} dimension of the current population individual $\vec{X}_{i,G}$. Blending recombination has one parameter b, which is randomly selected from 0.1, 0.5, and 0.9. The concrete analysis can be referred to in.[105]

Selection: The canonical DE selects the offspring based on a simple greedy strategy. However, suppose the fitness landscape gets corrupted with noise. In that case, the greedy selection suffers a lot because in this case the original fitness of parent and offspring is unknown and it can be well nigh impossible to infer when an offspring is superior or inferior to its parent. Thus, the design of selection is the key to anti-noise. To handle the presence of noise, a novel distance-based selection mechanism is introduced without any extra parameter. There are three cases of the proposed selection mechanism which are described subsequently:

$$\vec{X}_{i,G+1} = \begin{cases} \vec{U}_{i,G}, \ if \ \frac{f(\vec{U}_{i,G})}{f(\vec{X}_{i,G})} \leq 1 \\ \vec{U}_{i,G}, \ if \ \frac{f(\vec{U}_{i,G})}{f(\vec{X}_{i,G})} > 1 \ and \ p_s \leq e^{-\frac{\Delta f}{Dis}} \\ \vec{X}_{i,G}, \ else \end{cases} \tag{3.16}$$

In case 1, when $\frac{f(\vec{U}_{i,G})}{f(\vec{X}_{i,G})} \leq 1$, the offspring replaces the parent and survives to the next generation.

In case 2, although the parent performs better than the offspring, the offspring still can be preserved and survive into the next generation based on a stochastic principle. And the probability is calculated by $e^{-\frac{\Delta f}{Dis}}$, where $\Delta f = \left| f(\vec{U}_{i,G}) - f(\vec{X}_{i,G}) \right|$ represents the absolute fitness difference between $\vec{U}_{i,G}$ and $\vec{X}_{i,G}$, $Dis = \sum_{k=1}^{D} |u_{i,k} - x_{i,k}|$ is the Manhattan

distance between these two vectors. Manhattan distance is applied because of its simplicity and computational efficiency, and p_s is a random number generated from 0 to 1.

In case 3, If the parent significantly outperforms the offspring, then the offspring is removed and the parent persists to the next generation.

This selection process is further illustrated in Figure 3.7.

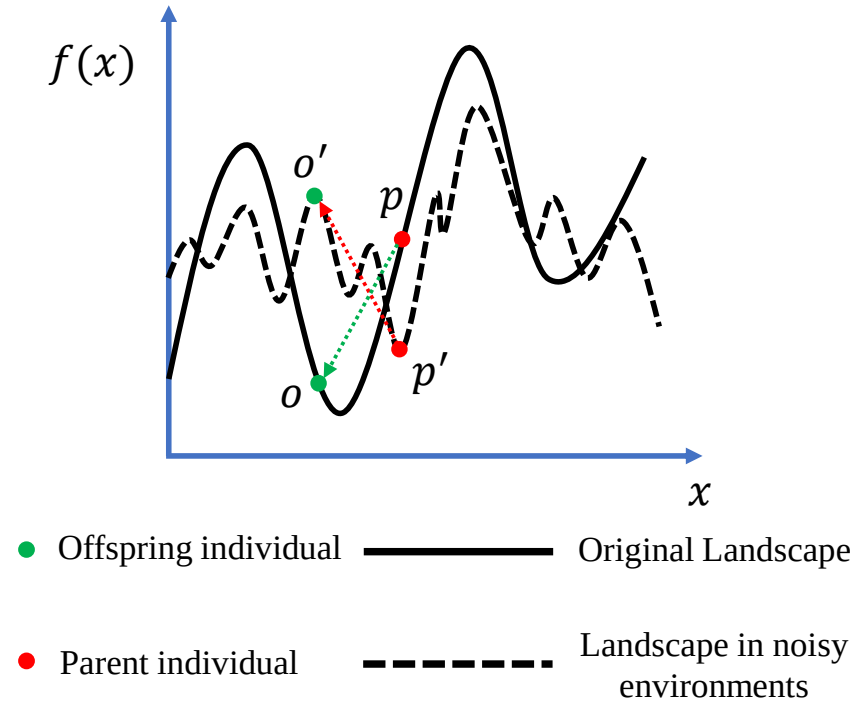


Figure 3.7: A selection works on fitness landscape in noisy environments

Figure 3.7 shows a fitness landscape scenario both in noiseless environments and noisy environments, p and $p^{'}$ represent the parent individual in the original fitness landscape and landscape in noisy environments, o and $o^{'}$ represent the offspring individual in original fitness landscape and landscape in noisy environments respectively. The fitness information we can observe is only in noisy environments, so in minimization problems, $o^{'}$ will be rejected to replace the $p^{'}$ in the next generation. The objective value of p is better than o in the real fitness landscape, and if we re-evaluate the $o^{'}$ and $p^{'}$, the domination may be changed. The mechanism of selection in MDE-DS allows the algorithm to give us some

probabilistic flexibility to select worse solutions as in noise-affected landscapes.

In summary, the pseudocode of MDE-DS is shown in Algorithm 3.8

Algorithm MDE-DS

Require: Population size : s; Maximum iteration : T; Dimension : D
Ensure: Optimized population : X; Fitness of population X : F

$t \leftarrow 0$
$X \leftarrow$ **initialPop**(s, D)
$F \leftarrow$ **evaluate**(X)
while $t < T$ and not stop criterion **do**
  for $i = 0$ *to* s **do**
    ▶ (Mutation)
    $r \leftarrow$ **rand**$(0, 1)$
    if $r \leq 0.5$ **then**
      Update $\vec{V}_{i,G}$ by centrality-based mutation.
    else
      Update $\vec{V}_{i,G}$ by DMP-based mutation.
    end if
    ▶ (Crossover)
    $Cr \leftarrow$ **rand**$(0.3, 1)$
    $b \leftarrow$ **randChoice**$([0.1, 0.5, 0.9])$
    for $j = 0$ *to* D **do**
      Update $u_{j,i,G}$ by blending crossover.
    end for
    ▶ (Selection)
    $F_i \leftarrow$ **evaluate**(U_i)
    $\Delta f_i = \left| f(\vec{U}_i) - f(\vec{X}_i) \right|$
    $Dis = \sum_{k=1}^{D} |u_{i,k} - x_{i,k}|$
    $p_s \leftarrow$ **rand**$(0, 1)$
    Choose $\vec{X}_{i,G+1}$ by distance-based selection.
    $i \leftarrow i + 1$
  end for
  $t \leftarrow t + 1$
end while
return X, F

Figure 3.8: The pseudocode of MDE-DS.

3.3 Mathematical support

It is evident that the optimization guided by LMF can identify the interactions in the noiseless environment because the individuals containing correct linkage information have lower linkage measurement values and higher fitness, which prefer to survive in the selection of EGA. An example we mentioned before is Eq. (3.10). An important explanation is why LMM can identify the interactions in noisy environments. Here, we provide theoretical support.

$Corollary$: Let x_i and x_j be separable decision variables, and x_m and x_n be nonseparable decision variables. $I(x_i, x_j) = (f(s_{ij}) - f(s_i)) - (f(s_j) - f(s))$ and $I^N(x_i, x_j) = (f^N(s_{ij}) - f^N(s_i)) - (f^N(s_j) - f^N(s))$ represent the intensity of interaction between x_i and x_j in noiseless environment and noisy environments respectively. In noisy environments, if we prove that the probability $P(I^N(x_m, x_n) > I^N(x_i, x_j)) > 0$, which means it is possible that the intensity of the interaction between nonseparable variables can be stronger than separable variables in noisy environments, then the minimization of LMF can guide the direction to search for more interactions.

$Proposition$: In noisy environments, the probability $P(I^N(x_m, x_n) > I^N(x_i, x_j)) > 0$, and individuals containing correct detected interactions have better fitness to survive.

$Proof$: In noisy environments, the noise $\beta \sim N(0, \sigma^2)$. The relationship between $I^N(\cdot)$ and $I(\cdot)$ is defined in Eq. (3.17)

$$\begin{aligned} I^N(x_i, x_j) &= I(x_i, x_j) + (\beta_1 f(s_{ij}) - \beta_2 f(s_i)) - (\beta_3 f(s_j) - \beta_4 f(s)) \\ &= I(x_i, x_j) + \phi_{ij} \end{aligned} \tag{3.17}$$

Where $\phi_{ij} = (\beta_1 f(s_{ij}) - \beta_2 f(s_i)) - (\beta_3 f(s_j) - \beta_4 f(s))$, and ϕ_{ij} follows the distribution:

$$\phi_{ij} \sim N(0, (f^2(s_{ij}) + f^2(s_j) + f^2(s_j) + f^2(s))\sigma^2) \tag{3.18}$$

And $I^N(x_i, x_j)$ follows the distribution:

$$I^N(x_i, x_j) \sim N(I(x_i, x_j), (f^2(s_{ij}) + f^2(s_j) + f^2(s_j) + f^2(s))\sigma^2) \tag{3.19}$$

Due to x_i and x_j are separable variables, x_m and x_n are nonseparable variables, similarly:

$$\begin{aligned} I^N(x_i, x_j) &\sim N(0, (f^2(s_{ij}) + f^2(s_j) + f^2(s_j) + f^2(s))\sigma^2) \\ I^N(x_m, x_n) &\sim N(I(x_m, x_n), (f^2(s_{mn}) + f^2(s_m) + f^2(s_n) + f^2(s))\sigma^2) \end{aligned} \tag{3.20}$$

Here, we introduce a distribution $Y = I^N(x_m, x_n) - I^N(x_i, x_j)$, and the problem is transformed to prove $P(Y > 0) > 0$. Y follows the distribution:

$$\begin{aligned} Y \sim N(I(x_m, x_n), (f^2(s_{ij}) + f^2(s_j) + f^2(s_j) + f^2(s))\sigma^2 + \\ (f^2(s_{mn}) + f^2(s_m) + f^2(s_n) + f^2(s))\sigma^2) \\ \sim N(I(x_m, x_n), \sigma_{ijmn}) \end{aligned} \tag{3.21}$$

The expectation of Y is $I(x_m, x_n)$, and two cases need to be discussed:

case 1: $I(x_m, x_n) > 0$: In this case, $P(Y > 0) > 0.5$.

case 2: $I(x_m, x_n) < 0$: In this case, $0 < P(Y > 0) < 0.5$.

In summary, $P(I^N(x_m, x_n) > I^N(x_i, x_j)) > 0$ is true, and the optimization of LMF has the probability of detecting more interactions in noisy environments, which can be employed as the objective function in our experiment.

3.4 Numerical experiments

This Section implements a set of experiments to evaluate our proposal, MDE-DSCC-LMM. Section 3.4.1 introduces the experiment settings, including benchmark functions, comparing methods, and performance indicators. In Section 3.4.2, we provide the experimental results of our proposal and the compared methods.

3.4.1 Experiment Settings

Benchmark Functions: We design 15 test functions in noisy environments based on CEC2013 LSGO Suite, Eq. (3.22) defines the benchmark functions in our experiments.

$$f_i^N(x) = f_i(x) \cdot (1 + \beta),\ i \in [1, 15] \tag{3.22}$$

$\beta \sim N(0, 0.01)$. Briefly, this benchmark suite consists of 15 test functions with 4 categories.

(1) $f_1^N(x)$ to $f_3^N(x)$: fully separable functions in noisy environments;

(2) $f_4^N(x)$ to $f_7^N(x)$: partially separable functions with 7 none-separable parts in noisy environments;

(3) $f_8^N(x)$ to $f_{11}^N(x)$: partially separable functions with 20 none-separable parts in noisy environments;

(4) $f_{12}^N(x)$ to $f_{15}^N(x)$: functions with overlapping sub-problems in noisy environments;

f_{13} and f_{14} consist of 905 decision variables, and the rest functions are 1000-D problems.

Comparing methods and Parameters: In our experiment design, we compare the decomposition strategy of our proposal with various grouping methods. The algorithms applied in the comparisons are listed in Table 3.1. Table 3.2 shows the parameters of our proposal in the decomposition stage. We also conduct the experiment between MDE-DSCC-LMM and DECC-LMM to show the effect of the introduction of MDE-DS. The maximum FEs including decomposition and optimization is $3,000,000$, and the population size of optimization for each sub-problem is set to 30.

Table 3.1: A summary of the algorithms under comparison

Algorithms	Decomposition methods
DECC-D	Delta grouping [108]
DECC-G	Random grouping [20]
DECC-DG	DG [17]
DECC-RDG	RDG [35]
DECC-DG2	DG2 [34]
DECC-LMM MDE-DSCC-LMM	LMM

Performance Indicators: Two stages of our proposal need to be evaluated: LMM and MDE-DSCC.

To evaluate the LMM, three metrics are employed: FEs consumed in decomposition,

Table 3.2: The parameters of decomposition optimization

Parameter	value
Optimization direction	Minimization
Optimizer	EGA
Population size	10
Maximum iteration	100
Genome length	5

decomposition accuracy (DA), and optimization results. We adopt the calculation method of the DA in.[35] Essentially, DA is the ratio of the number of interacting variables that are correctly grouped to the total number of interacting variables. To determine the existence of significance, we apply the Kruskal-Wallis test to the fitness at the end of the optimization in 25 trial runs between different decomposition methods. If significance exists, then we apply the p-value acquired from the Mann-Whitney U test to do the Holm test. If LMM is significantly better than the second-best algorithm, we mark * (significance level 5%) or ** (significance level 10%) in the convergence curve.

To evaluate the MDE-DS, we apply the Mann-Whitney U test between MDE-DSCC-LMM and DECC-LMM. If MDE-DSCC-LMM is significantly better than DECC-LMM, we mark *(significance level 5%) or **(significance level 10%) at the end of optimization.

3.4.2 Performance of our proposal: MDE-DSCC-LMM

In this Section, the performance of MDE-DSCC-LMM is studied, both on the decomposition and optimization. Experiments are conducted on the benchmark functions presented in Section 3.4.1

Performance of LMM: To verify the analysis in Section 3.1 that DG-based decomposition methods cannot detect the interactions in noisy environments, we apply DG, DG2, and RDG to decompose the benchmark functions. Table 3.3 shows the decomposition results.

The decomposition results of DG-based methods prove our analysis, all variables are divided into a sub-problem, and interactions failed to be detected completely. Next, we

Table 3.3: The detailed decomposition results of DG, DG2, RDG on CEC2013 LSGO Suite in noisy environments

Func	Sep. vars	Nonsep. vars	Nonsep. groups	DG($\varepsilon = 10^{-3}$)/DG2/RDG($\alpha = 10^{-10}$)		
				Grouped sep. vars	Grouped nonsep. vars	Formed group's number
f_1	1000	0	0	0/0/0	1000/1000/1000	1/1/1
f_2	1000	0	0	0/0/0	1000/1000/1000	1/1/1
f_3	1000	0	0	0/0/0	1000/1000/1000	1/1/1
f_4	700	300	7	0/0/0	1000/1000/1000	1/1/1
f_5	700	300	7	0/0/0	1000/1000/1000	1/1/1
f_6	700	300	7	0/0/0	1000/1000/1000	1/1/1
f_7	700	300	7	0/0/0	1000/1000/1000	1/1/1
f_8	0	1000	20	0/0/0	1000/1000/1000	1/1/1
f_9	0	1000	20	0/0/0	1000/1000/1000	1/1/1
f_{10}	0	1000	20	0/0/0	1000/1000/1000	1/1/1
f_{11}	0	1000	20	0/0/0	1000/1000/1000	1/1/1
f_{12}	0	1000	1	0/0/0	1000/1000/1000	1/1/1
f_{13}	0	905	1	0/0/0	905/905/905	1/1/1
f_{14}	0	905	1	0/0/0	905/905/905	1/1/1
f_{15}	0	1000	1	0/0/0	1000/1000/1000	1/1/1

provide the DA and FEs for the decomposition of DG, RDG, DG2, and LMM in Table 3.4, because the decomposition results of LMM are different in every trial run, the DA and consumed FEs are calculated with the mean of 25 trial runs. The best DA is in bold.

Finally, the optimization results of DECC-D, DECC-G, DECC-DG, DECC-DG2, DECC-RDG, and DECC-LMM are provided in Table 3.5, the best solution is in bold.

Performance of MDE-DSCC: The mean and standard deviation (std) of the optimum between DECC-LMM and MDE-DSCC-LMM are shown in Table 3.6.

Additionally, the convergence curves of 25 independent runs of all compared methods are shown in Figure 3.9.

Table 3.4: The DA and consumed FEs of DG, RDG, DG2, and LMM on CEC2013 LSGO Suite in noisy environments

Func	DG		DG2		RDG		LMM	
	DA	FEs	DA	FEs	DA	FEs	DA	FEs
f_1	-	2.00e+03	-	5.01e+05	-	6.16e+03	-	6.14e+04
f_2	-	2.00e+03	-	5.01e+05	-	6.16e+03	-	6.21e+04
f_3	-	2.00e+03	-	5.01e+05	-	6.16e+03	-	6.17e+04
f_4	10.0%	2.00e+03	10.0%	5.01e+05	10.0%	6.16e+03	**16.1%**	5.82e+04
f_5	10.0%	2.00e+03	10.0%	5.01e+05	10.0%	6.16e+03	**15.8%**	5.82e+04
f_6	10.0%	2.00e+03	10.0%	5.01e+05	10.0%	6.16e+03	**16.4%**	5.82e+04
f_7	10.0%	2.00e+03	10.0%	5.01e+05	10.0%	6.16e+03	**17.1%**	5.82e+04
f_8	10.0%	2.00e+03	10.0%	5.01e+05	10.0%	6.16e+03	**14.1%**	5.12e+04
f_9	10.0%	2.00e+03	10.0%	5.01e+05	10.0%	6.16e+03	**13.4%**	4.56e+04
f_{10}	10.0%	2.00e+03	10.0%	5.01e+05	10.0%	6.16e+03	**13.8%**	4.67e+04
f_{11}	10.0%	2.00e+03	10.0%	5.01e+05	10.0%	6.16e+03	**14.2%**	4.12e+04
f_{12}	**100%**	2.00e+03	**100%**	5.01e+05	**100%**	6.16e+03	11.9%	4.80e+04
f_{13}	-	1.81e+03	-	4.09e+05	-	5.43e+03	-	4.66e+04
f_{14}	-	1.81e+03	-	4.09e+05	-	5.43e+03	-	4.76e+04
f_{15}	**100%**	2.00e+03	**100%**	5.01e+05	**100%**	6.16e+03	12.7%	4.80e+04

3.5 Discussion

In this Section, we will analyze the performance of LMM and MDE-DS.

3.5.1 LMM in noisy environment

Theoretical analysis in Section 3.3 shows that LMM has the potential to correctly detect the interactions between decision variables in noisy environments. Experimental results in Section 3.4.2 further support this analysis. The identification of interactions in noisy environments is a difficult task, LMM identifies the decision variables with relatively strong intensity as nonseparable. Although the fitness difference will be affected by noise, the relative intensity of interactions between separable variables and nonseparable variables still has a possible gap, which is the main reason for successful implementation in noisy environments.

Table 3.5: Optimization results of DECC-D, DECC-G, DECC-DG, DECC-DG2, DECC-RDG, and DECC-LMM

Func	DECC-D		DECC-G		DECC-DG		DECC-DG2		DECC-RDG		DECC-LMM	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
f_1	1.52e+08	2.81e+07	1.49e+08	2.9e+07	1.92e+09	2.81e+08	2.44e+09	2.56e+08	2.00e+09	2.62e+08	**6.10e+07**	**1.65e+07**
f_2	8.10e+04	3.30e+03	7.89e+04	3.19e+03	9.16e+04	3.35e+03	9.24e+04	2.43e+03	9.09e+04	3.43e+03	**5.65e+04**	**9.39e+03**
f_3	2.11e+01	8.01e-02	2.12e+01	5.07e-02	2.11e+01	1.07e-01	2.12e+01	8.75e-02	2.11e+01	1.11e-01	**2.11e+01**	**8.62e-02**
f_4	1.48e+12	6.30e+11	1.38e+12	6.36e+11	9.69e+11	2.41e+11	1.12e+12	2.34e+11	8.71e+11	2.31e+11	**6.75e+11**	**2.31e+11**
f_5	1.36e+07	1.47e+06	1.27e+07	1.60e+06	1.53e+07	1.15e+06	1.54e+07	1.30e+06	1.56e+07	1.73e+06	**1.06e+07**	**1.03e+06**
f_6	1.048e+06	2.98e+03	1.049e+06	3.33e+03	1.048e+06	3.16e+03	1.051e+06	3.70e+03	1.050e+06	3.94e+03	**1.044e+06**	**5.88e+03**
f_7	3.77e+09	1.65e+09	3.70e+09	1.06e+09	3.40e+09	7.28e+08	3.84e+09	1.21e+09	4.04e+09	6.31e+08	**2.88e+09**	**1.14e+09**
f_8	2.73e+16	1.20e+16	3.26e+16	2.13e+16	4.76e+16	1.79e+16	5.16e+16	1.84e+16	4.87e+16	2.07e+16	**1.99e+16**	**5.33e+15**
f_9	9.41e+08	1.86e+08	9.38e+08	1.81e+08	1.15e+09	1.16e+08	1.20e+09	7.30e+07	1.14e+09	9.42e+07	**8.22e+08**	**1.07e+08**
f_{10}	9.33e+07	3.98e+05	9.31e+07	4.12e+05	9.31e+07	3.79e+05	9.33e+07	4.43e+05	9.32e+07	5.21e+05	**9.21e+07**	**6.88e+05**
f_{11}	4.41e+11	1.63e+11	5.46e+11	2.40e+11	4.57e+11	1.63e+11	5.05e+11	1.26e+11	4.77e+11	1.23e+11	**2.54e+11**	**8.98e+10**
f_{12}	5.82e+12	2.68e+11	5.88e+12	2.05e+11	6.54e+12	2.46e+11	6.55e+12	2.01e+11	6.47e+12	2.06e+11	**3.55e+12**	**3.59e+11**
f_{13}	2.79e+10	9.27e+09	2.63e+10	5.61e+09	3.69e+10	6.90e+09	4.11e+10	7.81e+09	3.92e+10	8.94e+09	**1.81e+10**	**3.42e+09**
f_{14}	5.94e+11	1.69e+11	5.60e+11	1.47e+11	6.96e+11	1.51e+11	7.97e+11	1.45e+11	7.15e+11	1.17e+11	**3.68e+11**	**6.65e+10**
f_{15}	4.34e+07	7.92e+06	4.04e+07	5.55e+06	6.03e+07	1.42e+07	7.05e+07	1.29e+07	5.65e+07	8.69e+06	**2.20e+07**	**2.37e+06**

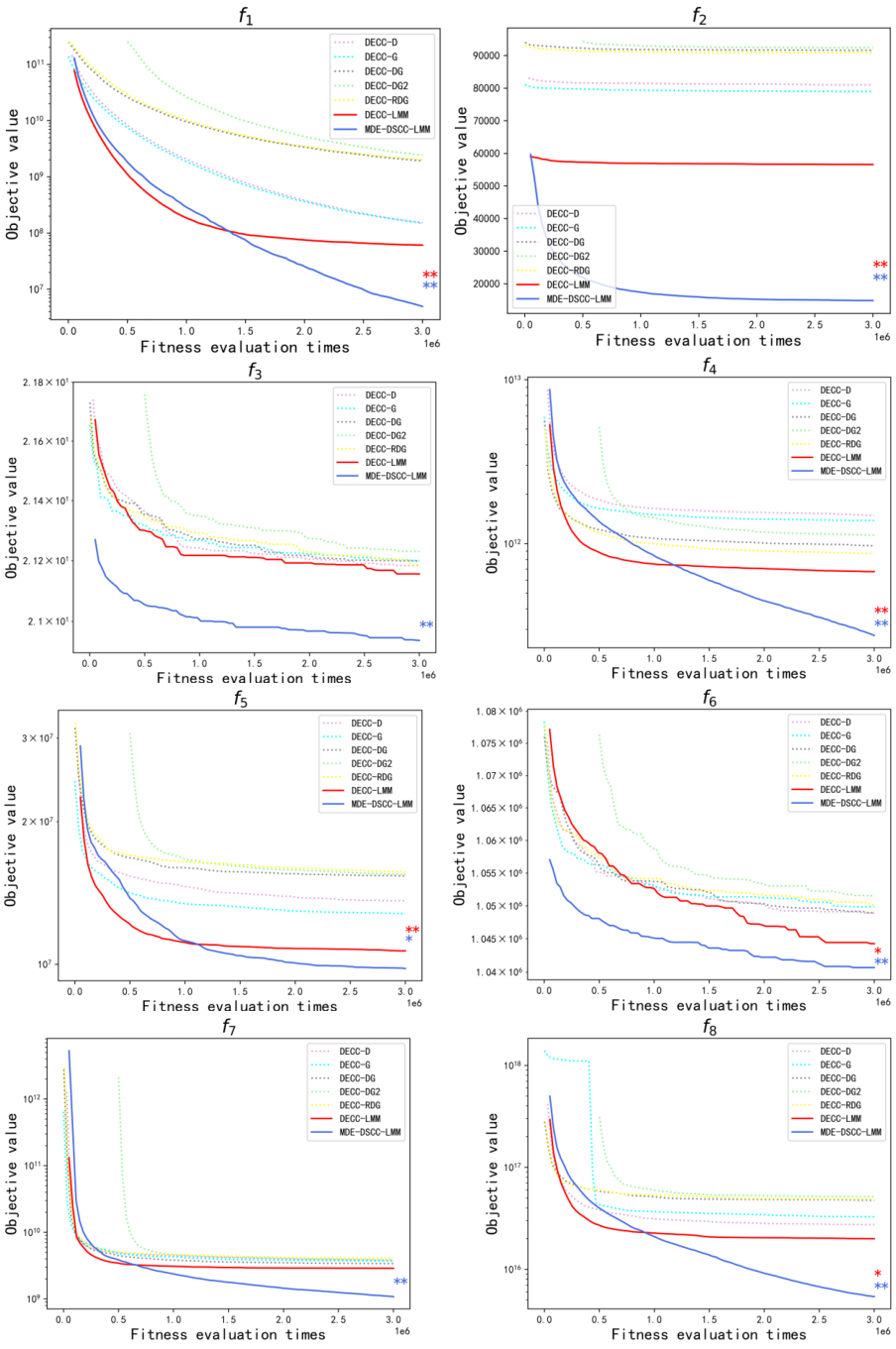
Table 3.6: Optimization results between DECC-LMM and MDE-DSCC-LMM

Func	DECC-LMM		MDE-DSCC-LMM	
	mean	std	mean	std
f_1	6.10e+07	1.65e+07	**4.95e+06**	**2.49e+06**
f_2	5.65e+04	9.39e+03	**1.48e+04**	**9.15e+02**
f_3	2.11e+01	8.62e-02	**2.09e+01**	**4.02e-02**
f_4	6.75e+11	2.31e+11	**2.75e+11**	**1.64e+11**
f_5	1.06e+07	1.03e+06	**9.79e+06**	**1.63e+06**
f_6	1.044e+06	5.88e+03	**1.040e+06**	**2.72e+03**
f_7	2.88e+09	1.14e+09	**1.08e+09**	**3.03e+08**
f_8	1.99e+16	5.33e+15	**5.39e+15**	**1.53e+15**
f_9	**8.22e+08**	**1.07e+08**	8.80e+08	9.42e+07
f_{10}	**9.21e+07**	**6.88e+05**	9.39e+07	4.77e+05
f_{11}	2.54e+11	8.98e+10	**1.61e+11**	**8.12e+10**
f_{12}	3.55e+12	3.59e+11	**5.11e+08**	**1.27e+08**
f_{13}	1.81e+10	3.42e+09	**1.17e+10**	**3.10e+09**
f_{14}	3.68e+11	6.65e+10	**2.59e+11**	**7.79e+10**
f_{15}	2.20e+07	2.37e+06	**9.58e+06**	**1.44e+06**

However, the optimization of LMF is not an easy task. From the DA in Table 3.4, the interactions that LMM can detect in noisy environments are limited. Although LMF can lead the direction of optimization to search for more correct interactions, a more powerful optimizer will allow LMM to find more interactions.

3.5.2 LMM vs. DG-based decomposition methods

DG-based decomposition methods detect the interactions by determining the difference between the fitness difference and a certain parameter ϵ, and even in fully separable functions, the fitness difference will be amplified by noise and larger than ϵ easily, which is the main reason of detection failure in noisy environments, and all decision variables are divided into a sub-problem and optimized directly. Due to the curse of dimensionality, it is difficult for DE to find an acceptable solution with this division. Thus, although the DA of DG-based methods is higher than LMM in f_{12} and f_{15}, LMM still performed better than



DG-based methods in the optimization of these functions, and DG-based methods are the most environmentally sensitive grouping method among the compared methods.

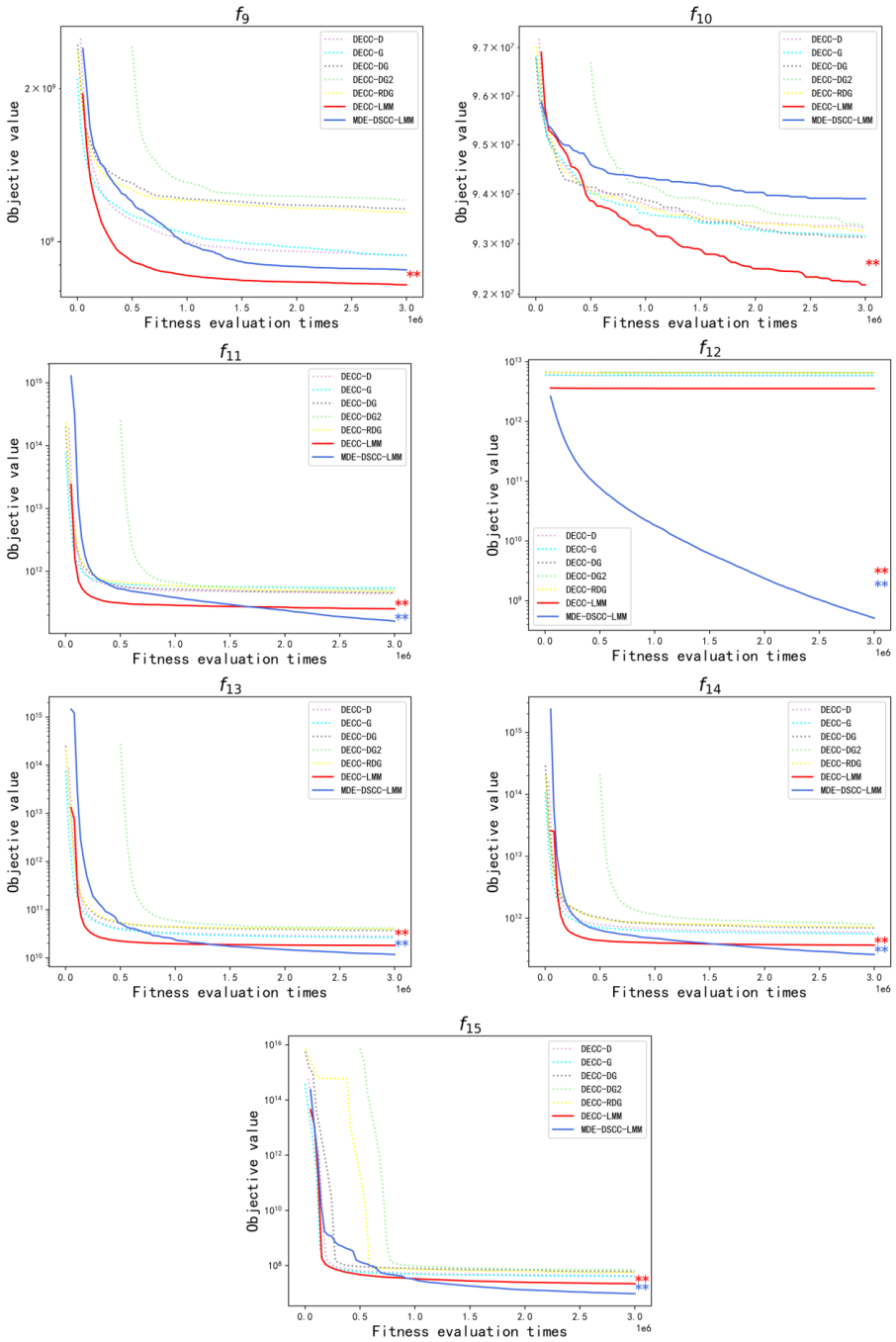


Figure 3.9: The convergence curve of DECC-D, DECC-G, DECC-DG, DECC-DG2, DECC-RDG, DECC-LMM, and MDE-DSCC-LMM. The gap in the initial period is FEs consumed for decomposition.

3.5.3 LMM vs. D

The schematic diagram of Delta Grouping is shown in Figure 3.10. Delta grouping distin-

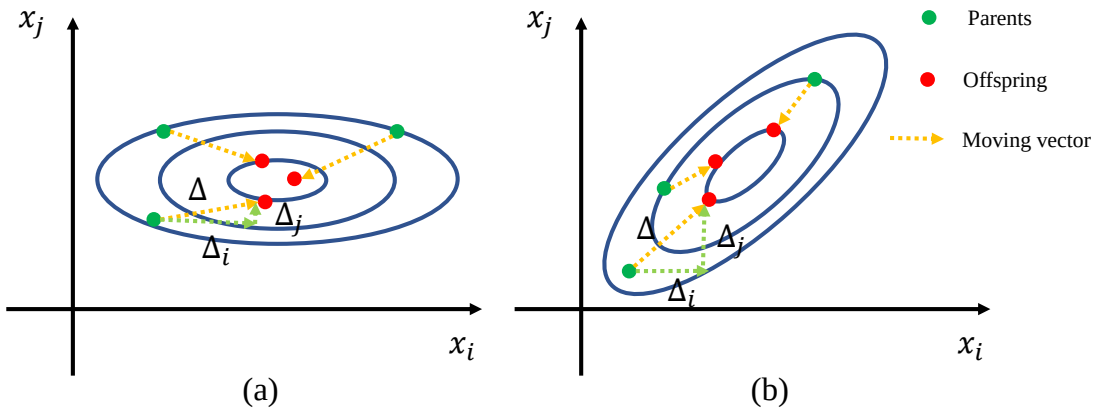


Figure 3.10: (a). Delta grouping works on the separable function. (b). Delta grouping works on the nonseparable function.

guishes between separable and nonseparable variables based on the disparity in coordinates between the initial random population and the optimized population. As depicted in Figure 3.10, when the difference between Δ_i and Δ_j is significant, Delta grouping identifies variables x_i and x_j as separable. However, this rough estimation is susceptible to noise since Delta grouping samples from the fitness landscape, and the moving vector remains influenced by noise. Consequently, Delta grouping ranks second in sensitivity among our comparison methods. Experimental results presented in Table 3.5 and Figure 3.9 consistently demonstrate the superior performance of our proposed LMM over DECC-D.

3.5.4 LMM vs. Random grouping

It is unnecessary to provide any information about the fitness landscape to Random grouping, therefore, Random grouping is the most environmentally insensitive decomposition method. Although[20] has proven that Random grouping is efficient and has a high probability of capturing some interactions, it cannot detect sufficient interactions and form subproblems properly, and LMM can detect more corrected interactions, which is the main reason that LMM outperformed than Random grouping.

3.5.5 The efficiency of MDE-DS

Figure 3.9 and Table 3.6 all prove that MDE-DS has a strong ability to search for better solutions compared with the canonical DE in most benchmark functions, although canonical DE performs better in f_9 and f_{10}. However, no optimization algorithm can solve all optimization problems perfectly. According to No Free Lunch Theory[109] in optimization, the average performance of any pair of algorithms A and B is identical on all possible problems. Therefore, if an algorithm performs well on a certain class of problems, it must pay for that with performance degradation on the remaining problems, since this is the only way for all algorithms to have the same performance on average across all functions. Thus, although MDE-DS may perform worse than the canonical DE in noiseless functions, it is a great success in introducing MDE-DS to solve problems in noisy environments.

3.6 Conclusion

This work introduces a novel strategy that treats the decomposition problem as a combinatorial optimization challenge and devises the linkage measurement function (LMF) to direct the optimization process. Additionally, we introduce an advanced optimizer named modified DE with distance-based selection (MDE-DS) to address optimization tasks in noisy environments. Our numerical experiments demonstrate that LMM exhibits competitive performance in identifying interactions within noisy environments when compared to other grouping methods. Moreover, MDE-DS showcases robust search capabilities, significantly enhancing optimization processes in noisy environments.

Chapter 4

Linkage identification in computationally expensive problems

This chapter introduces the linkage identification in computationally expensive problems[1]. Section 4.1 presents the motivation of this research, Section 4.2 introduces related works, Section 4.3 details our proposal: efficient dual differential grouping (EDDG) and surrogate ensemble assisted differential evolution (SEADE), Section 4.4 describes the experiments on the CEC2013 LSGO suite, and Section 4.5 discusses the performance of SEADECC-EDDG. Finally, Section 4.6 concludes this chapter.

4.1 Motivation

Two critical components of implementing the CC framework are the decomposition strategy and the optimization technique. Over time, numerous techniques have been proposed to decompose LSOPs. Among these methods, one of the most widely used mechanisms for decomposition is the differential grouping (DG)[17] and its subsequent extensions.[31,32,111]

[1]This chapter was previously published as Zhong, R., Zhang, E. & Munetomo, M. Cooperative co-evolutionary surrogate ensemble-assisted differential evolution with efficient dual differential grouping for large-scale expensive optimization problems. Complex Intell. Syst. 10, 2129–2149 (2024). https://doi.org/10.1007/s40747-023-01262-6.[110]

These approaches identify additive interactions based on fitness differences between perturbed samples and are renowned for their precision in decomposition. However, the original DG incurs high computational costs for decomposition. In extreme cases, DG may require 1,001,000 fitness evaluations (FEs) for a 1000-dimensional fully separable function, rendering it impractical to extend DG to address large-scale expensive optimization problems (LSEOPs) encountered in real-world applications such as aerodynamic design optimization,[112] flow-shop scheduling problems,[113] and others.

Nevertheless, recursive DG (RDG)[35] enables the extension of DG to LSEOPs. RDG examines interactions between subsets rather than individual variables using a binary search approach, significantly reducing the average number of FEs to $1.47e4$ on CEC2010 and CEC2013 benchmark functions. Furthermore, efficient RDG (ERDG)[36] and merged DG (MDG)[37] further minimize computational requirements and enhance decomposition accuracy. These advancements hold great promise for addressing LSEOPs based on the CC framework with DG-based decomposition techniques.

Another crucial component of the CC framework is optimization. Conventional EAs are often characterized as stochastic search methods,[114] requiring a large number of FEs to discover satisfactory solutions. This limitation severely hampers the scalability of EAs when dealing with expensive optimization problems (EOPs).[115] To address this challenge, a well-established approach is to employ relatively low-computational surrogate models to assist in EA optimization. This category of algorithms is known as surrogate-assisted EAs (SAEAs).[116] Various techniques can be utilized to construct surrogate models, including radial basis functions (RBF),[117] polynomial regression (PR) models,[118] Gaussian process regression (GPR)[119] or Kriging,[120] and artificial neural networks (ANN).[121] However, the performance of SAEAs is closely tied to the quality of the surrogate model, and the curse of dimensionality in LSEOPs can rapidly degrade the accuracy and quality of surrogate models. While some studies[122–124] have employed dimension reduction techniques and efficient surrogate model management schemes to address high-dimensional EOPs, most

focus on problems with fewer than 200 decision variables and seldom tackle scales of 1000 dimensions. To confront the substantial challenge posed by 1000-dimensional LSEOPs, we propose integrating the CC framework with SAEAs to address LSEOPs, which is also the primary motivation behind our research.

Various contributions have been made by scholars to address LSEOPs. Ivanoe et al.[125] were among the pioneers who introduced SAEAs into the CC framework, proposing a surrogate-assisted CC (SACC) optimizer. Ren et al.[126] further advanced the SACC framework by integrating it with RBF-embedded success-history-based adaptive differential evolution (RBF-SHADE-SACC). Sun et al.[26] proposed a surrogate ensemble-assisted large-scale expensive evolutionary algorithm with random grouping (SEA-LEEA-RG) for LSEOPs. This approach utilizes a random grouping strategy to decompose the original problem, constructs local and global RBF models using partial and all samples, and incorporates the best-predicted sample into the subsequent optimization process. Similarly, Sun et al.[27] combined the random grouping strategy with RBF-assisted DE in their LSEO with a switching strategy (LSEO-SS). Here, the switching strategy governs the utilization of the local and global surrogate models, while a unique escape mechanism involving re-initialization is designed to prevent premature convergence. Additionally, Gu et al.[28] proposed the SADE with adaptive multi-subspace search (SADE-AMSS) for LSEOPs. In decomposition, SADE-AMSS employs principal component analysis (PCA) and random decision variable selection to construct multiple sub-spaces, integrating three efficient search strategies adaptively within SADE. However, the majority of studies primarily focus on optimizing algorithms and employ only random strategies in sub-component division. While capturing interactions between decision variables based on probability can be efficient, accurate decomposition techniques often require additional FEs for interaction identification. Moreover, incorrectly determined interactions may misdirect the optimization process.[127] Therefore, achieving both accurate decomposition and optimal performance in LSEOPs necessitates the integration of high-accuracy, computationally inexpensive decomposition

methods with efficient optimizers.

4.2 Related works

4.2.1 Generalized regression neural network (GRNN)

GRNN is a highly parallel radial basis function network based on the one-pass algorithm,[3,128] and the universal approximation theorem[129] endows an ability to GRNN for approximating any functions theoretically. Specifically, the topology of GRNN is shown in Figure 4.1

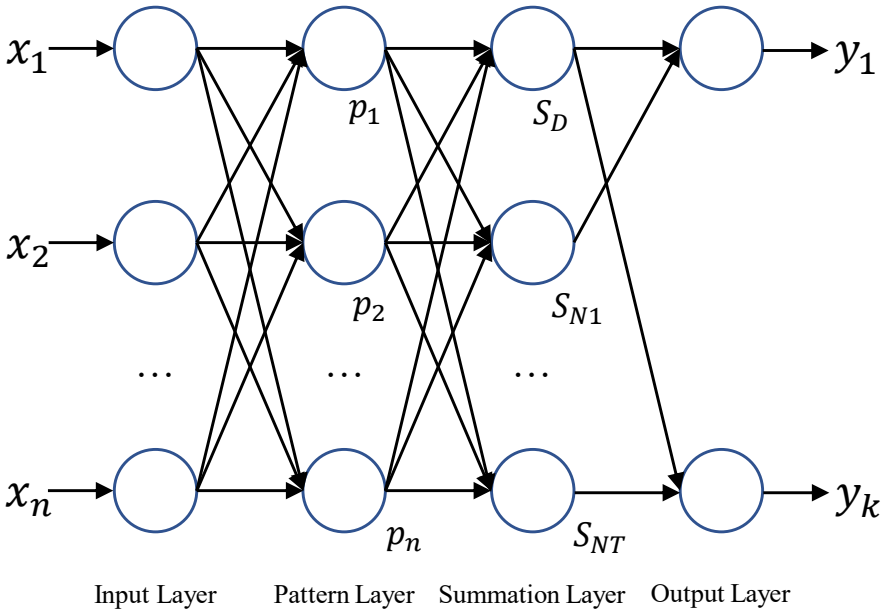


Figure 4.1: The structure of GRNN.[3]

GRNN consists of four layers: the input layer, pattern layer, summation layer, and output layer. Eq. (4.1) summarizes the GRNN logic in an equivalent nonlinear regression formula:[130]

$$E[Y/X] = (\int_{-\infty}^{+\infty} Y f(X,Y)dY)/(\int_{-\infty}^{+\infty} f(X,Y)dY) \tag{4.1}$$

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.[131]

4.2.2 Gaussian process regression (GPR)

GPR as a popular probabilistic surrogate model has been widely applied in SAEAs.[132–134] Deriving from the Bayesian theory, the GPR can be seen as a random process to undertake the non-parametric regression with the Gaussian processes.[45] For any inputs, the corresponding probability distribution over function $f(x)$ follows the Gaussian distribution as:

$$f(x) \sim \mathcal{GPR}(m(x), k(x, x')) \tag{4.2}$$

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

$$\begin{aligned} m(x) &= E(f(x)) \\ k(x, x') &= E[(m(x) - f(x'))(m(x) - f(x'))] \end{aligned} \tag{4.3}$$

$E(\cdot)$ 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

$$y \sim N(0, k(x, x') + \sigma_n^2 I_n) \tag{4.4}$$

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[135]

$$\begin{bmatrix} \frac{y}{y'} \end{bmatrix} \sim N(0, \begin{bmatrix} k(x,x) + \sigma_n^2 I_n & k(x,x') \\ k(x,x')^T & k(x',x') \end{bmatrix}) \tag{4.5}$$

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[136]

$$\begin{aligned} L(\theta) = \frac{1}{2}\log[\det \lambda(\theta) + \frac{1}{2} y^T \lambda^{-1}(\theta) y + \frac{n}{2}\log(2\pi)] \\ \lambda(\theta) = k(\theta) + \sigma_n^2 I_n \end{aligned} \tag{4.6}$$

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

$$\begin{aligned} p(y'|x', x, y) \sim N(y'|\bar{y}', \mathrm{cov}(y')) \\ \bar{y}' = k(x,x')^T [k(x,x) + \sigma_n^2 I_n]^{-1} y \\ \mathrm{cov}(y') = k(x',x') - \\ k(x,x')^T [k(x,x) + \sigma_n^2 I_n]^{-1} k(x,x') \end{aligned} \tag{4.7}$$

where $\bar{y}'$ stands for the corresponding mean values of prediction. $\mathrm{cov}(y')$ denotes a variance matrix to reflect the uncertainty range of these predictions. More details of mathematical support can be found in.[137]

4.3 Proposal: SEADECC-EDDG

This chapter introduces 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[111] and RDG3,[138] 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.

4.3.1 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 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 larger than the pre-defined maximum scale. In summary, the pseudocode of EDDG is shown in Figure 4.2.

NaN is a non-numeric value. In Algorithm 4.2, EDDG first detects the interactions

Algorithm 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: $nonsep$

$sep, nonsep \leftarrow \emptyset$
$\mathbf{x}_{l,l} \leftarrow \mathbf{lb}$; $y_{l,l} \leftarrow f(\mathbf{x}_{l,l})$
$X_1 \leftarrow \{x_1\}$; $X_2 \leftarrow \{x_2, x_3, ..., x_N\}$
while X_2 is not empty **do**
    $\mathbf{x}_{u,l} \leftarrow \mathbf{x}_{l,l}$; $\mathbf{x}_{u,l}(X_1) \leftarrow \mathbf{ub}(X_1)$
    $y_{u,l} \leftarrow f(\mathbf{x}_{u,l})$
    $FA \leftarrow \{y_{l,l}, y_{u,l}, NaN, NaN\}$
    $(X_1^*, \hat{\beta}_{add}, \hat{\beta}_{multi}) \leftarrow \mathbf{Interact}(X_1, X_2, \mathbf{x}_{l,l}, \mathbf{x}_{u,l}, \mathbf{lb}, \mathbf{ub}, F)$
    if $\mathbf{len}(X_1^*) = \mathbf{len}(X_1)$ **then**
        if $\mathbf{len}(X_1^*) > 1$ **then**
            $nonsep \leftarrow \{nonsep, X_1^*\}$
        else
            $sep \leftarrow \{sep, X_1^*\}$
        end if
        $X_1 \leftarrow \mathbf{pop}(X_2)$
    else
        $X_1 \leftarrow X_1^*$; $X_2 \leftarrow X_2 - X_1$
    end if
    if $X_2 = \emptyset$ **then**
        if $\mathbf{len}(X_1) >$ MS **then**
            Separate X_1 to X_{11}, X_{12},...
            $nonsep \leftarrow \{nonsep, X_{11}, X_{11}, ...\}$
        else if $\mathbf{len}(X_1) > 1$ **then**
            $nonsep \leftarrow \{nonsep, X_1\}$
        else
            $sep \leftarrow \{sep, X_1\}$
        end if
    end if
end while
return $sep, nonsep$

Figure 4.2: The pseudocode of EDDG.

between x_1 and the rest decision variables. If these two subsets are separable, 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, otherwise, it is allocated to a separable decision variable. 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. EDDG repeats the above process until all decision variables are well placed. Besides, the main difference between EDDG and ERDG is 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

$$Acc = \frac{MS}{D} \cdot 100\% \tag{4.8}$$

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 4.2 shows the framework of EDDG which mainly inherits the skeleton of ERDG, and Figure 4.3 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 4.3, EDDG first calculates the additive difference and multiplicative difference, 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 $X_2^{''}$, 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 $X_2^{''}$, 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 $X_2^{''}$, which can save the FEs for decomposition.

Similarly, our proposed EDDG inherits the interaction identification technique of ERDG

Algorithm 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: β_{add}, Multiplicative fitness difference: β_{multi},

$nonSep \leftarrow 1$
if $FA[2] = NaN$ **then**
  $\mathbf{x}_{m,l} \leftarrow \mathbf{x}_{l,l}$; $\mathbf{x}_{u,m} \leftarrow \mathbf{x}_{u,l}$
  $\mathbf{x}_{m,l}(X_2) \leftarrow (\mathbf{lb}(X_2) + \mathbf{ub}(X_2))/2$
  $\mathbf{x}_{u,m}(X_2) \leftarrow (\mathbf{lb}(X_2) + \mathbf{ub}(X_2))/2$
  $FA[2] \leftarrow f(\mathbf{x}_{m,l})$; $FA[3] \leftarrow f(\mathbf{x}_{u,m})$;
  $\Delta_1 \leftarrow FA[0] - FA[1]$; $\Delta_2 \leftarrow FA[2] - FA[3]$
  $\beta_{add} \leftarrow \mathbf{abs}(\Delta_1 - \Delta_2)$
  $\beta_{multi} \leftarrow \beta_{multi}^{MAX}$
  if elements in FA are all positive **then**
    $\Delta_3 \leftarrow \ln(FA[0]) - \ln(FA[1])$
    $\Delta_4 \leftarrow \ln(FA[2]) - \ln(FA[3])$
    $\beta_{multi} \leftarrow \mathbf{abs}(\Delta_3 - \Delta_4)$
  end if
  if $\beta_{add} < \varepsilon_{add}$ or $\beta_{multi} < \varepsilon_{multi}$ **then**
    $nonSep \leftarrow 0$
  end if
end if
if $nonSep = 1$ **then**
  if $\mathbf{len}(X_2) > 1$ **then**
    Divide X_2 into two equal-sized and mutually exclusive subsets $X_2^{'}$ and $X_2^{''}$
    $tFA \leftarrow \{FA[0], FA[1], NaN, NaN\}$
    $(X_1^{'}, \hat{\beta}_{add}, \hat{\beta}_{multi}) \leftarrow \mathbf{Interact}(X_1, X_2^{'}, \mathbf{x}_{l,l}, \mathbf{x}_{u,l}, \mathbf{lb}, \mathbf{ub}, tFA)$
    if $\beta_{add} \neq \hat{\beta}_{add}$ and $\beta_{multi} \neq \hat{\beta}_{multi}$ **then**
      if $\mathbf{len}(X_1^{'}) = \mathbf{len}(X_1)$ **then**
        $(X_1^{''}, \beta_{add}^{'}, \beta_{multi}^{'}) \leftarrow \mathbf{Interact}(X_1, X_2^{''}, \mathbf{x}_{l,l}, \mathbf{x}_{u,l}, \mathbf{lb}, \mathbf{ub}, F)$
      else
        $tFA \leftarrow \{FA[0], FA[1], NaN, NaN\}$
        $(X_1^{''}, \beta_{add}^{'}, \beta_{multi}^{'}) \leftarrow \mathbf{Interact}(X_1, X_2^{''}, \mathbf{x}_{l,l}, \mathbf{x}_{u,l}, \mathbf{lb}, \mathbf{ub}, tFA$
      end if
      $X_1 \leftarrow \{X_1^{'}, X_1^{''}\}$
    else
      $X_1 \leftarrow X_1^{'}$
    end if
  else
    $X_1 \leftarrow \{X_1, X_2\}$
  end if
end if
return $X_1, \beta_{add}, \beta_{multi}$

Figure 4.3: The pseudocode of interaction identifications.

and embeds the subset-to-subset version of DDG to EDDG. 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 4.3 is invoked by Algorithm 4.2 $t = D - 1$ 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 4.3 is invoked by Algorithm 4.2 $t = 1$ 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,[36] 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,[38] 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

$$\begin{aligned} &\Delta_1 \neq \Delta_2 \\ &\Delta_1 = \ln(f(\mathbf{x} + l_1\mathbf{u}_1^*)) - \ln(f(\mathbf{x})) \\ &\Delta_2 = \ln(f(\mathbf{x} + l_1\mathbf{u}_1^* + l_2\mathbf{u}_2^*)) - \ln(f(\mathbf{x} + l_2\mathbf{u}_2^*)) \end{aligned} \tag{4.9}$$

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,[38] which can be formulated as

$$\begin{aligned}
&\Delta_1 = \ln(f(s_i)) - \ln(f(s)) \\
&\Delta_2 = \ln(f(s_{ij})) - \ln(f(s_j)) \\
&\text{If } |\Delta_1 - \Delta_2| < \varepsilon_{multi} \\
&\text{Then } x_i \; and \; x_j \; are \; multiplicatively \; separable
\end{aligned} \tag{4.10}$$

where s is a trial solution, s_i and s_j perturb δ on s in the i^{th} and j^{th} dimension respectively, and s_{ij} perturb δ in i^{th} and j^{th} dimension simultaneously. Here, we define $g(s) = \ln(f(s))$. Given $f(s) > 0$ is always true, thus, this definition holds, and Eq. (4.10) can be transformed to Eq. (4.11), which is equivalent to the canonical version of DG.

$$\begin{aligned}
&\Delta_1 = g(s_i) - g(s) \\
&\Delta_2 = g(s_{ij}) - g(s_j) \\
&\text{If } |\Delta_1 - \Delta_2| < \varepsilon_{multi} \\
&\text{Then } x_i \; and \; x_j \; are \; multiplicatively \; separable
\end{aligned} \tag{4.11}$$

the only difference is the allowable error ε_{multi}. Therefore, we can reasonably extend Eq. (4.11) to the subset-to-subset version based on the theoretical fundamentals in RDG[35] and MDG:[37]

$$\begin{aligned}
&\Delta_1 = g(\mathbf{x}^* + l_1\mathbf{u}_1 + l_2\mathbf{u}_2) - g(\mathbf{x}^* + l_2\mathbf{u}_2) \\
&\Delta_2 = g(\mathbf{x}^* + l_1\mathbf{u}_1) - g(\mathbf{x}^*) \\
&\text{If } |\Delta_1 - \Delta_2| < \varepsilon_{multi} \\
&\text{Then } x_i \; and \; x_j \; are \; multiplicatively \; separable
\end{aligned} \tag{4.12}$$

and we replace the $g(\mathbf{x})$ to $\ln(f(\mathbf{x}))$, which is the subset-to-subset version of DDG. Thus, Eq. (4.9) 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 4.1.

Table 4.1: The design of threshold ε in popular DG-based techniques

Algorithms	Setting of the threshold
DG [17]	10^{-3}
FII [139]	10^{-2}
RDG [35]	$\alpha * \min\{\|f(\mathbf{x}_1)\|, ..., \|f(\mathbf{x}_{10})\|\}$
RDG2 [111] ERDG [36]	$\gamma_{\sqrt{n}+2} * (\|f(\mathbf{x}_{l,l})\| + \|f(\mathbf{x}_{u,l})\| + \|f(\mathbf{x}_{l,m})\| + \|f(\mathbf{x}_{u,m})\|)$
DDG [38]	$\varepsilon_{add} = 10^{-3}$ $\varepsilon_{multi} = 10^{-8}$
MDG [37]	$2^{-52} * c * n * f_{max}$
EDDG	$\varepsilon_{add} = \gamma_{\sqrt{n}+2} * (\|f(\mathbf{x}_{l,l})\| + \|f(\mathbf{x}_{u,l})\| + \|f(\mathbf{x}_{l,m})\| + \|f(\mathbf{x}_{u,m})\|)$ $\varepsilon_{multi} = \gamma_{\sqrt{n}+2} * (\|\ln(f(\mathbf{x}_{l,l}))\| + \|\ln(f(\mathbf{x}_{u,l}))\| + \|\ln(f(\mathbf{x}_{l,m}))\| + \|\ln(f(\mathbf{x}_{u,m}))\|)$

In Table 4.1, 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,[31] where $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{10}$ are randomly sampled solutions, and $\alpha = 10^{-12}$ is suggested in.[35] 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.[140] 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.

4.3.2 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 4.4 shows the flowchart of SEADE to optimize a sub-component.

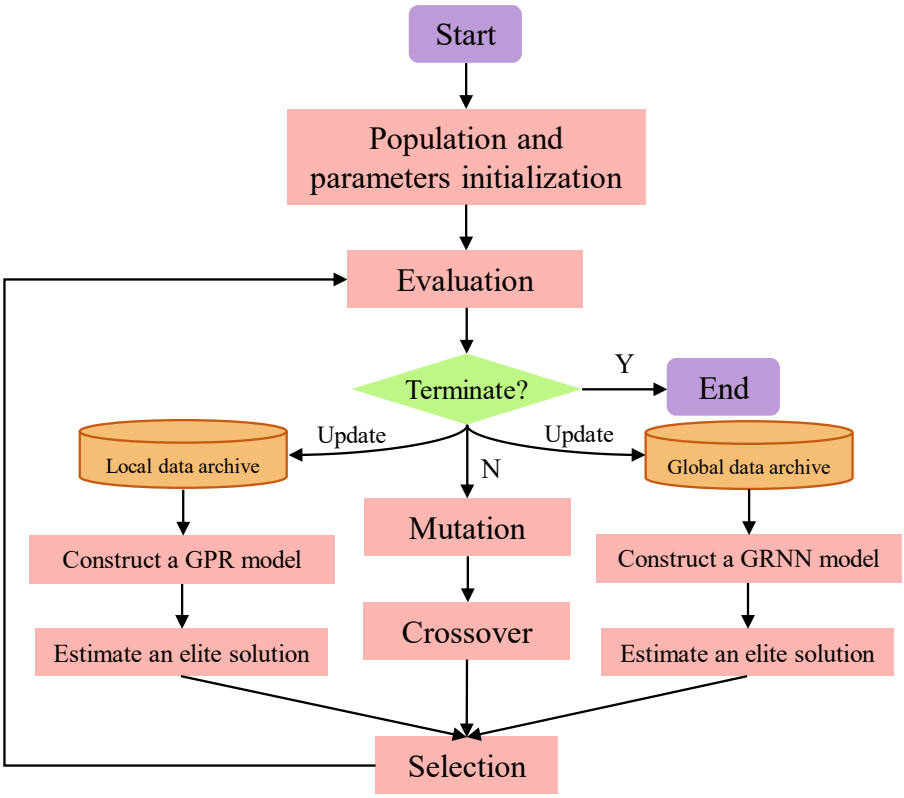


Figure 4.4: The flowchart of SEADE.

For each sub-component optimization, SEADE first initializes the population and parameters and evaluates the population by the objective function. If the termination criterion 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.[141–143] Gaussian process regression (GPR), also known as the Kriging model, is a highly flexible model that can formulate uncertainty with great precision to build update points,[144] and it has been employed as the local surrogate model in many pieces of literature.[145–147] 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 4.5 provides an example of two training data selection fashions for model construction. Figure 4.5 (b) selects the recent data to construct the GRP, which can approximate the local fitness landscape around the current individual. Figure 4.5 (c) selects all historical data to construct the GRNN, and this scheme can learn the global information of the fitness landscape.

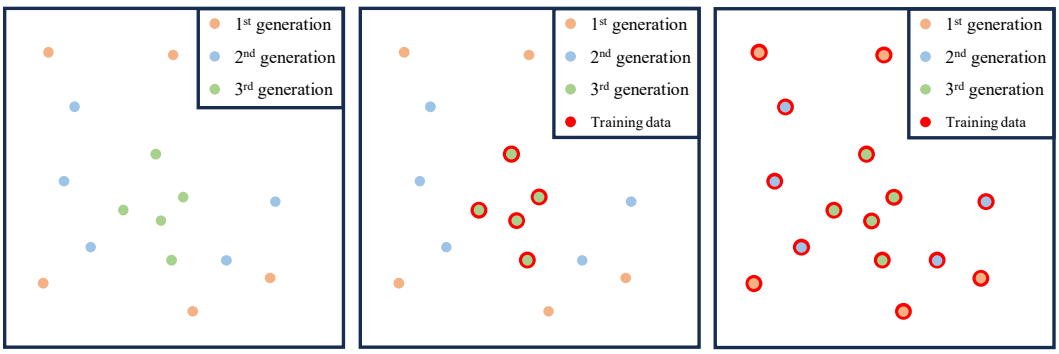


Figure 4.5: Training data for constructing surrogate models, and the sample surrounded by the red circle is the selected data. (a). The distribution of individuals as the generation increases. (b). The training data for constructing the GPR. (c). The training data for constructing the GRNN.

Besides, given the values 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,[148] 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. (4.13)

$$\mathrm{EI}(x) = \max(f(x^*) - f(x), 0) \tag{4.13}$$

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 Figure 4.6.

Algorithm Random search

Require: Surrogate model: $f(\cdot)$, Initial solution: x, Lower and Upper bound: **lb, ub**

Ensure: Estimated solution: x^*

$x^* \leftarrow x$
while not Terminate **do**
  Construct solution $x^{'}$ within **lb** and **ub** randomly
  if $f(x^{'})$ has better EI than $f(x^*)$ **then**
    $x^* \leftarrow x^{'}$
  end if
end while
return x^*

Figure 4.6: The pseudocode of random search.

In summary, the pseudocode of our proposed SEADE is shown in Figure 4.7.

4.4 Numerical experiments

In this section, we implement numerical experiments to evaluate our proposal: SEADECC-EDDG. Section 4.4.1 introduces the experiment settings, and section 4.4.2 provides the experimental results of our proposal with compared methods.

Algorithm 7 SEADE

Require: Population size: S, Maximum generation: T, Lower and Upper bound: **lb, ub**

Ensure: Optimum: x^*

$P \leftarrow \textbf{popInitial}(S, \textbf{lb, ub})$
$x^* \leftarrow \textbf{best}(P)$
$t \leftarrow 0$
while $t < T$ **do**
    Generate $S - 2$ offspring O with mutation and crossover
    Construct GRNN with all historical samples and GPR with current generation samples
    Estimate elites e_1 and e_2 by random search
    Add elites e_1 and e_2 to offspring O
    Evaluate the offspring O and select survival individuals
    $x^* \leftarrow \textbf{best}(P)$
end while
return x^*

Figure 4.7: The pseudocode of SEADE.

4.4.1 Experiment settings

Experimental environments and implementation: SEADECC-EDDG is programmed with Python 3.11 and implemented in Hokkaido Univerisity'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[149] and the scikit-learn[150] 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 4.2.

Table 4.2: A summary of the algorithms under comparison

Algorithms	Decomposition methods
DECC-RDG	RDG [35]
DECC-RDG2	RDG2 [111]
DECC-ERDG	ERDG [36]
DECC-MDG	MDG [37]
DECC-EDDG	EDDG
SEADECC-EDDG	

In the decomposition, the maximum scale of sub-components MS is 100. In the optimization, all decomposition 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),[36] 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:

$$DA_{sep} = \frac{|sep \cap sep'|}{|sep'|} \tag{4.14}$$

For non-separable decision variables:

$$DA_{nonsep} = \frac{\sum_{g' \in nonsep} |g'|}{\sum_{g' \in nonsep'} |g'|} \tag{4.15}$$

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.

4.4.2 Experimental results

Comparison on decomposition: Table 4.3 summarizes the decomposition results including DA and consumed FEs among RDG2, ERDG, MDG, and EDDG on CEC2013 bench-

mark functions.

Comparison on optimization: Table 4.4 summarizes the optimization results among compared decomposition techniques, and Table 4.5 lists the optimization results between DECC-EDDG, SADECC-EDDG-i, SADECC-EDDG-ii, and SEADECC-EDDG to evaluate the performance of SEADE.

4.5 Discussion

4.5.1 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

Table 4.3: 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%**	50876	-	**100%**	50866	-	**100%**	26980	-	**100%**	**3001**	-	10%	29100
f_{13}	-	99.8%	8345	-	58.0%	15187	-	58.0%	7599	-	**100%**	9440	-	10%	**7583**
f_{14}	-	**100%**	9542	-	**100%**	16150	-	**100%**	8420	-	**100%**	9567	-	10%	**8170**
f_{15}	-	**100%**	6173	-	**100%**	5992	-	**100%**	3996	-	**100%**	**3001**	-	10%	3996

Table 4.4: 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** $\approx$	**8.67e+07**	2.54e+08 $\approx$	1.05e+08	3.40e+08 $\approx$	5.29e+08	3.09e+08 $\approx$	5.36e+08	2.59e+08	2.17e+08
f_2	**1.04e+03** $\approx$	**3.61e+01**	1.06e+03 $\approx$	4.31e+01	1.04e+03 $\approx$	3.87e+01	1.06e+03 $\approx$	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 $\approx$	5.46e+12	1.82e+13 $\approx$	7.42e+12	**1.47e+13** $\approx$	**4.00e+12**	1.63e+13 $\approx$	6.42e+12	1.62e+13	5.31e+12
f_5	4.80e+07 $\approx$	5.02e+06	4.82e+07 $\approx$	5.58e+06	4.62e+07 $\approx$	3.62e+06	**4.47e+07** $\approx$	**4.58e+06**	4.72e+07	3.41e+06
f_6	1.07e+06 $+$	1.29e+03	1.07e+06 $\approx$	1.55e+03	1.07e+06 $\approx$	8.25e+02	**1.07e+06** $\approx$	**1.44e+03**	1.07e+06	1.67e+03
f_7	8.99e+14 $+$	7.90e+14	2.59e+13 $\approx$	1.91e+13	2.53e+13 $\approx$	2.01e+13	**2.08e+13** $\approx$	**1.37e+13**	2.48e+13	1.33e+13
f_8	6.72e+17 $\approx$	1.78e+17	6.71e+17 $\approx$	1.93e+17	6.08e+17 $\approx$	1.79e+17	**6.00e+17** $\approx$	**2.02e+17**	6.48e+17	1.89e+17
f_9	1.06e+09 $\approx$	6.41e+07	1.05e+09 $\approx$	5.99e+07	**1.01e+09** $\approx$	**6.85e+07**	1.02e+09 $\approx$	6.95e+07	1.02e+09	6.87e+07
f_{10}	9.59e+07 $\approx$	3.13e+05	**9.53e+07** $\approx$	**4.06e+05**	9.54e+07 $\approx$	3.08e+05	9.54e+07 $\approx$	2.61e+05	9.58e+07	3.28e+05
f_{11}	2.31e+13 $+$	2.53e+13	**5.92e+11** $\approx$	**2.18e+11**	6.45e+11 $\approx$	2.05e+11	6.26e+11 $\approx$	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
	$+/\approx/-$: 7/7/1		$+/\approx/-$: 4/10/1		$+/\approx/-$: 4/10/1		$+/\approx/-$: 4/10/1			

Table 4.5: 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** $\approx$	**9.67e-03**	2.12e+01 $\approx$	7.75e-03	2.12e+01 $\approx$	1.02e-02	2.12e+01	6.64e-03
f_4	1.62e+13 $\approx$	5.31e+12	1.47e+13 $\approx$	1.67e+12	1.64e+13 $\approx$	2.06e+12	**1.44e+13**	**4.44e+12**
f_5	4.72e+07 $+$	3.41e+06	4.23e+07 $\approx$	5.18e+06	4.83e+07 $+$	3.84e+06	**3.93e+07**	**4.17e+06**
f_6	1.07e+06 $\approx$	1.67e+03	1.07e+06 $\approx$	9.80e+02	1.07e+06 $\approx$	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 $\approx$	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** $\approx$	**3.28e+05**	9.59e+07 $\approx$	2.08e+05	9.60e+07 $\approx$	2.75e+05	9.58e+07	1.29e+05
f_{11}	6.54e+11 $\approx$	1.45e+11	6.80e+11 $\approx$	1.05e+11	6.03e+11 $\approx$	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 $\approx$	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
	$+/\approx/-$: 9/5/1		$+/\approx/-$: 7/7/1		$+/\approx/-$: 8/6/1			

in f_3, our proposed DECC-EDDG has superiority to the compared DECC-RDG, DECC-RDG2, DECC-ERDG, and DECC-MDG.

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 Figure 4.8.

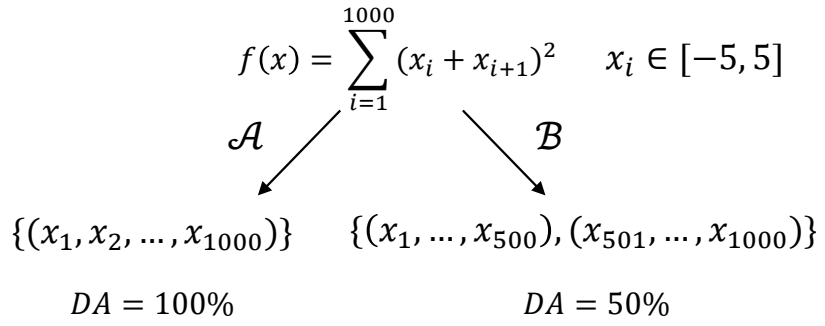


Figure 4.8: A simple example to illustrate the relationship between DA and the optimization performance for the overlapping function.

$f(x) = \sum_{i=1}^{1000}(x_i + x_{i+1})^2$, $x_i \in [-5, 5]$ 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 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}.

In the fully non-separable function f_{15}, EDDG also sacrifices the DA and further decomposes the sub-component with a relatively large scale, thus the deterioration of DA can be observed from Table 4.3 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.

4.5.2 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 4.5. 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 signifi-

cantly 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 4.4, 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.

4.6 Conclusion

This chapter 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.

Chapter 5

Improved vegetation evolution

This chapter presents an improved vegetation evolution optimizer[1]. Section 5.1 presents the motivation of this research, Section 5.2 introduces the original VEGE, Section 5.3 details our proposal: improved VEGE (iVEGE), Section 5.4 presents the experiments on the CEC benchmark functions, and Section 5.5 discusses the performance of iVEGE. Finally, Section 5.6 concludes this chapter.

5.1 Motivation

As one of the newest EC algorithms, vegetation evolution (VEGE) repeatedly simulates the behavior of plants in different periods to balance exploration and exploitation from a fresh perspective.[63] Due to the superior performance of the conventional VEGE compared with some classic EC algorithms, e.g. differential evolution (DE),[49] particle swarm optimization (PSO),[52] and enhanced fireworks algorithm (EFWA),[152] it has attracted widespread attention and several improved versions have been proposed. For example, Yu et al. introduced different mutation strategies into the growth period and maturity period to increase population diversity[153] and proposed multiple different generation strategies to increase

[1]This chapter was previously published as Zhong, R., Peng, F., Zhang, E., Yu, J., & Munetomo, M. Vegetation Evolution with Dynamic Maturity Strategy and Diverse Mutation Strategy for Solving Optimization Problems. Biomimetics 2023, 8, 454. https://doi.org/10.3390/biomimetics8060454.[151]

the global search ability.[154] Besides, they also analyzed the effects of various operations and parameter settings on the performance of the conventional VEGE.[155] Although many pieces of literature have shown the effectiveness of the VEGE, there is still much room for improvement.

The main objective of this chapter is to introduce two new search strategies to further improve the search efficiency of the VEGE and propose an improved VEGE to better balance search efficiency and population diversity. More specifically, the first strategy, i.e., the dynamic maturity strategy, appropriately introduces competition among individuals to generate more potential seed individuals. In this strategy, the concept of the proximate optimal principle (POP)[156] is considered that well-performed solutions have similar structures, thus, we dynamically allocate the computational resources to plants in the seeding operator, where the better individual can generate more seeds and vice versa. The second strategy, i.e., the diverse mutation strategy. We observe the relatively weak capacity of VEGE in getting rid of local optima, and the ensemble of the mutation module can improve the ability to escape from trapped local areas. These ideas motivate us to develop a more efficient variant of VEGE.

In numerical experiments, 30-D and 50-D CEC2013 benchmark functions are employed to evaluate the overall optimization performance of our proposed improved VEGE, and seven engineering problems are adopted to investigate the capacity of our proposal in real-world scenarios. Seven advanced EAs and the conventional VEGE are the competitor algorithms. Besides, we also investigate the contribution of the two strategies to performance improvement and their application scenarios.

5.2 Vegetation Evolution (VEGE)

Since genetic algorithm (GA)[157,158] has sparked a wave of research in the EC community, many well-known EC algorithms have been proposed one after another. Initially, practi-

tioners borrowed biological evolution or the intelligent behavior of animal groups to design new EC algorithms. Then many natural phenomena and human culture have also become sources of inspiration and developed various novel EC algorithms. However, only a few of them get inspiration from plants to propose new algorithms, such as flower pollination algorithm[159] and dandelion optimizer.[160] Fortunately, there has also recently been attention to developing new algorithms inspired by the behavior of different plants. As one of the latest members of this branch, the conventional VEGE simulates the common life cycle of plants derived from observations of different plants rather than simulating the behavior of a particular plant to find the global optimum.

Similar to most heuristic evolutionary algorithms, the conventional VEGE is also population-based and iteratively improves the accuracy of individuals (candidate solutions) to converge to the global optimum.[161,162] Here, a real plant is modeled as an individual, and each individual sequentially goes through two distinct life periods, growth and maturity. Among them, the growing individuals are responsible for local search while the mature individuals are responsible for global search. Thus, the unique contribution of the conventional VEGE is to interactively switch between two different search capabilities to balance exploitation and exploration well.

The general optimization process of the conventional VEGE can be simply summarized as follows. Usually, random initialization is used to generate an initial population consisting of multiple individuals. Eq. (5.1) represents the process of random initialization.

$$P = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ \vdots \\ X_n \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ x_{31} & x_{32} & \cdots & x_{3m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix} \tag{5.1}$$

$$x_{ij} = r_1 \times (UB_j - LB_j) + LB_j$$

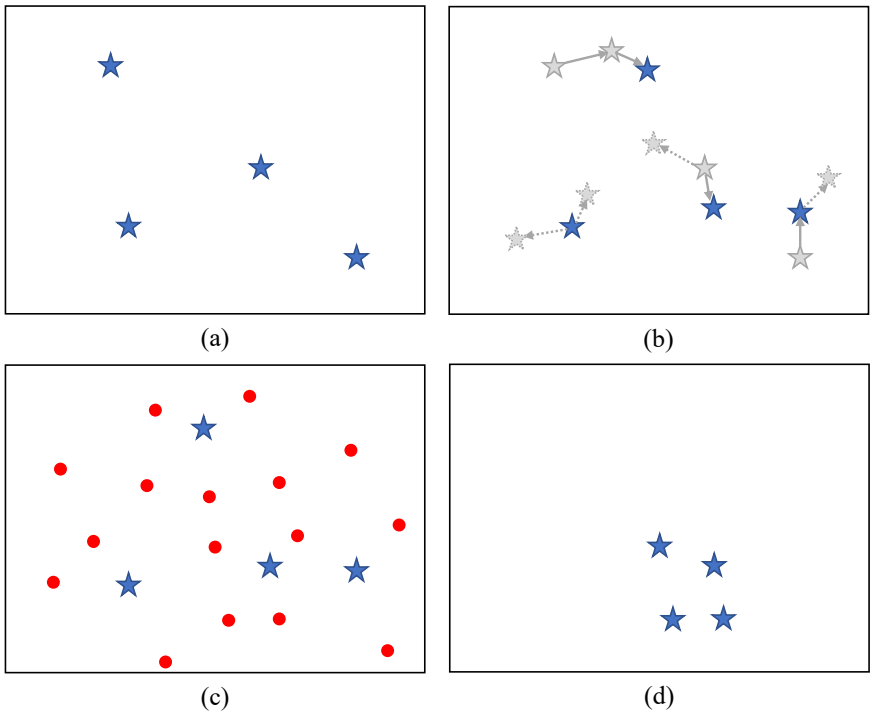


Figure 5.1: The optimization process of the conventional VEGE, and (a) - (d) demonstrate initialization, growth period, maturity period, and selection, respectively. The dashed arrows indicate the growth vector of individuals, and all red dots indicate generated seed individuals.

where LB_j and UB_j are the lower and upper bounds of the j^{th} dimension, n and m are the population size and the dimensionality of the optimization problem, respectively, and r_1 is a random number in (0, 1). These randomly initialized individuals first enter the growth period, and each individual generates only one offspring individual by Eq. (5.2).

$$\vec{\boldsymbol{X}}_i^{t+1} = \vec{\boldsymbol{X}}_i^{t} + \vec{\boldsymbol{GR}} \odot \vec{\boldsymbol{GD}} \tag{5.2}$$

where $\vec{\boldsymbol{GR}}$ is a vector to denote the growth radius of plants in every dimension, and $\vec{\boldsymbol{GD}}$ in (-1, 1) represents the growth direction of plants in every dimension. If the generated offspring individual is better than its parent individual, it will replace the parent individual, otherwise, the parent individual is directly copied to the next generation. After going through a number of growths, i.e., reaching a predefined maximum number of growths, all

individuals enter the maturity period. Then, each individual will generate multiple seed individuals by DE/cur/1-like mutation strategy[49] in Eq. (5.3). Note that this operation will only be performed once, and the individuals that survive to the next generation will enter the growth period again.

$$\vec{\boldsymbol{X}}_{seed} = \vec{\boldsymbol{X}}_i + \vec{\boldsymbol{MS}} \odot (\vec{\boldsymbol{X}}_{r1} - \vec{\boldsymbol{X}}_{r2}) \tag{5.3}$$

where $\vec{\boldsymbol{MS}}$ is the moving scale and set as a random vector in (-2, 2) to simulate the uncertainty from real environments such as the wind, water flow, and animal behaviors. $\vec{\boldsymbol{X}}_{r1}$ and $\vec{\boldsymbol{X}}_{r2}$ are two mutually different individuals from $\vec{\boldsymbol{X}}_i$. Subsequently, all current individuals and seed individuals are mixed together to select the best n (n: population size) individuals to enter the next generation according to their fitness ranking. Finally, these selected individuals will start a new cycle, that is, go through two different periods again until a termination condition is satisfied.

5.3 Proposal: improved VEGE (iVEGE)

The conventional VEGE extracts and simulates some common characteristics of the plant life cycle to gradually update the population, where each individual is treated equally and allocated the same resources, such as the same number of growths and seeds. Actually, the real plant ecosystem is much more complex than what we have observed, and differences exist between different species and even between different individuals of the same species. Here, we introduce two new search strategies, i.e., the dynamic maturity and diverse mutation strategies, into the conventional VEGE to simulate the survival mode of real plants more realistically. To better understand the procedures of our proposal, the flowchart is visualized in Figure 5.2.

Dynamic maturity strategy: Due to different living environments, plants not only

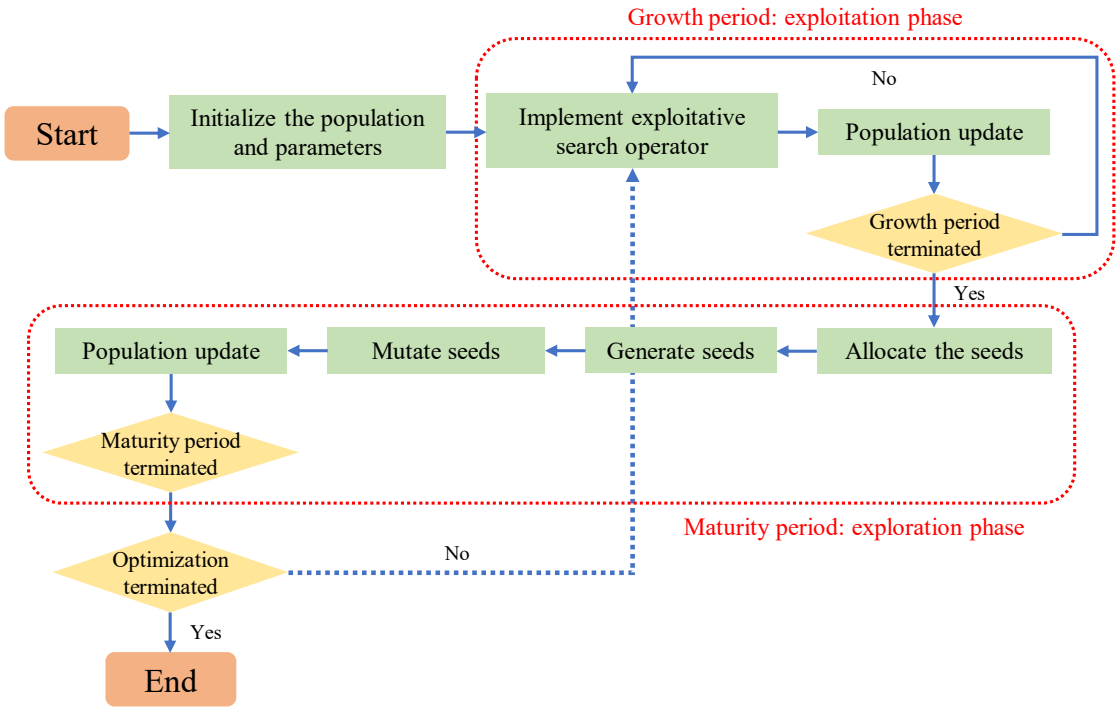


Figure 5.2: The flowchart of our proposal.

need cooperation to ensure the continuation of species but also often face competition for survival resources, such as space and nutrients. The diversity of individuals enables them to adapt to changing environments and not be eliminated by nature. However, the conventional VEGE divides all resources equally and each individual generates exactly the same number of seed individuals. Based on the competitive relationship that exists in nature, we thus introduce the competitive relationship into the conventional VEGE to reasonably allocate the number of seed individuals generated by each individual.

Suppose the total number of seed individuals that can be generated in each generation is SI, we employ the proposed dynamic maturity strategy to assign the number of seed individuals for each individual in the maturity period. First, each individual will generate the same number of k seed individuals, and then the remaining $(SI - k \times n)$ seed individuals will be allocated in a competitive manner based on the fitness of all individuals. Here, we use Eq. (5.4) to determine the probability of each individual being selected and take the roulette wheel to allocate the remaining seed resources. Finally, individuals with better

fitness have a higher probability of generating more seed individuals.

$$p_i = \text{softmax}(\frac{1}{f_i}) = \frac{\exp(\frac{1}{f_i})}{\sum_{j=1}^{n} \exp(\frac{1}{f_i})} \tag{5.4}$$

where p_i is the probability that the i-th individual is selected, and f_i is the fitness of the i-th individual. Figure 5.3 describes the pseudocode of this strategy.

Algorithm Dynamic maturity strategy

Require: Population X.
Ensure: Allocated seeding resources SR.
Obtain the current population X and the fitness value f.
Initialize the seeding resources $SR = [k, k, ..., k]$ for each plant.
Calculate the probability p for each plant.
Calculate the cumulative probability CP.
for $i = 0$ *to* $(SI - k \times n)$ **do**
    Generate a random value r in (0, 1).
    for $j = 1$ *to* n **do**
        if $CP_{j-1} \leq r < CP_j$ **then**
            $SR_{j-1} \leftarrow SR_{j-1} + 1$.
        end if
    end for
end for
Output the allocated seeding resources SR.

Figure 5.3: The pseudocode of dynamic maturity strategy.

Since we are taking the minimization problem as an example, the probability is calculated by taking the inverse of fitness. For maximization problems, fitness can be used directly.

Diverse mutation strategy: Although mutation is one of the important means to increase population diversity, the conventional VEGE did not introduce any mutation strategy when it was originally designed. Later, some practitioners realized this defect and adopted different mutation methods to simulate external mutation and internal mutation for individuals in growth and maturity periods, respectively.[153] However, there are various ways of mutation in nature since the complexity of the ecosystem is far beyond our imagination. We thus introduce multiple different mutation methods only for seed individuals to provide

more diversified potential individuals. In other words, each newly generated seed individual will randomly select one of the following three methods with a uniform probability. Note that these mutations are performed on the seed individuals before they are evaluated.

- genes of a seed individual add a Gaussian noise with a 10% probability. Here, the Gaussian noise is generated by multiplying a standard Gaussian noise by $0.05\times$ (search upper bound - search lower bound).

- the seed individual mutates with its parent by crossover operator with equal probability.

- genes of a seed individual are replaced with a random value in the global space with a probability of 1%.

So far, the two proposed strategies have been explained, and they are both improvements for the maturity period. Figure 5.4 gives the general framework of the conventional VEGE combined with two proposed strategies.

5.4 Numerical experiments

Benchmark functions: Since the 28 benchmark functions from the CEC2013 test suite[163] have most of the common features, we run (the conventional VEGE + two strategies), the conventional VEGE, DE, PSO, DE with self-adaptive populations (DE-SAP), phasor PSO (PPSO), social ski-driver optimization (SSDO), RIME, and snow ablation optimization (SAO) on two different dimensions of these function to analyze the effectiveness of our proposed two strategies. Table 5.1 gives a detailed summary of the CEC2013 benchmark functions including multimodal, asymmetry, non-separable variables, and the global optimum value.

Algorithm Dynamic maturity strategy

Require: Population size: N; Dimension size: D; Maximum iteration: T
Ensure: Optimum: X_{best}

```
Initialize the population randomly.
Evaluate the fitness of all initial individuals.
while optimization is not terminated do
    if all individuals are in the growth period then
        Perform the growth operation using the method of the conventional VEGE.
    else
        Determine the number of seeds.
        All individuals generate seed individuals in turn.
        Use the proposed diverse mutation strategy for generated seed individuals.
        Evaluate all seed individuals after undergoing mutation.
        Mix the population and select the next generation.
    end if
end while
Output the optimum: X_best
```

Figure 5.4: Conventional VEGE with two proposed strategies.

Besides, we investigate the robustness of our proposal in real-world applications. Seven popular engineering optimization problems are employed as test functions, which are listed in Table 5.2 and provided by ENOPPY library.[164] Given the original version of all techniques cannot solve the constrained optimization problems, thus, we equip all EAs with the static penalty function,[165] which is defined by Eq. (5.5)

$$F(R_i) = f(R_i) + w \cdot \sum_{i=1}^{m} (\max(0, g_i(R_i))) \tag{5.5}$$

$F(\cdot)$ is the fitness function, while $f(\cdot)$ and $g_i(\cdot)$ are the objective function and constraint function respectively. w is a constant set to $10e7$ by default in ENOPPY library.

To ensure the fairness of the comparison, we use the number of fitness evaluations to terminate the comparative experiments, and the maximum number is set to $1000 \times dimension$ for CEC2013 28 benchmark functions and 20,000 for engineering problems. Besides, each dimension of each function is run 30 times independently to avoid randomness. Table 5.3 shows the parameter settings of five EC algorithms, and the parameter configuration of the two VEGE algorithms remains exactly the same. In addition, the compared DE, PSO,

Table 5.1: Summary of the CEC2013 suite: Uni.=Unimodal function, Multi.=Multimodal function, Comp.=Composition function

Fun.	Description	Feature	Optimum
f_1	Sphere Function	Uni.	-1400
f_2	Rotated High Conditioned Elliptic Function		-1300
f_3	Rotated Bent Cigar Function		-1200
f_4	Rotated Discus Function		-1100
f_5	Different Powers Function		-1000
f_6	Rotated Rosenbrock's Function	Multi.	-900
f_7	Rotated Schaffers F7 Function		-800
f_8	Rotated Ackley's Function		-700
f_9	Rotated Weierstrass Function		-600
f_{10}	Rotated Griewank's Function		-500
f_{11}	Rastrigin's Function		-400
f_{12}	Rotated Rastrigin's Function		-300
f_{13}	Non-Continuous Rotated Rastrigin's Function		-200
f_{14}	Schwefel's Function		-100
f_{15}	Rotated Schwefel's Function		100
f_{16}	Rotated Katsuura Function		200
f_{17}	Lunacek Bi-Rastrigin Function		300
f_{18}	Rotated Lunacek Bi-Rastrigin Function		400
f_{19}	Expanded Griewank's plus Rosenbrock's Function		500
f_{20}	Expanded Scaffer's F6 Function		600
f_{21}	Composition Function 1 (n=5, Rotated)	Comp.	700
f_{22}	Composition Function 2 (n=3, Unrotated)		800
f_{23}	Composition Function 3 (n=3, Rotated)		900
f_{24}	Composition Function 4 (n=3, Rotated)		1000
f_{25}	Composition Function 5 (n=3, Rotated)		1100
f_{26}	Composition Function 6 (n=5, Rotated)		1200
f_{27}	Composition Function 7 (n=5, Rotated)		1300
f_{28}	Composition Function 8 (n=5, Rotated)		1400

Table 5.2: Summary of seven engineering optimization problems.

Name	Abbr.	Dim.	# of constraints
Welded Beam Problem	WBP	4	7
Compression Spring Problem	CSP	4	4
Speed Reducer Problem	SRD	7	11
Three Bar Truss Problem	TBTD	2	3
Cantilever Beam Problem	CBD	5	1
Tubular Column Problem	TCD	2	6
Corrugated Bulkhead Problem	CBHD	4	6

DE-SAP, PPSO, and SSDO are provided by MEALPY library.[166]

Experimental results: Tables 5.4, 5.5, 5.6, and 5.7 give the experimental and statistical results of numerical experiments on 30-D and 50-D CEC2013 benchmark functions. We

Table 5.3: The parameter settings of five comparison algorithms.

MAs	Parameters	Value
DE [49]	Population size	100
	Scaling factor F	1
	Crossover rate Cr	0.9
	Mutation strategy	DE/cur-to-rand/1/bin
PSO [52]	Population size	100
	Inertia factor w	1
	Coefficients c_1 and c_2	2.05
	Max. and min. speed	2, -2
DE-SAP [167]	Population size	100
	Encoding method w	absolute encoding (Abs)
PPSO [168]	Population size	100
SSDO [169]	Population size	100
VEGE [63]	Population size	10
	Growth cycle GC	6
	Growth radius GR	2
	Growth direction GD	a random number in [-1,1]
	Total # of seed individuals	60
	Moving scaling MS	a random number in [-2,2]
RIME [4]	Population size	100
	parameter w	5
SAO [90]	Population size	100

applied the Kruskal-Wallis test and the Holm multiple comparison test to check whether there was a significant difference between them at the termination of the competitor algorithms. $+$, $\approx$, 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. Due to the limitation of space, optimization convergence curves of representative functions are provided in Figs. 5.5 and 5.6. Table 5.8 provides the detailed optimization results on seven engineering problems, and convergence curves of engineering problems are presented in Figure 5.7. Table 5.9 and 5.10 summarize the ablation experimental results

to investigate the respective contributions of our proposed two strategies in performance improvement. For the sake of simplicity, we name VEGE + dynamic maturity strategy as VEGE-i, VEGE + diverse mutation strategy as VEGE-ii, and VEGE + two strategies as the Proposal.

5.5 Discussion

We first want to discuss the new benefits of the two strategies. The first strategy, i.e., the dynamic maturity strategy, takes both fixed allocation and dynamic allocation into account so as to allocate the limited number of seed individuals more reasonably. When k is set to 0, the number of seed individuals that can be generated by each individual is dynamically allocated according to the fitness of individuals. When k is set to $\frac{SI}{n}$, the seed individuals are assigned in the same way as the conventional VEGE. Thus, the allocation method of the conventional VEGE can be seen as a special case of the proposed strategy. We can also adjust the value of k to assign different proportions to the two allocation methods, which means that the strategy can flexibly handle various optimization problems with different characteristics. Besides, the strategy can give better individuals more opportunities to search space while ensuring that poor individuals will not lose the opportunity to continue searching.

The second strategy, i.e., the diverse mutation strategy, provides a variety of different mutations to increase the diversity of the population, and each mutation method modifies the genes of seed individuals with different probabilities. Moreover, the seed individuals generated from the same individual have the opportunity to perform different mutation methods, which can explore local areas more diversified. Especially when the population stagnates, it is helpful to escape from trapped local areas. Since these mutation operations are performed before fitness evaluation, the second strategy does not add additional fitness consumption. Meanwhile, the CPU consumption resulting from both proposed strategies is

Table 5.4: Experimental and statistical results on 30-D CEC2013 benchmark functions. $f_1 - f_5$: Unimodal functions; $f_6 - f_{20}$: Basic Multimodal functions; $f_{21} - f_{28}$: Composition functions (Proposal: VEGE + two proposed strategies).

Func.		DE	PSO	DE-SAP	PPSO	SSDO	RIME	SAO	VEGE	Proposal
f_1	mean	2.44e+04 +	2.98e+04 +	6.71e+04 +	-2.71e+02 +	4.62e+04 +	-1.38e+03 +	9.49e+03 +	-1.40e+03 +	**-1.40e+03**
	std	2.97e+03	1.14e+04	8.84e+03	6.02e+02	3.06e+03	6.43e+00	4.06e+03	8.37e-01	**1.63e-01**
f_2	mean	2.79e+08 +	3.82e+08 +	1.25e+09 +	5.84e+07 +	6.36e+08 +	2.62e+07 +	1.04e+08 +	5.43e+06 ≈	**5.35e+06**
	std	8.22e+07	2.54e+08	4.14e+08	2.56e+07	1.25e+08	1.15e+07	4.91e+07	2.36e+06	**2.50e+06**
f_3	mean	9.14e+10 +	2.19e+14 +	3.97e+17 +	4.97e+10 +	7.34e+15 +	1.27e+08 +	8.36e+10 +	2.98e+07 +	**5.73e+06**
	std	1.12e+10	8.96e+14	1.64e+18	2.91e+10	1.11e+16	1.52e+08	5.66e+10	4.89e+07	**9.05e+06**
f_4	mean	1.57e+05 +	1.95e+05 +	1.18e+05 +	7.54e+04 +	6.27e+04 +	3.29e+04 +	5.73e+04 +	1.54e+04 ≈	**1.39e+04**
	std	1.85e+04	6.80e+04	2.32e+04	1.61e+04	2.21e+03	1.03e+04	6.35e+03	6.52e+03	**5.15e+03**
f_5	mean	6.44e+03 +	5.19e+04 +	7.78e+04 +	6.52e+02 +	4.23e+04 +	-8.18e+02 +	7.37e+03 +	-9.97e+02 +	**-9.99e+02**
	std	1.23e+03	1.79e+04	2.44e+04	1.91e+03	8.02e+03	5.99e+01	4.18e+03	4.32e+00	**2.07e-01**
f_6	mean	1.12e+03 +	4.26e+03 +	1.35e+04 +	-6.05e+02 +	7.16e+03 +	-8.19e+02 +	-2.77e+02 +	-8.32e+02 ≈	**-8.40e+02**
	std	4.32e+02	3.41e+03	4.60e+03	1.20e+02	1.04e+03	2.88e+01	2.39e+02	3.04e+01	**3.62e+01**
f_7	mean	2.10e+05 +	3.38e+06 +	1.83e+08 +	3.58e+06 +	4.37e+07 +	**1.08e+05** −	2.14e+05 +	3.42e+06 +	1.40e+05
	std	1.32e+04	4.68e+06	1.85e+08	5.59e+06	2.89e+07	**1.66e+04**	1.74e+05	1.24e+07	4.08e+04
f_8	mean	**-6.79e+02** ≈	-6.79e+02 ≈	-6.79e+02 ≈	-6.79e+02 ≈	-6.79e+02 ≈	-6.79e+02 ≈	-6.79e+02 ≈	-6.79e+02 ≈	-6.79e+02
	std	**6.05e-02**	4.48e-02	5.20e-02	7.12e-02	6.03e-02	6.93e-02	8.65e-02	4.44e-02	4.68e-02
f_9	mean	-5.59e+02 +	-5.62e+02 +	-5.57e+02 +	-5.61e+02 +	-5.58e+02 +	**-5.75e+02** −	-5.70e+02 ≈	-5.68e+02 ≈	-5.71e+02
	std	1.44e+00	3.92e+00	1.30e+00	2.58e+00	1.01e+00	**5.03e+00**	3.17e+00	3.98e+00	4.13e+00
f_{10}	mean	2.74e+03 +	4.82e+03 +	9.35e+03 +	-1.02e+01 +	6.07e+03 +	-4.46e+02 +	1.07e+03 +	-4.96e+02 +	**-4.97e+02**
	std	5.09e+02	2.05e+03	1.89e+03	2.76e+02	6.06e+02	2.19e+01	4.90e+02	6.48e-01	**6.01e-01**
f_{11}	mean	1.04e+02 +	3.71e+02 +	6.69e+02 +	5.85e+01 +	3.94e+02 +	-2.88e+02 +	-4.05e+01 +	1.03e+02 +	**-3.84e+02**
	std	3.34e+01	1.66e+02	1.60e+02	9.32e+01	4.50e+01	1.77e+01	7.24e+01	1.03e+02	**4.27e+00**
f_{12}	mean	2.65e+02 +	4.39e+02 +	7.95e+02 +	1.89e+02 +	4.47e+02 +	**-1.55e+02** −	6.56e+01 ≈	1.54e+02 +	5.69e+01
	std	4.69e+01	1.67e+02	1.27e+02	1.09e+02	3.80e+01	**3.68e+01**	7.28e+01	1.10e+02	9.84e+01
f_{13}	mean	3.21e+02 +	5.17e+02 +	7.94e+02 +	3.62e+02 +	5.03e+02 +	**1.06e+01** −	1.66e+02 ≈	3.70e+02 +	1.91e+02
	std	3.82e+01	1.60e+02	1.62e+02	9.39e+01	4.31e+01	**2.59e+01**	6.33e+01	1.24e+02	7.60e+01
f_{14}	mean	6.03e+03 +	8.31e+03 +	8.35e+03 +	5.18e+03 +	8.37e+03 +	2.42e+03 +	5.32e+03 +	4.59e+03 +	**2.49e+02**
	std	3.85e+02	2.80e+02	2.86e+02	8.16e+02	2.19e+02	4.21e+02	7.98e+02	4.93e+02	**1.66e+02**

Table 5.5: Experimental and statistical results on 30-D CEC2013 benchmark functions (Continued).

Func.		DE	PSO	DE-SAP	PPSO	SSDO	RIME	SAO	VEGE	Proposal
f_{15}	mean	8.11e+03 +	8.57e+03 +	8.46e+03 +	6.32e+03 +	7.82e+03 +	5.08e+03 +	5.80e+03 +	4.52e+03 ≈	**4.20e+03**
	std	3.64e+02	4.61e+02	3.11e+02	8.14e+02	2.94e+02	6.66e+02	5.23e+02	4.96e+02	**5.88e+02**
f_{16}	mean	2.03e+02 +	2.03e+02 +	2.03e+02 +	2.02e+02 +	2.03e+02 +	2.02e+02 +	**2.01e+02** ≈	2.02e+02 +	2.01e+02
	std	3.23e-01	3.48e-01	3.15e-01	5.98e-01	3.54e-01	5.34e-01	**3.59e-01**	4.48e-01	3.57e-01
f_{17}	mean	1.75e+03 +	1.50e+03 +	1.63e+03 +	1.00e+03 +	1.18e+03 +	5.03e+02 +	7.43e+02 +	1.15e+03 +	**3.54e+02**
	std	1.68e+02	2.41e+02	2.53e+02	1.01e+02	3.80e+01	2.76e+01	8.99e+01	1.94e+02	**5.91e+00**
f_{18}	mean	1.71e+03 +	2.04e+03 +	3.31e+03 +	1.70e+03 +	2.60e+03 +	**6.28e+02** −	1.23e+03 +	1.92e+03 +	7.08e+02
	std	1.23e+02	4.13e+02	3.50e+02	2.82e+02	1.07e+02	**3.70e+01**	1.94e+02	3.64e+02	8.71e+01
f_{19}	mean	1.13e+05 +	2.93e+05 +	2.95e+06 +	2.72e+03 +	6.88e+05 +	**5.19e+02** ≈	8.66e+03 +	5.32e+02 +	5.19e+02
	std	5.39e+04	3.10e+05	1.44e+06	5.16e+03	2.20e+05	**3.79e+00**	1.19e+04	5.45e+00	6.41e+00
f_{20}	mean	6.14e+02 ≈	6.15e+02 +	6.15e+02 +	6.15e+02 +	6.15e+02 +	**6.13e+02** −	6.15e+02 +	6.15e+02 +	6.14e+02
	std	1.94e-01	4.84e-01	2.26e-07	1.81e-01	1.95e-04	**7.51e-01**	4.62e-01	1.96e-01	7.55e-01
f_{21}	mean	4.10e+03 +	3.71e+03 +	4.50e+03 +	2.05e+03 +	3.20e+03 +	**1.67e+03** −	2.59e+03 +	1.83e+03 +	1.69e+03
	std	1.88e+02	6.52e+02	5.64e+02	1.24e+02	8.10e+01	**2.63e+02**	1.85e+02	2.45e+02	2.69e+02
f_{22}	mean	7.31e+03 +	9.77e+03 +	9.86e+03 +	6.56e+03 +	1.01e+04 +	3.78e+03 +	7.44e+03 +	6.26e+03 +	**1.21e+03**
	std	3.42e+02	5.71e+02	4.61e+02	8.86e+02	2.56e+02	4.96e+02	9.26e+02	7.09e+02	**1.40e+02**
f_{23}	mean	9.06e+03 +	9.52e+03 +	9.69e+03 +	7.59e+03 +	9.62e+03 +	6.14e+03 ≈	7.86e+03 +	5.86e+03 ≈	**5.85e+03**
	std	3.24e+02	3.09e+02	3.76e+02	7.65e+02	2.87e+02	9.04e+02	1.11e+03	6.87e+02	**7.33e+02**
f_{24}	mean	1.31e+03 +	1.31e+03 +	1.36e+03 +	1.32e+03 +	1.36e+03 +	**1.26e+03** −	1.29e+03 +	1.30e+03 +	1.28e+03
	std	3.56e+00	8.40e+00	1.77e+01	7.86e+00	1.35e+01	**1.02e+01**	1.28e+01	1.34e+01	1.12e+01
f_{25}	mean	1.41e+03 +	1.42e+03 +	1.45e+03 +	1.43e+03 +	1.47e+03 +	**1.38e+03** −	1.42e+03 +	1.43e+03 +	1.40e+03
	std	6.05e+00	8.92e+00	9.25e+00	1.34e+01	1.52e+01	**8.43e+00**	1.31e+01	1.42e+01	1.62e+01
f_{26}	mean	1.52e+03 ≈	1.58e+03 +	1.61e+03 +	1.59e+03 +	1.60e+03 +	1.52e+03 ≈	1.56e+03 +	1.53e+03 ≈	**1.51e+03**
	std	7.56e+01	4.77e+01	3.51e+01	5.02e+01	3.29e+01	7.11e+01	5.99e+01	8.66e+01	**8.33e+01**
f_{27}	mean	2.71e+03 +	2.72e+03 +	3.11e+03 +	2.75e+03 +	3.21e+03 +	**2.26e+03** −	2.55e+03 +	2.67e+03 +	2.43e+03
	std	2.90e+01	8.05e+01	1.33e+02	7.33e+01	1.20e+02	**1.00e+02**	9.49e+01	1.12e+02	8.71e+01
f_{28}	mean	4.91e+03 +	6.10e+03 +	7.41e+03 +	5.04e+03 +	5.70e+03 +	**2.32e+03** −	4.45e+03 +	4.93e+03 +	3.64e+03
	std	2.10e+02	7.46e+02	6.90e+02	5.27e+02	1.83e+02	**4.34e+02**	3.91e+02	4.48e+02	1.22e+03
		+/≈/−: 25/3/0	+/≈/−: 27/1/0	+/≈/−: 27/1/0	+/≈/−: 27/1/0	+/≈/−: 27/1/0	+/≈/−: 13/4/11	+/≈/−: 23/5/0	+/≈/−: 20/8/0	

Table 5.6: Experimental and statistical results on 50-D CEC2013 benchmark functions.

Func.		DE	PSO	DE-SAP	PPSO	SSDO	RIME	SAO	VEGE	Proposal
f_1	mean	7.75e+04 +	6.46e+04 +	9.55e+04 +	1.34e+03 +	7.03e+04 +	-1.28e+03 +	2.36e+04 +	-1.39e+03 +	**-1.40e+03**
	std	7.23e+03	1.47e+04	1.11e+04	9.91e+02	3.31e+03	2.69e+01	5.14e+03	2.03e+00	**1.07e+00**
f_2	mean	1.11e+09 +	1.63e+09 +	3.68e+09 +	1.47e+08 +	1.83e+09 +	7.47e+07 +	2.06e+08 +	**9.29e+06** −	1.15e+07
	std	1.34e+08	8.33e+08	8.69e+08	4.46e+07	3.93e+08	1.95e+07	6.91e+07	**2.54e+06**	2.89e+06
f_3	mean	2.92e+11 +	9.75e+14 +	5.17e+17 +	6.95e+10 +	4.29e+13 +	3.33e+09 +	1.19e+11 +	2.21e+08 +	**7.26e+07**
	std	2.16e+10	3.15e+15	1.79e+18	2.88e+10	4.17e+13	4.83e+09	4.18e+10	4.31e+08	**1.70e+08**
f_4	mean	2.76e+05 +	2.95e+05 +	1.85e+05 +	1.20e+05 +	8.59e+04 +	6.49e+04 +	8.72e+04 +	**1.06e+04** −	1.47e+04
	std	2.24e+04	9.53e+04	5.35e+04	1.95e+04	3.71e+03	1.35e+04	6.38e+03	**4.20e+03**	4.77e+03
f_5	mean	4.19e+04 +	1.06e+05 +	9.11e+04 +	1.51e+03 +	3.70e+04 +	-4.03e+02 +	9.54e+03 +	-9.95e+02 +	**-9.98e+02**
	std	4.96e+03	6.02e+04	3.08e+04	1.90e+03	6.90e+03	1.34e+02	3.43e+03	1.36e+00	**6.83e-01**
f_6	mean	6.88e+03 +	5.05e+03 +	1.28e+04 +	-4.65e+02 +	6.72e+03 +	-7.86e+02 ≈	4.62e+02 +	-8.04e+02 ≈	**-8.11e+02**
	std	9.37e+02	2.49e+03	3.80e+03	1.17e+02	5.61e+02	5.00e+01	4.09e+02	4.49e+01	**4.28e+01**
f_7	mean	8.19e+05 +	2.86e+07 +	3.61e+08 +	4.97e+06 +	1.02e+07 +	**4.38e+05** ≈	5.85e+05 ≈	3.95e+06 +	4.69e+05
	std	7.36e+04	4.76e+07	3.59e+08	7.04e+06	3.79e+06	**6.13e+04**	2.37e+05	8.69e+06	8.88e+04
f_8	mean	-6.79e+02 ≈	-6.79e+02 ≈	-6.79e+02 ≈	-6.79e+02 ≈	-6.79e+02 ≈	**-6.79e+02** ≈	-6.79e+02 ≈	-6.79e+02 ≈	-6.79e+02
	std	3.54e-02	3.53e-02	4.84e-02	6.62e-02	3.84e-02	**5.28e-02**	5.53e-02	4.25e-02	3.67e-02
f_9	mean	-5.24e+02 +	-5.30e+02 +	-5.22e+02 +	-5.28e+02 +	-5.23e+02 +	**-5.47e+02** −	-5.40e+02 +	-5.38e+02 +	-5.43e+02
	std	1.23e+00	4.42e+00	1.91e+00	3.59e+00	1.33e+00	**5.63e+00**	3.54e+00	4.62e+00	5.21e+00
f_{10}	mean	8.37e+03 +	9.74e+03 +	1.67e+04 +	5.27e+02 +	1.00e+04 +	-2.63e+02 +	2.51e+03 +	-4.88e+02 ≈	**-4.89e+02**
	std	1.32e+03	2.64e+03	2.67e+03	3.40e+02	6.53e+02	7.04e+01	7.06e+02	1.88e+00	**2.29e+00**
f_{11}	mean	9.78e+02 +	9.69e+02 +	1.08e+03 +	4.05e+02 +	7.17e+02 +	-7.11e+01 +	2.72e+02 +	5.33e+02 +	**-3.66e+02**
	std	7.25e+01	2.36e+02	2.16e+02	1.12e+02	4.65e+01	4.88e+01	7.64e+01	1.51e+02	**7.80e+00**
f_{12}	mean	1.10e+03 +	9.74e+02 +	1.31e+03 +	5.70e+02 +	8.89e+02 +	**8.24e+01** −	3.89e+02 ≈	5.69e+02 +	4.08e+02
	std	1.07e+02	2.32e+02	1.68e+02	1.08e+02	4.11e+01	**6.73e+01**	9.51e+01	1.55e+02	1.30e+02
f_{13}	mean	1.18e+03 +	1.02e+03 +	1.31e+03 +	7.92e+02 +	8.97e+02 +	**2.89e+02** −	5.05e+02 −	7.81e+02 +	5.80e+02
	std	1.08e+02	1.87e+02	1.61e+02	8.04e+01	3.70e+01	**7.01e+01**	8.49e+01	1.59e+02	1.35e+02
f_{14}	mean	1.12e+04 +	1.53e+04 +	1.49e+04 +	1.02e+04 +	1.52e+04 +	6.11e+03 +	1.08e+04 +	8.28e+03 +	**3.27e+02**
	std	4.80e+02	7.13e+02	5.68e+02	1.24e+03	4.71e+02	7.76e+02	1.21e+03	9.40e+02	**1.62e+02**

Table 5.7: Experimental and statistical results on 50-D CEC2013 benchmark functions (Continued).

Func.		DE	PSO	DE-SAP	PPSO	SSDO	RIME	SAO	VEGE	Proposal
f_{15}	mean	1.50e+04 +	1.56e+04 +	1.58e+04 +	1.27e+04 +	1.51e+04 +	1.08e+04 +	1.23e+04 +	8.77e+03 ≈	**8.67e+03**
	std	3.75e+02	4.18e+02	3.84e+02	1.39e+03	5.10e+02	1.10e+03	9.49e+02	7.62e+02	**8.16e+02**
f_{16}	mean	2.04e+02 +	2.04e+02 +	2.04e+02 +	2.03e+02 +	2.04e+02 +	2.03e+02 +	**2.02e+02** ≈	2.03e+02 +	2.02e+02
	std	3.34e-01	3.08e-01	5.68e-01	7.99e-01	2.64e-01	6.25e-01	**4.24e-01**	5.45e-01	6.19e-01
f_{17}	mean	4.36e+03 +	2.69e+03 +	2.34e+03 +	1.60e+03 +	1.69e+03 +	8.28e+02 +	1.19e+03 +	2.12e+03 +	**4.09e+02**
	std	4.57e+02	5.31e+02	2.83e+02	1.54e+02	2.83e+01	6.32e+01	1.05e+02	3.14e+02	**9.50e+00**
f_{18}	mean	3.99e+03 +	4.00e+03 +	4.75e+03 +	3.02e+03 +	3.78e+03 +	**9.54e+02** ≈	2.00e+03 +	3.45e+03 +	1.02e+03
	std	2.33e+02	6.43e+02	4.75e+02	2.93e+02	1.14e+02	**8.31e+01**	2.36e+02	5.13e+02	1.25e+02
f_{19}	mean	2.71e+06 +	6.27e+05 +	2.07e+06 +	4.33e+03 +	3.27e+05 +	5.56e+02 +	1.59e+04 +	5.68e+02 +	**5.39e+02**
	std	8.77e+05	7.00e+05	1.62e+06	3.63e+03	5.09e+04	9.34e+00	1.08e+04	1.10e+01	**6.63e+00**
f_{20}	mean	6.24e+02 ≈	6.25e+02 +	6.25e+02 +	6.25e+02 +	6.25e+02 +	**6.24e+02** ≈	6.24e+02 ≈	6.24e+02 ≈	6.24e+02
	std	2.35e-01	2.20e-01	1.91e-06	7.82e-02	6.78e-02	**6.35e-01**	3.66e-01	3.66e-01	5.46e-01
f_{21}	mean	7.93e+03 +	6.90e+03 +	6.36e+03 +	1.78e+03 +	4.58e+03 +	1.36e+03 +	3.48e+03 +	1.19e+03 +	**1.14e+03**
	std	5.48e+02	1.43e+03	8.57e+02	3.50e+02	9.06e+01	4.11e+02	2.71e+02	2.17e+02	**3.17e+00**
f_{22}	mean	1.29e+04 +	1.73e+04 +	1.72e+04 +	1.30e+04 +	1.75e+04 +	7.88e+03 +	1.43e+04 +	1.12e+04 +	**1.31e+03**
	std	5.04e+02	6.23e+02	4.88e+02	1.41e+03	3.98e+02	9.87e+02	1.43e+03	1.17e+03	**1.33e+02**
f_{23}	mean	1.61e+04 +	1.67e+04 +	1.73e+04 +	1.48e+04 +	1.71e+04 +	1.23e+04 +	1.50e+04 +	1.11e+04 ≈	**1.09e+04**
	std	3.57e+02	7.21e+02	4.83e+02	1.21e+03	4.20e+02	1.17e+03	1.42e+03	1.06e+03	**9.74e+02**
f_{24}	mean	1.40e+03 +	1.41e+03 +	1.67e+03 +	1.44e+03 +	1.62e+03 +	**1.33e+03** −	1.41e+03 +	1.41e+03 +	1.36e+03
	std	5.83e+00	1.68e+01	1.38e+02	3.20e+01	7.81e+01	**1.57e+01**	2.44e+01	1.29e+01	1.47e+01
f_{25}	mean	1.51e+03 ≈	1.52e+03 +	1.59e+03 +	1.54e+03 +	1.62e+03 +	**1.46e+03** −	1.53e+03 +	1.58e+03 +	1.50e+03
	std	8.24e+00	1.63e+01	2.61e+01	1.66e+01	2.50e+01	**1.44e+01**	1.93e+01	2.38e+01	1.84e+01
f_{26}	mean	1.69e+03 +	1.68e+03 +	1.74e+03 +	1.70e+03 +	1.73e+03 +	**1.62e+03** −	1.64e+03 +	1.67e+03 +	1.64e+03
	std	5.49e+00	1.17e+01	4.18e+01	1.41e+01	7.34e+00	**4.17e+01**	6.64e+01	1.30e+01	1.10e+01
f_{27}	mean	3.63e+03 +	3.64e+03 +	4.77e+03 +	3.82e+03 +	4.52e+03 +	**2.86e+03** −	3.47e+03 +	3.57e+03 +	3.25e+03
	std	5.36e+01	1.31e+02	5.24e+02	2.13e+02	1.93e+02	**1.35e+02**	1.77e+02	2.51e+02	1.28e+02
f_{28}	mean	9.18e+03 +	1.27e+04 +	1.33e+04 +	9.65e+03 +	9.95e+03 +	**3.21e+03** ≈	7.66e+03 +	8.88e+03 +	4.93e+03
	std	4.78e+02	1.74e+03	1.57e+03	9.01e+02	3.26e+02	**9.33e+02**	5.01e+02	1.16e+03	2.08e+03
		+/≈/−: 25/3/0	+/≈/−: 27/1/0	+/≈/−: 27/1/0	+/≈/−: 27/1/0	+/≈/−: 27/1/0	+/≈/−: 15/6/7	+/≈/−:v 22/5/1	+/≈/−: 20/6/2	

Table 5.8: Experimental and statistical results on seven engineering problems.

Func.		DE	PSO	DE-SAP	PPSO	SSDO	RIME	SAO	VEGE	Proposal
WBP	mean	2.0480e+00 +	2.0480e+00 +	2.3833e+00 +	2.2624e+00 +	1.4229e+05 +	1.9252e+00 +	1.9104e+00 +	1.7238e+00 +	**1.6916e+00**
	std	2.0936e-01	2.0936e-01	3.4810e-01	5.3722e-01	7.6623e+05	2.2225e-01	2.5516e-01	**3.1670e-02**	4.3967e-02
	worst	2.5772e+00	2.5772e+00	3.1889e+00	3.8419e+00	4.2686e+06	2.8182e+00	2.8941e+00	**1.8171e+00**	1.9271e+00
	best	1.7887e+00	1.7887e+00	1.8893e+00	1.7229e+00	2.6867e+00	1.7036e+00	1.6829e+00	1.6848e+00	**1.6826e+00**
CSP	mean	6.0775e-03 +	6.0775e-03 +	6.5158e-03 +	6.0990e-03 ≈	6.0818e-03 +	6.0785e-03 +	6.1013e-03 +	6.0871e-03 +	**6.0761e-03**
	std	1.5168e-06	1.5168e-06	5.2997e-04	1.2300e-04	5.9274e-06	4.0493e-06	1.2203e-04	1.2788e-05	**1.3320e-08**
	worst	6.0811e-03	6.0811e-03	8.5403e-03	6.7614e-03	6.0986e-03	6.0978e-03	6.7582e-03	6.1218e-03	**6.0762e-03**
	best	**6.0761e-03**	**6.0761e-03**	6.0777e-03	**6.0761e-03**	6.0763e-03	6.0762e-03	**6.0761e-03**	**6.0761e-03**	**6.0761e-03**
SRD	mean	3.0695e+03 +	3.0695e+03 +	3.2475e+03 +	3.0741e+03 +	1.2766e+06 +	2.9963e+03 +	2.9990e+03 +	3.0450e+03 +	**2.9869e+03**
	std	5.3510e+01	5.3510e+01	4.3277e+01	9.4483e+01	1.7862e+06	7.0867e+00	9.1124e+00	2.7661e+01	**1.0080e-03**
	worst	3.2223e+03	3.2223e+03	3.3639e+03	3.3639e+03	6.1677e+06	3.0180e+03	3.0377e+03	3.1019e+03	**2.9869e+03**
	best	3.0298e+03	3.0298e+03	3.1683e+03	2.9879e+03	3.2053e+03	2.9876e+03	2.9886e+03	3.0000e+03	**2.9869e+03**
TBTD	mean	2.6419e+02 +	2.6419e+02 +	2.6864e+02 +	2.6413e+02 +	2.6999e+02 +	2.6413e+02 +	2.6423e+02 +	**2.6390e+02** ≈	2.6390e+02
	std	1.8476e-01	1.8476e-01	2.0998e+00	3.8377e-01	5.0347e+00	3.8677e-01	1.1078e+00	**2.4545e-05**	2.2217e-03
	worst	2.6475e+02	2.6475e+02	2.7107e+02	2.6522e+02	2.8284e+02	2.6553e+02	2.6985e+02	**2.6390e+02**	2.6391e+02
	best	2.6395e+02	2.6395e+02	2.6419e+02	**2.6390e+02**	2.6400e+02	**2.6390e+02**	**2.6390e+02**	**2.6390e+02**	**2.6390e+02**
CBD	mean	2.7914e+00 +	2.7914e+00 +	4.0830e+00 +	2.8157e+00 +	2.5804e+00 +	1.3574e+00 +	1.3415e+00 +	1.3432e+00 +	**1.3402e+00**
	std	5.7027e-01	5.7027e-01	1.8919e+00	1.1259e+00	6.3506e-01	1.5191e-02	1.7858e-03	1.8963e-03	**4.4879e-04**
	worst	4.2211e+00	4.2211e+00	8.3554e+00	5.5371e+00	3.8432e+00	1.4100e+00	1.3480e+00	1.3474e+00	**1.3423e+00**
	best	2.1447e+00	2.1447e+00	1.5797e+00	1.4229e+00	1.3684e+00	1.3409e+00	1.3401e+00	1.3405e+00	**1.3400e+00**
TCD	mean	3.0542e+01 +	3.0542e+01 +	3.0904e+01 +	3.0254e+01 +	3.1549e+01 +	3.0328e+01 +	3.0153e+01 +	**3.0150e+01** ≈	3.0152e+01
	std	2.3954e-01	2.3954e-01	4.4253e-01	3.8837e-01	6.3834e-01	2.0913e-01	1.0563e-02	**5.3976e-04**	1.1610e-02
	worst	3.1181e+01	3.1181e+01	3.2113e+01	3.2213e+01	3.2711e+01	3.0857e+01	3.0208e+01	**3.0153e+01**	3.0214e+01
	best	3.0192e+01	3.0192e+01	3.0287e+01	**3.0150e+01**	3.0289e+01	3.0151e+01	**3.0150e+01**	**3.0150e+01**	**3.0150e+01**
CBHD	mean	7.8378e+00 +	7.8378e+00 +	8.5143e+00 +	7.2506e+00 +	1.0699e+01 +	6.8542e+00 +	6.9393e+00 +	6.8762e+00 +	**6.8430e+00**
	std	7.1741e-01	7.1741e-01	6.6499e-01	4.0215e-01	1.2514e+00	7.7909e-03	1.5303e-01	4.5820e-02	**3.0522e-08**
	worst	1.0352e+01	1.0352e+01	1.0069e+01	8.4715e+00	1.3665e+01	6.8709e+00	7.5109e+00	7.0912e+00	**6.8430e+00**
	best	7.0410e+00	7.0410e+00	7.7228e+00	6.8774e+00	8.2201e+00	6.8444e+00	6.8431e+00	6.8451e+00	**6.8430e+00**
		+/≈/−: 7/0/0	+/≈/−: 7/0/0	+/≈/−: 7/0/0	+/≈/−: 6/1/0	+/≈/−: 7/0/0	+/≈/−: 7/0/0	+/≈/−: 7/0/0	+/≈/−: 5/2/0	

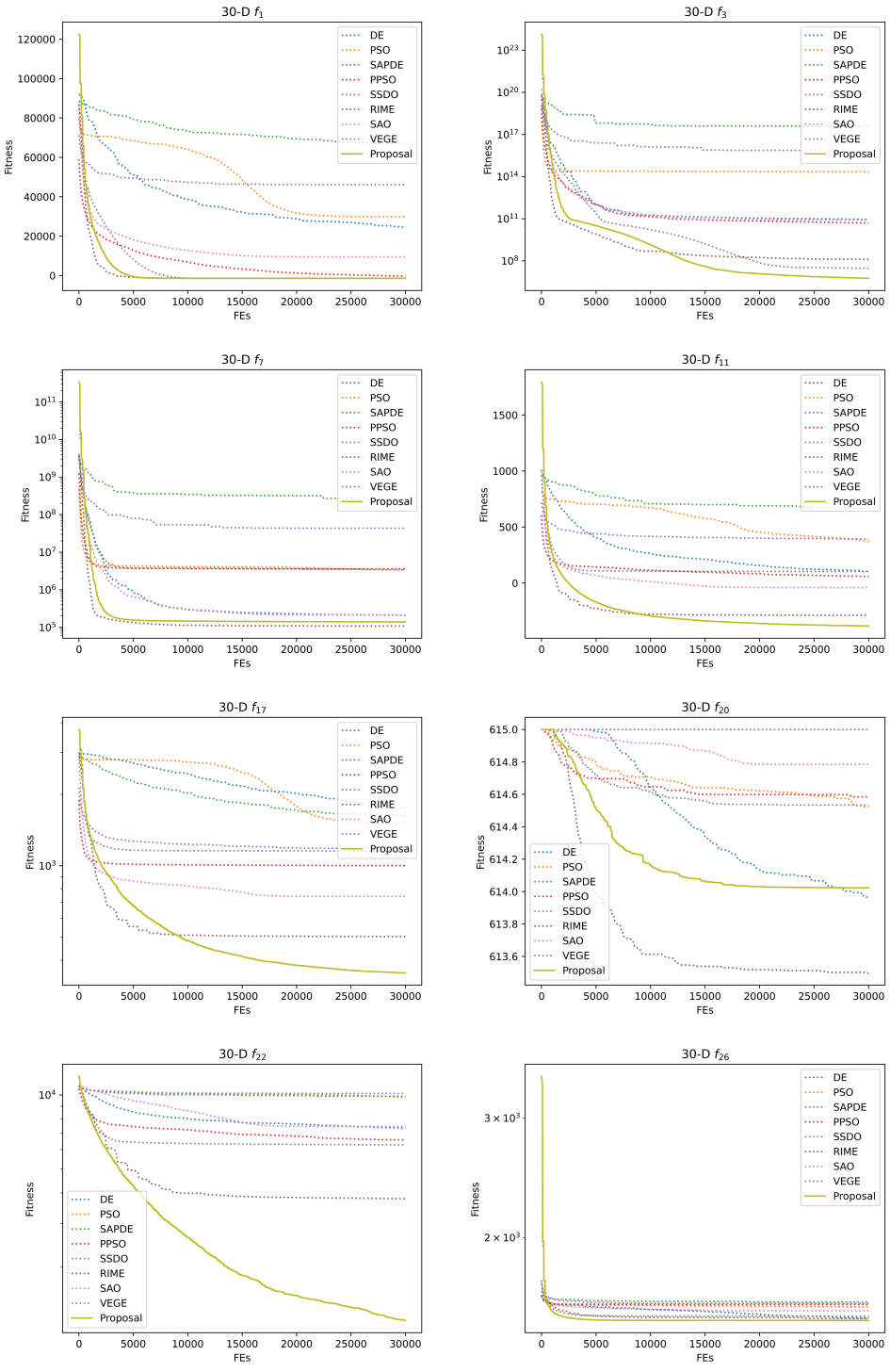


Figure 5.5: Convergence curves of competitor algorithms on 30-D CEC2013 representative benchmark functions (f_1 and f_3: unimodal functions; f_7, f_{11}, f_{17} and f_{20}: multimodal functions; f_{22} and f_{26}: composite functions).

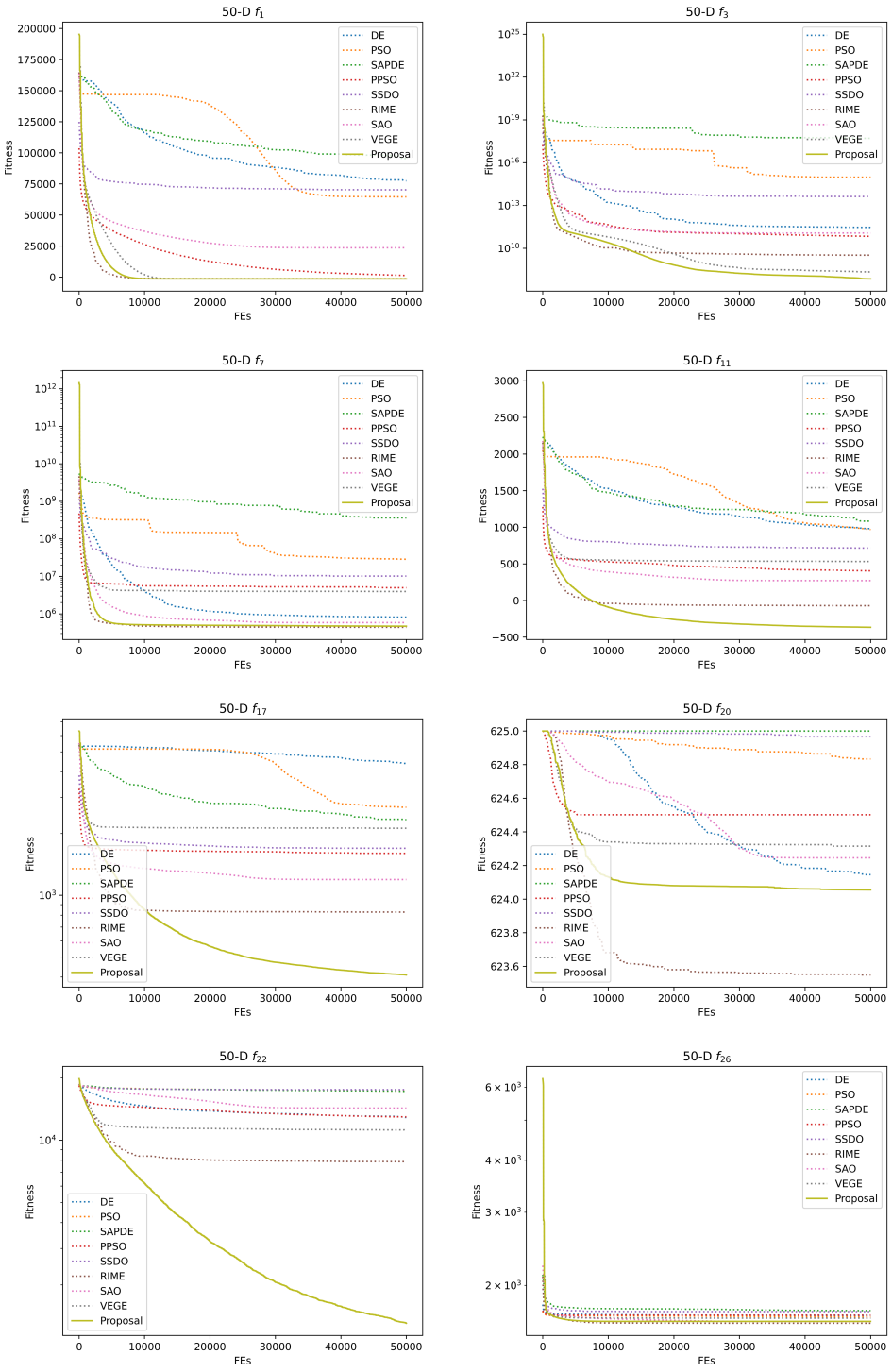


Figure 5.6: Convergence curves of competitor algorithms on 50-D CEC2013 representative benchmark functions.

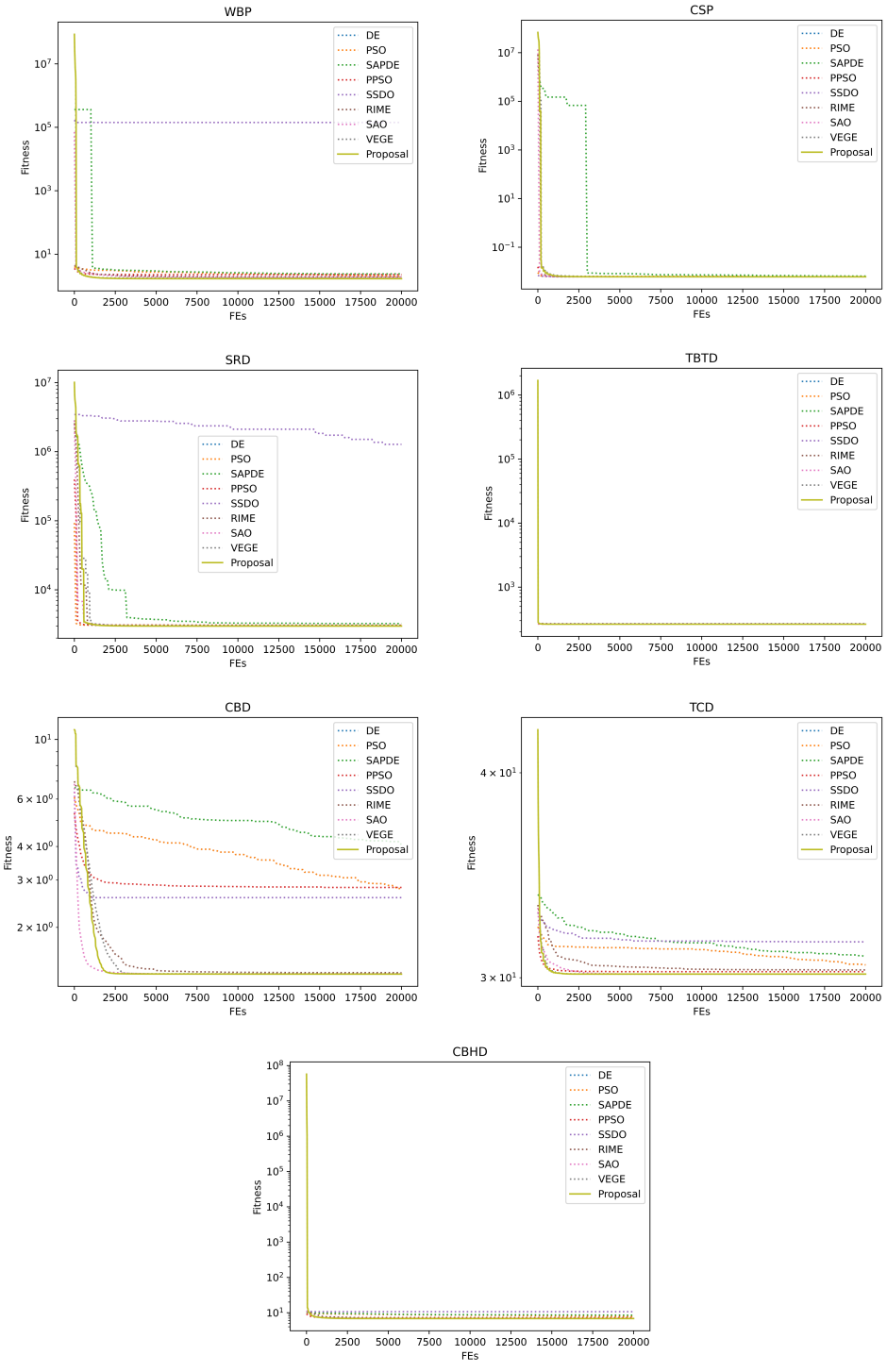


Figure 5.7: Convergence curves of competitor algorithms on seven engineering optimization problems.

Table 5.9: Ablation experimental results on 30-D CEC2013 benchmark functions.

Func	VEGE		VEGE-i		VEGE-ii		Proposal	
	mean	std	mean	std	mean	std	mean	std
f_1	-1.40e+03 +	8.37e-01	-1.40e+03 +	6.88e-01	**-1.40e+03** ≈	**1.32e-01**	-1.40e+03	1.63e-01
f_2	5.43e+06 ≈	2.36e+06	**4.84e+06** ≈	**2.22e+06**	4.90e+06 ≈	1.92e+06	5.35e+06	2.50e+06
f_3	2.98e+07 +	4.89e+07	6.16e+07 +	1.89e+08	**4.43e+06** ≈	**8.65e+06**	5.73e+06	9.05e+06
f_4	1.54e+04 ≈	6.52e+03	1.47e+04 ≈	6.42e+03	1.44e+04 ≈	6.29e+03	**1.39e+04**	**5.15e+03**
f_5	-9.97e+02 +	4.32e+00	-9.97e+02 +	2.62e+00	-9.99e+02 ≈	2.55e-01	**-9.99e+02**	**2.07e-01**
f_6	-8.32e+02 ≈	3.04e+01	-8.36e+02 ≈	3.15e+01	-8.39e+02 ≈	2.99e+01	**-8.40e+02**	**3.62e+01**
f_7	3.42e+06 +	1.24e+07	2.45e+06 +	8.68e+06	1.40e+05 ≈	2.95e+04	**1.40e+05**	**4.08e+04**
f_8	-6.79e+02 ≈	4.44e-02	-6.79e+02 ≈	4.98e-02	**-6.79e+02** ≈	**6.07e-02**	-6.79e+02	4.68e-02
f_9	-5.68e+02 ≈	3.98e+00	-5.67e+02 +	3.50e+00	**-5.73e+02** ≈	**3.57e+00**	-5.71e+02	4.13e+00
f_{10}	-4.96e+02 ≈	6.48e-01	-4.96e+02 +	7.67e-01	-4.97e+02 ≈	9.79e-01	**-4.97e+02**	**6.01e-01**
f_{11}	1.03e+02 +	1.03e+02	6.81e+01 +	1.12e+02	-3.83e+02 ≈	5.52e+00 -	**-3.84e+02**	**4.27e+00**
f_{12}	1.54e+02 +	1.10e+02	1.95e+02 +	8.51e+01	**3.07e+01** ≈	**1.06e+02**	5.69e+01	9.84e+01
f_{13}	3.70e+02 +	1.24e+02	3.91e+02 +	9.34e+01	1.98e+02 ≈	8.52e+01	**1.91e+02**	**7.60e+01**
f_{14}	4.59e+03 +	4.93e+02	4.52e+03 +	5.85e+02	**2.39e+02** ≈	**1.21e+02**	2.49e+02	1.66e+02
f_{15}	4.52e+03 ≈	4.96e+02	4.55e+03 ≈	5.89e+02	4.23e+03 ≈	5.42e+02	**4.20e+03**	**5.88e+02**
f_{16}	2.02e+02 +	4.48e-01	2.02e+02 +	4.48e-01	**2.01e+02** ≈	**6.01e-01**	2.01e+02	3.57e-01
f_{17}	1.15e+03 +	1.94e+02	1.22e+03 +	1.42e+02	**3.51e+02** ≈	**5.10e+00**	3.54e+02	5.91e+00
f_{18}	1.92e+03 +	3.64e+02	1.93e+03 +	3.67e+02	7.10e+02 ≈	8.06e+01	**7.08e+02**	**8.71e+01**
f_{19}	5.32e+02 +	5.45e+00	5.33e+02 +	4.19e+00	**5.19e+02** ≈	**4.93e+00**	5.19e+02	6.41e+00
f_{20}	6.15e+02 +	1.96e-01	6.14e+02 ≈	1.30e-01	6.14e+02 ≈	6.85e-01	**6.14e+02**	**7.55e-01**
f_{21}	1.83e+03 +	2.45e+02	1.75e+03 +	2.96e+02	1.75e+03 ≈	2.16e+02	**1.69e+03**	**2.69e+02**
f_{22}	6.26e+03 +	7.09e+02	6.30e+03 +	7.92e+02	**1.21e+03** ≈	**1.82e+02**	1.21e+03	1.40e+02
f_{23}	5.86e+03 ≈	6.87e+02	6.03e+03 ≈	6.28e+02	**5.70e+03** ≈	**6.43e+02**	5.85e+03	7.33e+02
f_{24}	1.30e+03 +	1.34e+01	1.30e+03 +	1.22e+01	1.28e+03 ≈	9.28e+00	**1.28e+03**	**1.12e+01**
f_{25}	1.43e+03 +	1.42e+01	1.44e+03 +	1.72e+01	**1.40e+03** ≈	**1.29e+01**	1.40e+03	1.62e+01
f_{26}	1.53e+03 +	8.66e+01	1.54e+03 +	8.37e+01	1.52e+03 ≈	8.05e+01	**1.51e+03**	**8.33e+01**
f_{27}	2.67e+03 +	1.12e+02	2.74e+03 +	1.55e+02	2.43e+03 ≈	1.24e+02	**2.43e+03**	**8.71e+01**
f_{28}	4.93e+03 +	4.48e+02	4.72e+03 +	6.73e+02	**3.57e+03** ≈	**1.03e+03**	3.64e+03	1.22e+03
	+/≈/−:20/8/0		+/≈/−:21/7/0		+/≈/−:0/28/0			

also negligible, but the performance improvement is indeed significant. They thus can be attributed to low cost and high return.

Next, we want to discuss the potential of the proposed two strategies and provide some open topics. Not limited to the conventional VEGE, our proposal can be easily combined with other improved versions of the VEGE. Since the two strategies are separable, we can also use either of them to combine with the VEGE. Thus, a topic worthy of further research is to dynamically select the combination of strategies according to the characteristics of the optimization problem. Besides, we simply use three different mutation methods to simulate the mutation patterns of real plants, which is far from enough because the real situation is more diverse and complex. Thus, how to add more diverse mutation methods

Table 5.10: Ablation experimental results on 50-D CEC2013 benchmark functions.

Func	VEGE		VEGE-i		VEGE-ii		Proposal	
	mean	std	mean	std	mean	std	mean	std
f_1	-1.39e+03 +	2.03e+00	-1.39e+03 +	1.79e+00	**-1.40e+03** ≈	**8.74e-01**	-1.40e+03	1.07e+00
f_2	9.29e+06 −	2.54e+06	**8.94e+06** −	**2.44e+06**	1.02e+07 ≈	2.72e+06	1.15e+07	2.89e+06
f_3	2.21e+08 +	4.31e+08	2.83e+08 +	9.82e+08	7.89e+07 ≈	1.36e+08	**7.26e+07**	**1.70e+08**
f_4	**1.06e+04** −	**4.20e+03**	1.08e+04 −	3.81e+03	1.52e+04 ≈	6.96e+03	1.47e+04	4.77e+03
f_5	-9.95e+02 +	1.36e+00	-9.95e+02 +	1.13e+00	**-9.98e+02** ≈	**5.06e-01**	-9.98e+02	6.83e-01
f_6	-8.04e+02 ≈	4.49e+01	**-8.15e+02** ≈	**3.37e+01**	-8.09e+02 ≈	3.48e+01	-8.11e+02	4.28e+01
f_7	3.95e+06 +	8.69e+06	1.45e+06 +	1.38e+06	5.28e+05 ≈	2.43e+05	**4.69e+05**	**8.88e+04**
f_8	-6.79e+02 ≈	4.25e-02	-6.79e+02 ≈	3.46e-02	-6.79e+02 ≈	4.17e-02	**-6.79e+02**	**3.67e-02**
f_9	-5.38e+02 +	4.62e+00	-5.38e+02 +	4.32e+00	**-5.45e+02** ≈	**4.91e+00**	-5.43e+02	5.21e+00
f_{10}	-4.88e+02 ≈	1.88e+00	-4.88e+02 ≈	2.23e+00	**-4.90e+02** ≈	**2.42e+00**	-4.89e+02	2.29e+00
f_{11}	5.33e+02 +	1.51e+02	4.72e+02 +	1.39e+02	-3.65e+02 ≈	8.90e+00	**-3.66e+02**	**7.80e+00**
f_{12}	5.69e+02 +	1.55e+02	5.84e+02 +	1.58e+02	**3.41e+02** ≈	**9.74e+01**	4.08e+02	1.30e+02
f_{13}	7.81e+02 +	1.59e+02	7.61e+02 +	1.48e+02	5.85e+02 ≈	1.52e+02	**5.80e+02**	**1.35e+02**
f_{14}	8.28e+03 +	9.40e+02	8.31e+03 +	8.13e+02	**3.21e+02** ≈	**1.68e+02**	3.27e+02	1.62e+02
f_{15}	8.77e+03 ≈	7.62e+02	8.90e+03 ≈	1.04e+03	**8.49e+03** ≈	**1.06e+03**	8.67e+03	8.16e+02
f_{16}	2.03e+02 ≈	5.45e-01	2.03e+02 ≈	4.64e-01	**2.02e+02** ≈	**5.70e-01**	2.02e+02	6.19e-01
f_{17}	2.12e+03 +	3.14e+02	2.06e+03 +	2.77e+02	**4.08e+02** ≈	**9.63e+00**	4.09e+02	9.50e+00
f_{18}	3.45e+03 +	5.13e+02	3.32e+03 +	4.18e+02	1.10e+03 ≈	1.41e+02	**1.02e+03**	**1.25e+02**
f_{19}	5.68e+02 +	1.10e+01	5.71e+02 +	1.00e+01	5.41e+02 ≈	6.26e+00	**5.39e+02**	**6.63e+00**
f_{20}	6.24e+02 ≈	3.66e-01	6.24e+02 ≈	2.83e-01	6.24e+02 ≈	5.92e-01	**6.24e+02**	**5.46e-01**
f_{21}	1.19e+03 +	2.17e+02	1.15e+03 +	1.31e+00	**1.13e+03** ≈	**3.75e+00**	1.14e+03	3.17e+00
f_{22}	1.12e+04 +	1.17e+03	1.10e+04 +	1.23e+03	1.32e+03 ≈	1.77e+02	**1.31e+03**	**1.33e+02**
f_{23}	1.11e+04 ≈	1.06e+03	1.12e+04 ≈	9.20e+02	**1.07e+04** ≈	**1.12e+03**	1.09e+04	9.74e+02
f_{24}	1.41e+03 +	1.29e+01	1.42e+03 +	2.55e+01	**1.36e+03** ≈	**1.79e+01**	1.36e+03	1.47e+01
f_{25}	1.58e+03 +	2.38e+01	1.58e+03 +	2.92e+01	1.51e+03 ≈	2.02e+01	**1.50e+03**	**1.84e+01**
f_{26}	1.67e+03 +	1.30e+01	1.67e+03 +	1.45e+01	**1.64e+03** ≈	**4.53e+01**	1.64e+03	1.10e+01
f_{27}	3.57e+03 +	2.51e+02	3.60e+03 +	2.49e+02	3.25e+03 ≈	1.59e+02	**3.25e+03**	**1.28e+02**
f_{28}	8.88e+03 +	1.16e+03	8.96e+03 +	1.81e+03	**4.65e+03** ≈	**1.83e+03**	4.93e+03	2.08e+03
	+/≈/−: 19/7/2		+/≈/−: 19/7/2		+/≈/−: 0/28/0			

is also one of our future topics. Since the distribution of individuals is constantly changing with the convergence of the population, the probabilities of different mutation methods being executed should also be different. Thus, how to efficiently use mutations to guide the convergence of the algorithm is also a promising topic. Although we have only given a few topics, there are still many interesting topics and hope to give some inspiration to the latecomers.

Subsequently, we apply the Kruskal-Wallis test and the Holm multiple comparison test to check significant differences among the compared algorithms on the CEC2013 benchmark functions. The results of statistical tests show that our proposal two strategies can further improve the performance of the conventional VEGE on most optimization prob-

lems, and the deterioration situation is rare and only happens in 50-D f_2 and f_4, which fully proves the effectiveness of our proposal two strategies. Besides, from the experimental and statistical results compared with optimization algorithms in Table 5.4, 5.5, 5.6, and 5.7, our proposal is quite competitive with these state-of-the-art optimization algorithms, and this conclusion can be also observed from the convergence curve of optimization processes. Owing to the population size in VEGE and our proposal being variable and the initial population size of our proposal being 10 while the other compared algorithms are set to 100, thus the beginning of the convergence curve in VEGE and our proposal is worse than other algorithms, but the excellent exploration and exploitation ability drives our proposal to outperform the compared algorithms rapidly, and the final optimum found by our proposal is also superior, which shows the domination of our proposal to the competitor algorithms in practice.

However, in some multimodal functions, our proposed VEGE with two strategies is significantly inferior to RIME (e.g. 30-D f_7 and f_9), and we attempt to explain this degeneration by the No Free Lunch Theory (NFL).[109] NFL states that any pair of black-box optimization algorithms have identical averaging performance in all possible problems, and if an algorithm performs well on a certain category of problem, it must degenerate on the rest of the problems since it is the only way to achieve identical averaging performance. Although the improvement from VEGE to our proposal can be observed, the original skeleton limits the performance of VEGE in these functions, and we will further analyze the reasons for the failure and give corresponding countermeasures in our future work.

In addition, the ablation experiment results in Table 5.9 and 5.10 show that the second strategy brings greater performance improvement than the first strategy on many functions, and the difference between the combination of the two and the second strategy alone is not obvious. This also supports our previous topic, that is, how to reasonably select search strategies is one of the important means to improve performance.

Finally, we apply our proposal to simulate the optimization of engineering problems.

In real-world applications, another performance indicator that we are concerned about is the robustness of the algorithm. Owing the evaluation for the real-world problem may be computationally expensive, we hope that the optima found by every trial run should be acceptable and close. Under the identical limitation of fitness evaluations, our proposal can outperform the compared methods significantly, and the mean, the std, the best, and the worst support the robustness of our proposal adequately, which practically proves that our proposal has great potential to deal with real-world optimization problems.

5.6 Conclusion

We introduce two new strategies into the conventional VEGE to further improve performance. The first strategy uses the competitive relationship to rationally allocate resources and expect to generate potential individuals, while the second strategy provides a variety of mutation methods to increase diversity and the ability to escape from local areas. The experimental results confirmed that both proposed strategies are effective, and the performance gains become more pronounced as the dimensionality increases.

Chapter 6

Strengthened RIME algorithm

This chapter presents a strengthened RIME algorithm[1]. Section 6.1 presents the motivation of this research, Section 6.2 introduces the original RIME algorithm, Section 6.3 details our proposal: SRIME, Section 6.4 presents the experiments on the CEC benchmark functions, and Section 6.5 discusses the performance of iVEGE. Finally, Section 6.6 concludes this chapter.

6.1 Motivation

RIME algorithm,[4] as a physics-based optimization technique, simulates the motion of soft rime and hard rime particles in the rime ice and formulates these phenomena with mathematical models to realize the optimization. Meanwhile, the simplicity, efficiency, scalability, and applicability of RIME have attracted widespread attention from scholars. As one of the latest EAs, we also notice some shortages of RIME. For example, it has outstanding exploitation capacity but is relatively weak in the exploration aspect, which limits the performance of RIME in complex optimization problems. Besides, the enhanced greedy

[1]This chapter was previously published as Zhong, R., Yu, J., Zhang, C., & Munetomo, M.: SRIME: a strengthened RIME with Latin hypercube sampling and embedded distance-based selection for engineering optimization problems. Neural Comput & Applic 36, 6721–6740 (2024). https://doi.org/10.1007/s00521-024-09424-4. Reproduced with permission from Springer Nature.[170]

selection mechanism in the original RIME can survive high-quality solutions but lose the diversity of the population, and this selection strategy will lead the direction of optimization into local optima easily, which is an obstacle to the scalability of RIME in multi-modal problems.

The objective of this chapter is to introduce some efficient strategies to improve the performance of RIME in complex optimization problems and propose a strengthened RIME (SRIME). SRIME inherits the main skeleton of RIME and modifies the strategy in three aspects: (1). Initialization, (2). hard rime exploitation, and (3). Selection mechanism. In numerical experiments, we adopt nine state-of-the-art EAs including the original RIME as competitor algorithms and evaluate the performance of SRIME in 10-D, 30-D, and 50-D CEC2020 benchmark functions and 12 engineering optimization problems. Besides, we implement ablation experiments to investigate the performance of each strategy that we proposed separately. More specifically, the main contribution of this chapter is as follows:

(1). We propose an SRIME to improve the overall performance of the original RIME. In the swarm initialization stage, we replace the random seed generator with the Latin hypercube sampling, and the benefit of this method can generate more uniform solutions in the high-dimensional search space. Given the shortages that we mentioned before, we add uncertainty and randomness to the exploitation phase of RIME and modify the hard rime search strategy.

(2). The enhanced greedy selection mechanism in the original RIME can significantly accelerate the convergence speed of optimization, especially in unimodal problems, but on the contrary, this selection strategy is too greedy to make the algorithm trap into local optima easily in complex optimization problems, thus, we introduce an embedded distance-based selection mechanism to RIME so that even the deteriorating solution has an opportunity to be accepted, which can enhance the diversity of the population and endow the capacity to RIME for getting rid of local optima.

(3). We evaluate our proposed SRIME on the standard CEC2020 benchmark functions

and 12 popular engineering optimization problems with nine powerful optimization techniques, the experimental and statistical analysis prove the efficiency and applicability of SRIME sufficiently.

(4). Comprehensive ablation experiments are provided in our experimental study to investigate the performance of our proposed three improvements independently.

6.2 RIME algorithm

As same as most population-based EAs, population or swarm initialization is the first step of RIME. Similarly, RIME randomly generates the initial population matrix with uniform distribution. Eq. (6.1) describes the structure of rime population R and the generation of each rime particle x_{ij}.

$$R = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ x_{31} & x_{32} & \cdots & x_{3m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix} \tag{6.1}$$

$$x_{ij} = r_1 \cdot (UB_j - LB_j) + LB_j$$

LB_j and UB_j are the lower and the upper bound of the j^{th} dimension and r_1 is a random number in (0, 1). x_{ij} represents the rime particle of the j^{th} dimension of the i^{th} individual, where $i \in [1, n]$ and $j \in [1, m]$.

Figure 6.1 [2]. shows two different forms of rime ice: soft rime ice and hard rime ice, which is the inspiration of the RIME.

In the breeze environment, soft rime ice grows along the surface of branches with high randomness, which is described as an exploration operator in RIME that enables the algo-

[2]Pictures are downloaded from `https://pixabay.com/` as copyright-free images.
(a).`https://pixabay.com/photos/barbed-wire-frost-frozen-cold-ice-1938842/`.
(c).`https://pixabay.com/photos/thuja-ice-winter-cold-frozen-6015613/`

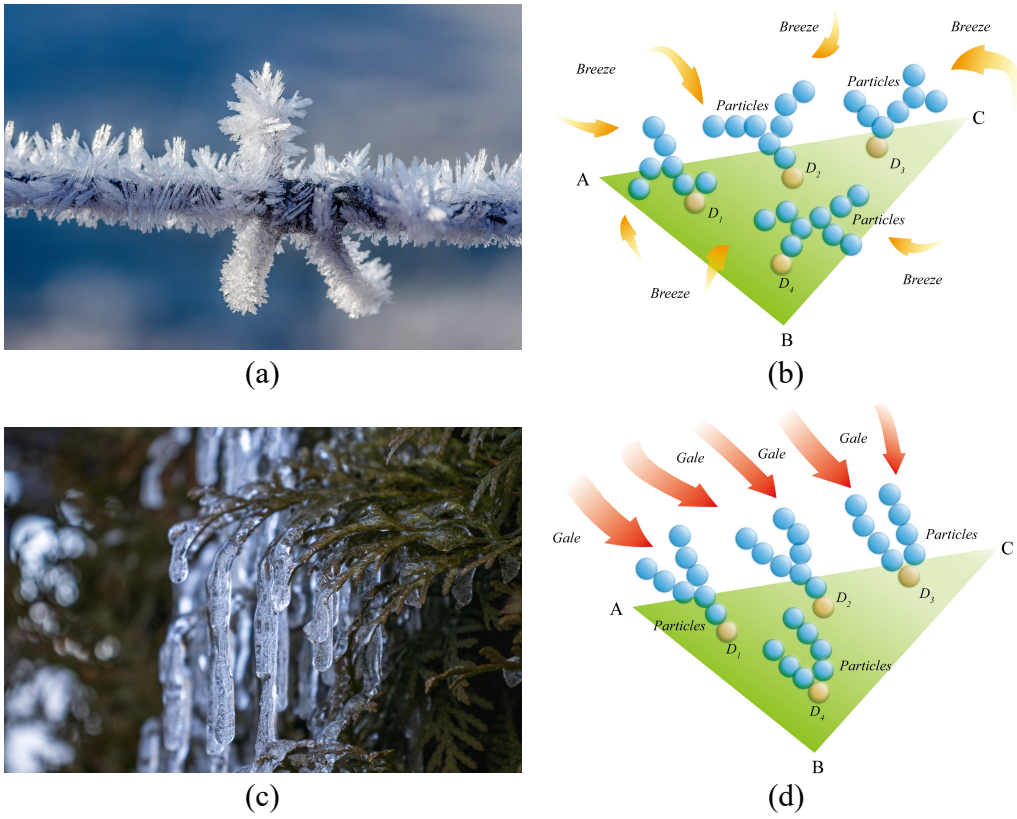


Figure 6.1: The real-world rime ice and corresponding mathematical models.[4] (a). soft rime ice. (b). mathematical model of soft rime ice. (c). hard rime ice. (d). mathematical model of hard rime ice.

rithm to cover the search space as much as possible in the early stage of optimization, and Eq. (6.2) formulates the growth of the soft rime ice.

$$R_{ij}^{t+1} = R_{best,j} + r_2 \cdot \cos(\theta) \cdot \beta \cdot (h \cdot (UB_j - LB_j) + LB_j),\ if\ r_3 < E \tag{6.2}$$

$R_{best,j}$ is the j^{th} rime particle of the best rime individual. r_2 is a random number in (-1, 1) to control the direction of particle movement cooperating with the $\cos(\theta)$, β is the self-adaptive environmental factor, h is a random number in (0, 1) represents the degree of adhesion, r_3 is also a random number in (0, 1), and E ($0 < E \leq 1$) is the coefficient to affect the condensation probability of an agent and increases as the optimization promotes.

Eq. (6.3) shows the computation of parameter θ, β, and E.

$$\begin{aligned} \theta &= \pi \cdot \frac{t}{10 \cdot t_{max}} \\ \beta &= 1 - \left[\frac{w \cdot t}{t_{max}} \right] / w \\ E &= \sqrt{t / t_{max}} \end{aligned} \tag{6.3}$$

where t and t_{max} are the current and the maximum number of the iteration time respectively, w is a constant which is suggested to 5, and $[\cdot]$ denotes the rounding operator.

In the gale environment, the mechanism of the hard rime ice growth is simpler and more regular than the growth of soft rime ice, which is expressed by Eq. (6.4).

$$R_{ij}^{t+1} = R_{best,j},\ if\ r_4 < f^{nor}(R_i) \tag{6.4}$$

where r_4 is a random value in (-1, 1) and $f^{nor}(R_i)$ is the normalized fitness of R_i. If the condition of Eq. (6.4) is satisfied, the i^{th} offspring individual replaces the value of j^{th} rime particle by the best rime individual R_{best}.

Besides, RIME adopts an enhanced greedy selection mechanism, which compares the parent individual with the corresponding offspring individual one by one, if the offspring individual has better fitness, it will directly replace the parent individual and participate in the subsequent optimization process.

In summary, the pseudocode of RIME is shown in Figure 6.2.

6.3 Proposal: SRIME

This section introduces our proposed SRIME in detail. We first provide the overview of SRIME and involved techniques are introduced subsequently.

The overview of SRIME: Figure 6.3 shows the flowchart of SRIME. First, SRIME initializes the population with the Latin hypercube sampling technique, and if the optimiza-

Algorithm RIME

Require: Dimension: D, Population size: PS, Maximum iteration: T,
Ensure: Global optimum: GO

Initialize the rime population $R = \{R_1^0, R_2^0, ..., R_{PS}^0\}$ with D and PS
Get the current best solution GO from R
$t = 0$
while $t < T$ **do**
    Calculate coefficient E.
    for $i = 0$ *to* PS **do**
        for $j = 0$ *to* D **do**
            Update the i^{th} rime individual R_i^t by soft rime strategy.
            Update the i^{th} rime individual R_i^t by hard rime strategy.
        end for
    end for
    for $i = 0$ *to* PS **do**
        Ensure the individual R_i^t is within search space and evaluate it
        if R_i^{t+1} has better fitness than R_i^t **then**
            Survive R_i^{t+1} to the next generation # positive selection
            if R_i^{t+1} has better fitness than GO **then**
                Update GO with R_i^{t+1} # global optimum update
            end if
        end if
    end for
    $t = t + 1$
end while
return GO

Figure 6.2: The pseudocode of RIME algorithm.

tion is not terminated, SRIME updates rime individuals with the soft rime and modified hard rime search strategy in order, then, the embedded distance-based selection mechanism selects the offspring to the subsequent generation of optimization. These processes are repeated until computational resources are exhausted. Next, we will introduce the Latin hypercube sampling, the modified hard rime search strategy, and the embedded distance-based selection mechanism.

Latin hypercube sampling: The initial population is critical to the optimization process since a high-quality population can help the algorithm capture the characteristics of the fitness landscape while a poor population may lead the direction of optimization into the bad search area. The conventional random sampling strategy can generate an approximately uniform population in the low-dimensional search space, however, in the high-dimensional search space, this simple scheme cannot ensure that each interval is represented in the sample or that the sample points are well-spaced, and clusters or gaps may exist in the sample space, which will affect the distribution of solutions in the fitness land-

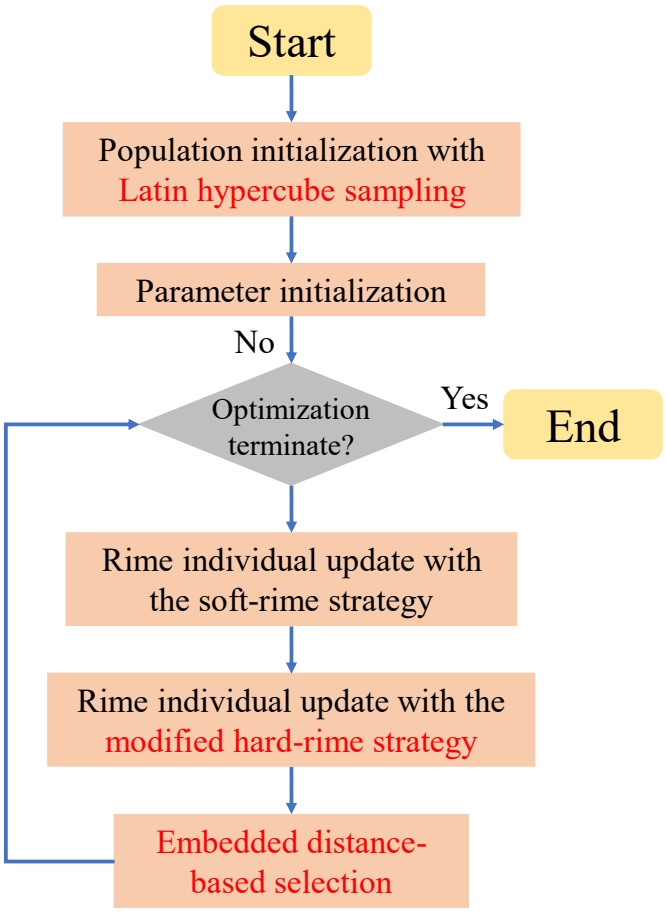


Figure 6.3: The flowchart of SRIME.

scape.

To overcome the shortage of simple random initialization, Latin hypercube sampling (LHS) was proposed by Michael Stein,[171] which is a statistical method for constructing a near-random sample of parameter values from a multi-dimensional distribution. Here, we first provide a visual demonstration of Latin hypercube sampling with four samples and two-dimensional search space in Figure 6.4.

LB_i and UB_i are the lower bound and the upper bound of the i^{th} dimension respectively. Latin hypercube sampling first divides the search space into intervals equally, and the interval for every dimension is paired randomly to form the sampling areas. Then, each sample is generated randomly within the sampling areas in order. This technique can eas-

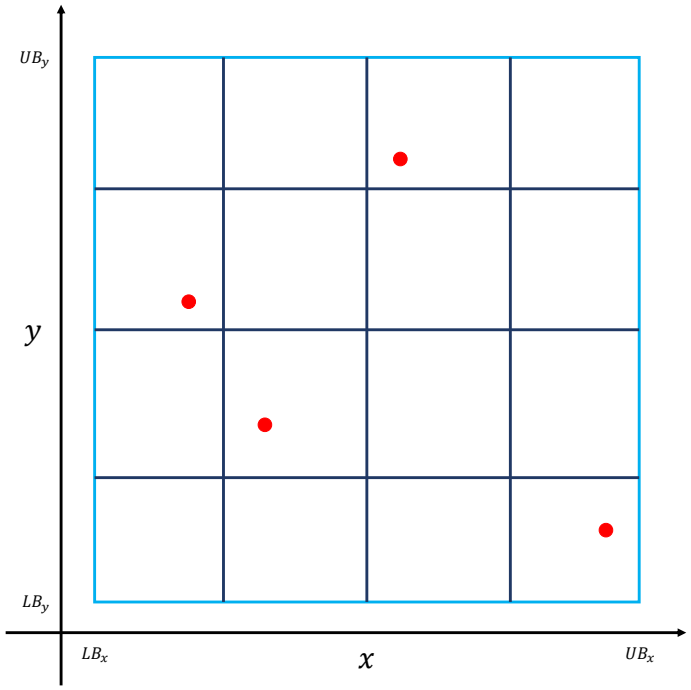


Figure 6.4: A visual demonstration of Latin hypercube sampling in two-dimensional search space, and the sample size is equal to four.

ily extend to high-dimensional search space. Compared with random initialization, Latin hypercube sampling can ensure that each interval is represented by a sample so that the sample points are more uniformly distributed at the cost of the computational budget.

Modified hard rime search strategy: The essence of the hard rime search strategy in the original RIME can be traced back to the crossover operator of DE.[49] Here, we list the hard rime search strategy of RIME and the conventional crossover operator of DE with a similar format for easy comparison.

$$R_{ij}^{t+1} = \begin{cases} R_{best,j},\ if\ r_4 < f^{nor}(R_i) \\ R_{ij}^{t+1},\ otherwise \end{cases} \tag{6.5a}$$

$$R_{ij}^{t+1} = \begin{cases} R_{ij}^{t+1},\ if\ r_4 < Cr \\ R_{ij}^{t},\ otherwise \end{cases} \tag{6.5b}$$

Eq. (6.5a) is the hard rime search strategy of RIME and Eq. (6.5b) is the crossover operator

of DE. Cr is the crossover rate. Two different components of these two strategies are the crossover objects and the crossover probability. RIME blends the current best solution R_{best} and soft rime generated solution R_i^{t+1}, while DE mixes the offspring individual R_i^{t+1} generated by mutation operator with the parent solution R_i^t. The hard rime search strategy of RIME utilizes the current best individual repeatedly, and the benefit of this operator can speed up the convergence of optimization. However, especially in the multi-modal problem, once the optimization direction of the RIME traps into a local optimum, in this case, R_{best} has a strong attraction to every individual, and this operator will stagnate the optimization rapidly. Therefore, it is necessary to add some randomness to the hard rime search strategy. Here, we propose a modified hard rime search strategy

$$R_{ij}^{t+1} = \begin{cases} R_{best,j} + f^{nor}(R_i) \cdot (R_{r_1,j} - R_{r_2,j}),\ if\ r_4 < f^{nor}(R_i) \\ R_{ij}^{t+1},\ otherwise \end{cases} \tag{6.6}$$

where $R_{r_1} - R_{r_2}$ are two mutually different rime individuals, and we simply use the normalized fitness of R_i as the scaling factor, and this extra term is added as the randomness to avoid the optimization premature.

Embedded distance-based selection: Distance-based selection is a parameter-free selection scheme first proposed in[105] to deal with the optimization in noisy environments. Given the uncertainty in noisy environments, the domination between two solutions may change as the intensity of noise changes dynamically. Thus, the distance-based selection allows the degenerated solution to still have a probability of being accepted and surviving. Eq. (6.7) describes the distance-based selection mechanism with the formulation.

$$R_i^{t+1} = \begin{cases} R_i^{t+1},\ if\ f(R_i^{t+1}) < f(R_i^t) \\ R_i^{t+1},\ else\ if\ r_5 \leq e^{-\frac{\Delta f}{Dist}} \\ R_i^t,\ otherwise \end{cases} \tag{6.7}$$

where r_5 is a random value in (0,1), $\Delta f = |f(R_i^{t+1}) - f(R_i^t)|$, and $Dist$ denotes the Manhattan distance between R_i^{t+1} and R_i^t, which can be calculated by Eq. (6.8).

$$Dist(R_i^{t+1}, R_i^t) = \sum_{j=1}^{D} |R_{ij}^{t+1} - R_{ij}^t| \tag{6.8}$$

Three cases are contained in the distance-based selection. Notice that this explanation is under the background of the minimization problem in noisy environments.

Case 1: If the offspring individual has better fitness in noisy environments, then it will replace the parent and survive.

Case 2: In this case, the fitness of the offspring individual is worse than the parent, but the condition of $r_5 \leq e^{-\frac{\Delta f}{Dist}}$ is satisfied, and the degrade offspring individual R_i^{t+1} is still accepted in this situation.

Case 3: If the degeneration of the offspring individual is unaffordable, then, the offspring individual will be rejected.

Here, we emphasize **Case 2** in the distance-based selection. Figure 6.5 shows a demonstration of the distance-based selection in noisy environments and the multi-modal functions.

The original intention of distance-based selection is to endow the anti-noise ability to the algorithm. As shown in Figure 6.5 (a), we can only observe the fitness of solutions in noisy environments, and the movement from the parent individual to the offspring individual in Figure 6.5 (a) will be rejected by the conventional greedy selection, however, the offspring individual actually performs better than the parent individual in the real fitness landscape, and as the noise changes, the domination between these two solutions will also change. Thus, the introduction of distance-based selection to noisy environments is a great success, both in theory and experimental support.

Considering the greed level of the selection mechanism in the original RIME is too high to limit the performance of RIME in multi-modal problems, we introduce the distance-

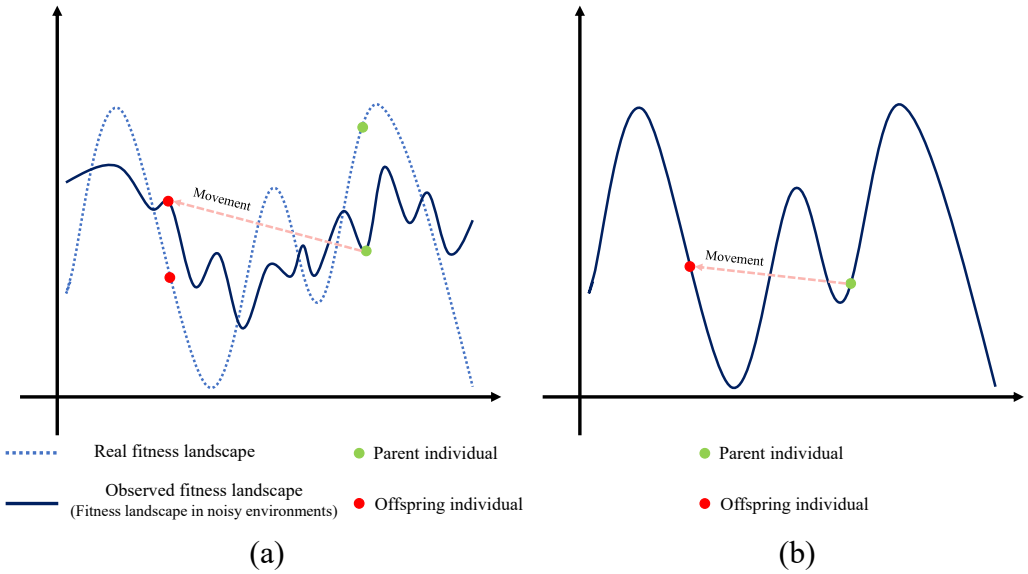


Figure 6.5: A visual demonstration of the distance-based selection. (a). In noisy environments. (b). In the multi-modal fitness landscape.

based selection to RIME. Figure 6.5 (b) shows a demonstration in the multi-modal fitness landscape. Due to the existence of basin attraction,[172] the green parent individual is difficult to get rid of local optimum, and the deteriorating movement in Figure 6.5 (b) will be rejected by enhanced greedy selection in the original RIME. However, to get rid of the local optimum and its attraction, this deteriorating movement is worthy to be accepted. Similarly, the distance-based selection also offers an opportunity for this acceptance. Besides, we focus on the threshold $e^{-\frac{\Delta f}{Dist}}$ in **Case 2** that we fix the Δf between two solutions, and if the Manhattan distance between these two solutions is farther, the value of this threshold $e^{-\frac{\Delta f}{Dist}}$ and the acceptance rate of deteriorating movement also become larger, which can help the direction of optimization get rid of the local optimum.

The above illustration explains the benefit of distance-based selection in complex optimization problems adequately, and we embed it into our proposed SRIME. Figure 6.6 shows the difference between RIME and our proposed SRIME in the offspring individuals generation and the update process.

In our proposed SRIME, the update process is activated immediately after a new off-

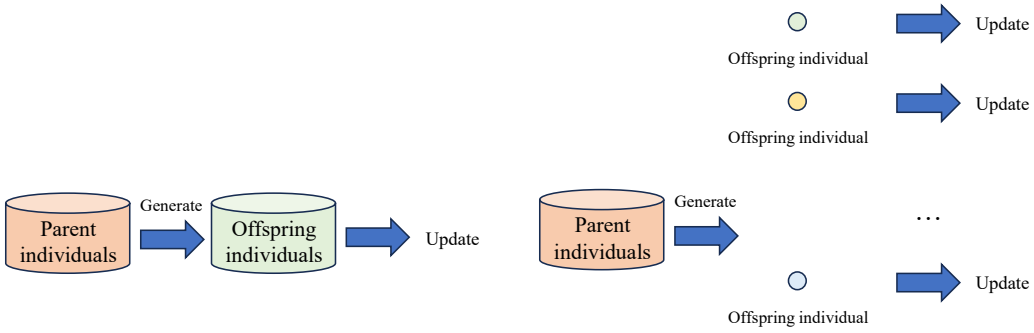


Figure 6.6: The difference between RIME and SRIME in the offspring individuals generation and the update process. (a). In the RIME, these two processes are separated strictly. (b). In the SRIME, the update is embedded after each offspring individual is generated.

spring individual is generated as shown in Figure 6.6 (b). Given both the soft rime and the hard rime search strategies in the original RIME use the current optimum R_{best} to generate offspring individuals, our proposed embedded update strategy can update and utilize the current optimum R_{best} promptly.

In summary, the pseudocode of SRIME is shown in Algorithm 6.7.

Computational complexity analysis: SRIME is a population-based stochastic optimization technique, thus, the skeleton of SRIME is like most EAs and can be roughly divided into two stages: initialization and iteration search (i.e. soft rime search and modified hard rime search). Given the dimension of the problem is D, the population size is N and the maximum iteration time is T. First, the computational complexity of population initialization is $O(N \cdot D)$, then, SRIME enters the soft rime search, modified hard rime search, and embedded distance-based selection asynchronously. For a single individual, the computational complexity of soft rime search and modified hard rime search is $O(D)$ and $O(D)$. In embedded distance-based selection, as we analyzed in Eq. (6.7) three cases are involved: In case 1 when the offspring individual has better fitness, it will be accepted, and the computational complexity is $O(1)$. In case 2 when the offspring individual has worse fitness but $r_5 \leq e^{-\frac{\Delta f}{Dist}}$ is satisfied, the offspring individual will be also accepted. In this case, the Manhattan distance between the offspring individual and the parent individual

Algorithm SRIME

Require: Dimension: D, Population size: PS, Maximum iteration: T,
Ensure: Global optimum: GO

Initialize the rime population $R = \{R_1^0, R_2^0, ..., R_{PS}^0\}$ with Latin hypercube sampling
Get the current best solution GO from R
$t = 0$
while $t < T$ **do**
    Calculate coefficient E.
    for $i = 0$ *to* PS **do**
        for $j = 0$ *to* D **do**
            Update the i^{th} rime individual R_i^t by soft rime strategy
            Update the i^{th} rime individual R_i^t by modified hard rime strategy
        end for
        Ensure the individual R_i^t is within search space and evaluate it
        if R_i^{t+1} has better fitness than R_i^t **then** # embedded selection
            Survive R_i^{t+1} to the next generation
            if R_i^{t+1} has better fitness than GO **then**
                Update GO with R_i^{t+1} # global optimum update
            end if
        else if $r_5 \leq e^{-\frac{\Delta f}{Dist}}$ is satisfied **then**
            Survive R_i^{t+1} to the next generation # distance-based selection
        end if
    end for
    $t = t + 1$
end while
return GO

Figure 6.7: The pseudocode of SRIME algorithm.

is calculated, and the computational complexity is $O(D)$. In case 3, the above conditions are not satisfied, the parent individual will survive to the next generation, and the computational complexity is $O(1)$. Besides, for a single RIME individual, only one selection operator will be activated, and the computational complexity of embedded distance-based selection is $\max\{O(1), O(D), O(1)\} = O(D)$.

In summary, the computational complexity of SRIME is

$$O(N \cdot D + T \cdot (N \cdot (D + D + D))) = O(N \cdot D + 3 \cdot T \cdot N \cdot D)) \tag{6.9}$$

6.4 Numerical experiments

This section introduces the numerical experiments in detail. Section 6.4.1 introduces the experiment settings: experimental environments, benchmark functions, and compared methods with their parameters. Section 6.4.2 shows the experimental and statistical results.

6.4.1 Experiment settings

Experimental environments and implementation: All optimization methods are programmed by Python 3.11 and implemented in Lenovo Legion R9000P, which is equipped with Windows 11, AMD Ryzen 7 5800H with Radeon Graphics 3.20 GHz, and 16GB RAM. All compared algorithms are provided by the MEALPY library,[166] CEC2020 benchmark functions by OpFuNu library,[173] and eight engineering optimization problems are provided by ENOPPY library.[164]

Benchmark functions: We evaluate our proposed SRIME on CEC2020 benchmark functions[174] and eight engineering optimization problems as same as in.[175] The summary of CEC2020 benchmark functions and eight engineering optimization problems are listed in Table 6.1 and 6.2 respectively. More detailed formulation and visualization of engineering optimization problems can be found in.[175]

Table 6.1: Summary of the CEC2020 benchmark functions: Uni.=Unimodal function, Multi.=Multimodal function, Hybrid.=Hybrid function, Comp.=Composition function

Fun.	Description	Feature	Optimum
f_1	Shifted and Rotated Bent Cigar Function	Uni.	100
f_2	Shifted and Rotated Schwefel's function	Multi.	1100
f_3	Shifted and Rotated Lunacek bi-Rastrigin function		700
f_4	Expanded Rosenbrock's plus Griewangk's function		1900
f_5	Hybrid function 1 (N = 3)	Hybrid.	1700
f_6	Hybrid function 2 (N = 4)		1600
f_7	Hybrid function 3 (N = 5)		2100
f_8	Composition function 1 (N = 3)	Comp.	2200
f_9	Composition function 2 (N = 4)		2400
f_{10}	Composition function 3 (N = 5)		2500

Compared methods and parameters: We compare our proposed SRIME to two categories of EAs with a total of nine EAs: (1). Classic EAs including PSO,[52] DE,[49] covariance matrix adaptation evolutionary strategy (CMA-ES),[176] and grey wolf optimizer (GWO).[177] (2). The latest EAs including artificial ecosystem-based optimization (AEO),[178] zebra optimization algorithm (ZOA),[179] war strategy optimization (WSO),[180] energy valley optimizer

Table 6.2: Summary of eight engineering optimization problems.

Name	Abbr.	Dim.	# of constraints
Welded Beam Problem	WBP	4	7
Speed Reducer Problem	SRD	7	11
Cantilever Beam Problem	CBD	5	1
I Beam Problem	IBD	4	2
Tubular Column Problem	TCD	2	6
Piston Lever Problem	PLD	4	4
Corrugated Bulkhead Problem	CBHD	4	6
Reinforced Concrete Beam Problem	RCB	3	2

(EVO),[181] and the original RIME.[4] The population size of all algorithms is set to 100, the maximum FEs for CEC2020 functions is $1000 \cdot$ D (D=dimension size), and the maximum FEs for engineering problems is 20,000. To alleviate the randomness in the optimization, each EA is executed with 30 trial runs. The detailed parameter settings of the competitor algorithms are listed in Table 6.3. All parameters are consistent with the suggested settings in corresponding papers.

Given the engineering optimization problem contains constraints and original EAs including SRIME cannot deal with constrained optimization problems, thus, we equip all EAs with the static penalty function,[165] which is defined by Eq. (6.10)

$$F(R_i) = f(R_i) + w \cdot \sum_{i=1}^{m} (\max(0, g_i(R_i))) \tag{6.10}$$

$F(\cdot)$ is the fitness function, while $f(\cdot)$ and $g_i(\cdot)$ are the objective function and constraint function respectively. w is a constant set to $10e7$ by default.

Besides, the ablation experiment on CEC2020 benchmark functions is also provided to evaluate the contribution of each proposed component. We re-emphasize our proposed three adjustments in RIME: (1). Latin hypercube sampling for population initialization. (2). Modified hard rime search strategy. (3). Embedded distance-based selection mechanism, which combined with the original RIME independently.

Table 6.3: The parameters of all compared optimization methods

EAs	Parameters	Value
PSO	Inertia factor w	1
	Acceleration coefficients c_1 and c_2	2.05
	Max. and min. speed	2, -2
DE	Mutation scheme	DE/cur-to-rand/1
	Scaling factor F	0.8
	Crossover rate Cr	0.9
CMA-ES	σ	1.3
GWO	Explicit hyperparameter-free	
AEO	Explicit hyperparameter-free	
ZOA	Explicit hyperparameter-free	
WSO	parameter ρ_r	0.5
EVO	Explicit hyperparameter-free	
RIME	parameter w	5
SRIME	parameter w	5

6.4.2 Experimental and statistical results

This section shows the experimental and statistical results of optimization, all EAs in each function are independently implemented 30 times. To determine the statistical significance between every pair of competitor algorithms, we first apply the Kruskal–Wallis. If the statistical significance exists, then, the Mann–Whitney U test is employed to calculate the p-value between every pair of algorithms. Finally, we use the Holm multiple comparison test[182] to correct the p-value obtained by the Mann–Whitney U test and identify the statistical significance level. Symbols $+$, $\approx$, and $-$ are applied to represent that our proposed SRIME is significantly better, with no significance, and significantly worse with the compared method. Additionally, the Friedman test is also conducted to calculate the averaging rank of every algorithm, and the best value is in bold.

Optimization performance on CEC2020 suite: Table 6.4, 6.5, 6.6, and 6.7 summarize the experimental and statistical results of optimization on 10-D, 30-D, 50-D, and 100-D CEC2020 benchmark functions among ten EAs. Due to the limitation of space, convergence curves of representative functions (the unimodal function f_1, the multimodal function f_2, the hybrid function $f5$, and the composition function) f_9 are provided in Figures 6.8 and 6.9.

Optimization performance on engineering problems: Table 6.8 summarizes the experimental and statistical results on eight engineering optimization problems among ten EAs, and Figure 6.10 visualizes the convergence curves.

Ablation experiment on CEC2020 benchmark functions: Table 6.9, 6.10, 6.11, and 6.12 summarize ablation experiments on 10-D, 30-D, 50-D, and 100-D CEC2020 benchmark functions. Here, we first define the abbreviation of compared algorithms. RIME$_{lhs}$: The original RIME combines with Latin hypercube sampling, RIME$_{mhs}$: The original RIME combines with the modified hard rime search strategy, and RIME$_{edbs}$: The original RIME combines with the embedded distance-based selection mechanism.

6.5 Discussion

In this section, we analyze the efficiency of our proposed SRIME based on the performance in the CEC2020 suite and eight engineering optimization problems, and some open topics are provided to further improve the SRIME in future works.

6.5.1 Performance analysis based on CEC2020 suite

Since benchmark functions in the CEC2020 suite have various characteristics such as shifted, rotated, unimodal, multi-modal, and composite, thus, it can fully evaluate the overall performance of our proposed SRIME. In the unimodal function (i.e. f_1), SRIME inherits the superior exploitation ability of RIME and outperforms PSO, DE, CMA-ES, GWO,

Table 6.4: Experimental and statistical results on 10-D CEC2020 benchmark functions. f_1: Unimodal function; $f_2 - f_4$: Multimodal functions; $f_5 - f_7$: Hybrid functions; $f_8 - f_{10}$: Composition functions; mean and std: the mean and the standard deviation of 30 trial runs.

Func.		PSO	DE	CMA-ES	GWO	AEO	ZOA	WSO	EVO	RIME	SRIME
f_1	mean	3.69e+08 +	7.64e+08 +	2.48e+07 +	4.28e+06 +	1.19e+08 +	2.29e+08 +	2.81e+09 +	4.15e+08 +	3.86e+05 +	**6.93e+04**
	std	5.84e+08	1.61e+08	5.32e+06	1.17e+07	1.41e+08	2.96e+08	1.46e+09	4.04e+08	4.57e+05	1.17e+05
f_2	mean	9.95e+09 +	7.50e+10 +	2.08e+09 +	2.16e+08 +	9.04e+09 +	2.04e+10 +	2.14e+11 +	2.67e+10 +	2.72e+07 +	**6.25e+06**
	std	1.25e+10	2.17e+10	6.73e+08	6.52e+08	8.35e+09	1.73e+10	8.71e+10	2.38e+10	2.44e+07	1.01e+07
f_3	mean	5.51e+09 +	3.49e+10 +	1.00e+09 +	1.70e+08 +	3.59e+09 +	6.47e+09 +	6.72e+10 +	1.15e+10 +	1.18e+07 +	**3.13e+06**
	std	1.21e+10	9.08e+09	3.14e+08	1.40e+08	4.42e+09	5.07e+09	3.49e+10	1.18e+10	1.23e+07	3.53e+06
f_4	mean	1.91e+03 +	1.93e+03 +	1.91e+03 +	1.90e+03 +	1.91e+03 +	1.91e+03 +	2.50e+03 +	1.93e+03 +	1.90e+03 +	**1.90e+03**
	std	1.32e+01	1.66e+01	8.08e-01	6.04e-01	6.97e+00	4.54e+00	9.95e+02	1.26e+02	9.00e-01	6.29e-01
f_5	mean	1.13e+05 +	1.23e+04 ≈	**2.07e+03** −	2.53e+04 +	1.68e+04 ≈	6.54e+04 +	5.34e+05 +	4.48e+04 +	1.19e+04 ≈	1.22e+04
	std	1.44e+05	3.46e+03	8.93e+01	1.23e+04	7.92e+03	9.13e+04	4.98e+05	7.45e+04	7.46e+03	8.46e+03
f_6	mean	8.42e+03 +	2.88e+03 +	**2.00e+03** ≈	3.22e+03 +	2.38e+03 +	5.68e+03 +	1.67e+04 +	6.32e+03 +	3.27e+03 ≈	2.25e+03
	std	7.56e+03	3.93e+02	1.92e+02	1.44e+03	5.16e+02	3.73e+03	7.96e+03	4.84e+03	2.52e+03	1.42e+03
f_7	mean	2.87e+05 +	5.72e+04 +	**6.47e+03** −	2.35e+04 ≈	1.28e+04 ≈	1.78e+04 ≈	5.70e+05 +	8.59e+04 +	1.83e+04 ≈	2.13e+04
	std	3.38e+05	2.27e+04	1.60e+03	1.48e+04	7.75e+03	8.90e+03	4.41e+05	2.32e+05	1.04e+04	1.61e+04
f_8	mean	2.32e+03 +	2.32e+03 +	2.31e+03 +	2.30e+03 ≈	2.31e+03 +	2.31e+03 +	2.33e+03 +	2.31e+03 +	2.30e+03 ≈	**2.30e+03**
	std	3.64e+00	1.48e+00	9.83e-01	3.62e+00	3.39e+00	2.40e+00	8.55e+00	2.92e+00	2.15e+00	1.43e+00
f_9	mean	3.46e+03 +	3.82e+03 +	2.87e+03 +	2.70e+03 +	3.04e+03 +	3.17e+03 +	4.26e+03 +	3.34e+03 +	2.63e+03 +	**2.59e+03**
	std	5.48e+02	1.50e+02	3.86e+01	5.77e+01	1.99e+02	3.34e+02	3.97e+02	4.22e+02	4.07e+01	3.38e+01
f_{10}	mean	3.09e+03 +	3.09e+03 +	3.00e+03 +	3.00e+03 +	3.07e+03 +	3.06e+03 +	3.15e+03 +	3.09e+03 +	2.99e+03 ≈	**2.99e+03**
	std	5.26e+01	3.37e+01	6.26e+00	1.26e+01	5.37e+01	3.80e+01	4.59e+01	3.44e+01	2.15e+01	1.99e+01
+/≈/− summary:		10/0/0	9/1/0	7/1/2	8/2/0	8/2/0	9/1/0	10/0/0	10/0/0	5/5/0	
Ave. rank		7.7	7.7	3.1	3.9	4.9	10	5.8	7.6	2.6	**1.7**

Table 6.5: Experimental and statistical results on 30-D CEC2020 benchmark functions.

Func.		PSO	DE	CMA-ES	GWO	AEO	ZOA	WSO	EVO	RIME	SRIME
f_1	mean	4.35e+09 +	3.87e+10 +	3.20e+09 +	5.10e+08 +	7.20e+09 +	1.75e+10 +	3.17e+10 +	1.01e+10 +	2.56e+07 +	**1.70e+06**
	std	1.85e+09	5.06e+09	5.67e+08	4.17e+08	2.16e+09	3.55e+09	8.66e+09	3.78e+09	1.06e+07	1.55e+06
f_2	mean	6.41e+11 +	3.79e+12 +	3.08e+11 +	6.21e+10 +	8.91e+11 +	2.00e+12 +	3.96e+12 +	9.99e+11 +	2.88e+09 +	**1.84e+08**
	std	5.70e+11	3.83e+11	4.39e+10	7.24e+10	3.63e+11	4.40e+11	9.07e+11	3.69e+11	1.16e+09	2.12e+08
f_3	mean	1.54e+11 +	1.24e+12 +	1.04e+11 +	1.52e+10 +	2.15e+11 +	6.14e+11 +	1.17e+12 +	3.54e+11 +	8.73e+08 +	**9.07e+07**
	std	8.98e+10	1.56e+11	1.85e+10	1.27e+10	6.58e+10	1.33e+11	2.32e+11	1.06e+11	4.15e+08	1.24e+08
f_4	mean	6.79e+03 +	9.84e+04 +	2.09e+03 +	1.92e+03 +	8.85e+03 +	2.43e+04 +	1.72e+05 +	9.15e+03 +	1.92e+03 +	**1.91e+03**
	std	6.93e+03	3.84e+04	9.09e+01	5.56e+00	1.00e+04	1.76e+04	1.07e+05	9.48e+03	3.54e+00	2.57e+00
f_5	mean	2.05e+07 +	1.51e+07 +	**5.51e+05** ≈	7.50e+05 ≈	5.06e+06 +	8.03e+06 +	4.89e+07 +	2.26e+06 +	9.33e+05 ≈	6.37e+05
	std	1.45e+07	5.77e+06	1.41e+05	4.98e+05	5.57e+06	8.67e+06	2.63e+07	2.63e+06	6.25e+05	3.56e+05
f_6	mean	3.84e+06 +	2.02e+05 +	**6.05e+03** −	3.64e+04 ≈	9.62e+04 +	1.76e+05 +	1.13e+07 +	1.98e+06 +	4.87e+04 ≈	3.82e+04
	std	9.37e+06	4.71e+05	1.51e+03	1.93e+04	1.16e+05	3.52e+05	1.46e+07	3.92e+06	1.91e+04	2.25e+04
f_7	mean	6.94e+07 +	3.56e+07 +	**5.40e+05** −	1.27e+06 ≈	1.36e+07 +	3.14e+07 +	2.29e+08 +	7.62e+06 +	1.40e+06 ≈	1.07e+06
	std	7.15e+07	1.06e+07	1.58e+05	7.95e+05	1.17e+07	2.51e+07	1.31e+08	7.33e+06	6.66e+05	7.06e+05
f_8	mean	2.79e+03 +	2.65e+03 +	2.45e+03 +	2.38e+03 ≈	2.91e+03 +	2.71e+03 +	3.20e+03 +	2.47e+03 +	2.40e+03 +	**2.38e+03**
	std	2.04e+02	3.37e+01	7.75e+00	9.81e+00	2.30e+02	8.70e+01	3.03e+02	2.77e+01	1.39e+01	8.46e+00
f_9	mean	8.61e+03 +	1.30e+04 +	6.27e+03 +	3.56e+03 +	8.46e+03 +	1.64e+04 +	2.21e+04 +	1.17e+04 +	2.97e+03 +	**2.70e+03**
	std	1.87e+03	6.30e+02	2.81e+02	8.88e+02	9.54e+02	2.20e+03	3.63e+03	1.88e+03	8.47e+01	7.07e+01
f_{10}	mean	3.68e+03 +	5.11e+03 +	3.11e+03 +	2.99e+03 +	3.53e+03 +	3.89e+03 +	4.80e+03 +	3.63e+03 +	2.95e+03 +	**2.93e+03**
	std	4.86e+02	4.66e+02	3.89e+01	4.86e+01	1.73e+02	2.56e+02	4.02e+02	1.82e+02	2.85e+01	1.51e+01
+/≈/− summary:		10/0/0	10/0/0	7/1/2	6/4/0	10/0/0	10/0/0	10/0/0	10/0/0	7/3/0	
Ave. rank		6.8	8.5	3.1	2.6	6.0	9.7	7.6	6.4	2.8	**1.5**

Table 6.6: Experimental and statistical results on 50-D CEC2020 benchmark functions.

Func.		PSO	DE	CMA-ES	GWO	AEO	ZOA	WSO	EVO	RIME	SRIME
f_1	mean	1.91e+10 +	1.07e+11 +	1.11e+10 +	2.70e+09 +	2.24e+10 +	5.15e+10 +	7.83e+10 +	3.95e+10 +	2.22e+08 +	**1.28e+07**
	std	3.76e+09	1.04e+10	3.09e+09	1.85e+09	4.87e+09	5.21e+09	9.89e+09	8.75e+09	6.34e+07	8.72e+06
f_2	mean	2.32e+12 +	1.15e+13 +	1.34e+12 +	2.38e+11 +	2.48e+12 +	5.62e+12 +	9.18e+12 +	4.21e+12 +	2.63e+10 +	**1.10e+09**
	std	5.45e+11	1.39e+12	4.36e+11	1.43e+11	5.73e+11	9.42e+11	1.25e+12	9.14e+11	8.67e+09	5.87e+08
f_3	mean	6.79e+11 +	4.13e+12 +	4.05e+11 +	9.48e+10 +	8.26e+11 +	1.94e+12 +	2.98e+12 +	1.22e+12 +	8.56e+09 +	**3.69e+08**
	std	1.98e+11	4.60e+11	8.35e+10	5.35e+10	1.99e+11	2.11e+11	5.35e+11	2.33e+11	2.52e+09	2.32e+08
f_4	mean	8.92e+04 +	1.40e+06 +	1.01e+04 +	1.96e+03 +	5.59e+04 +	2.94e+05 +	1.01e+06 +	1.07e+05 +	1.95e+03 +	**1.93e+03**
	std	1.31e+05	6.32e+05	6.69e+03	3.48e+01	5.80e+04	1.24e+05	4.79e+05	1.11e+05	1.10e+01	8.32e+00
f_5	mean	6.63e+07 +	6.23e+07 +	**2.39e+06** −	3.91e+06 −	2.18e+07 +	4.55e+07 +	1.33e+08 +	2.18e+07 +	6.89e+06 +	4.93e+06
	std	4.06e+07	2.18e+07	4.70e+05	3.78e+06	1.13e+07	2.84e+07	5.29e+07	1.64e+07	2.84e+06	2.44e+06
f_6	mean	1.03e+07 +	9.22e+07 +	**3.51e+04** −	3.93e+05 +	7.34e+06 +	2.75e+08 +	7.37e+08 +	8.72e+07 +	4.49e+05 +	7.91e+04
	std	1.46e+07	3.22e+07	6.68e+03	4.38e+05	8.34e+06	3.49e+08	5.46e+08	6.85e+07	4.49e+05	5.97e+04
f_7	mean	9.15e+08 +	7.53e+08 +	1.27e+07 +	6.93e+06 ≈	1.46e+08 +	6.09e+08 +	2.55e+09 +	1.81e+08 +	8.38e+06 ≈	**6.14e+06**
	std	7.65e+08	1.81e+08	3.96e+06	4.58e+06	1.10e+08	4.20e+08	1.56e+09	1.95e+08	4.46e+06	2.69e+06
f_8	mean	4.14e+03 +	3.19e+03 +	2.66e+03 +	**2.45e+03** −	6.30e+03 +	4.71e+03 +	6.81e+03 +	2.84e+03 +	2.52e+03 −	2.63e+03
	std	9.35e+02	1.06e+02	3.91e+01	1.90e+01	2.33e+03	7.31e+02	1.52e+03	9.98e+01	2.33e+01	8.87e+02
f_9	mean	2.31e+04 +	2.74e+04 +	1.12e+04 +	6.06e+03 +	1.85e+04 +	4.52e+04 +	5.56e+04 +	4.01e+04 +	3.85e+03 +	**2.96e+03**
	std	3.37e+03	1.91e+03	1.36e+03	1.51e+03	5.36e+03	5.31e+03	5.57e+03	8.14e+03	2.26e+02	2.41e+02
f_{10}	mean	8.44e+03 +	1.23e+04 +	4.58e+03 +	3.85e+03 +	7.03e+03 +	1.09e+04 +	1.71e+04 +	8.80e+03 +	3.71e+03 +	**3.41e+03**
	std	1.88e+03	1.78e+03	2.49e+02	2.59e+02	7.70e+02	1.38e+03	3.30e+03	1.00e+03	1.85e+02	1.00e+02
+/≈/− summary:		10/0/0	10/0/0	8/0/2	7/1/2	10/0/0	10/0/0	10/0/0	10/0/0	8/1/1	
Ave. rank		6.4	8.6	3.4	2.6	5.8	9.6	8.0	6.6	2.5	**1.5**

Table 6.7: Experimental and statistical results on 100-D CEC2020 benchmark functions.

Func.		PSO	DE	CMA-ES	GWO	AEO	ZOA	WSO	EVO	RIME	SRIME
f_1	mean	1.01e+11 +	3.33e+11 +	8.40e+10 +	1.50e+10 +	6.96e+10 +	1.75e+11 +	2.22e+11 +	1.48e+11 +	3.19e+09 +	**1.48e+08**
	std	1.33e+10	2.41e+10	1.42e+10	5.21e+09	1.35e+10	9.53e+09	1.66e+10	1.28e+10	6.62e+08	7.70e+07
f_2	mean	1.10e+13 +	3.08e+13 +	8.88e+12 +	1.42e+12 +	6.93e+12 +	1.95e+13 +	2.49e+13 +	1.40e+13 +	2.96e+11 +	**1.32e+10**
	std	1.38e+12	2.17e+12	1.66e+12	4.22e+11	1.36e+12	1.23e+12	2.10e+12	1.89e+12	6.43e+10	7.51e+09
f_3	mean	3.60e+12 +	1.17e+13 +	3.01e+12 +	6.19e+11 +	2.36e+12 +	6.53e+12 +	8.79e+12 +	5.05e+12 +	1.08e+11 +	**5.66e+09**
	std	4.66e+11	8.57e+11	6.10e+11	1.68e+11	4.02e+11	4.80e+11	5.79e+11	6.98e+11	2.26e+10	3.17e+09
f_4	mean	2.51e+05 +	2.10e+07 +	7.88e+04 +	3.40e+03 +	1.41e+05 +	1.13e+06 +	3.30e+06 +	6.52e+05 +	2.15e+03 +	**2.08e+03**
	std	1.10e+05	6.64e+06	3.02e+04	1.13e+03	6.70e+04	4.05e+05	9.07e+05	2.63e+05	8.49e+01	4.84e+01
f_5	mean	5.82e+08 +	7.70e+08 +	2.89e+07 ≈	**2.78e+07** ≈	1.76e+08 +	3.72e+08 +	6.73e+08 +	1.83e+08 +	6.28e+07 +	3.37e+07
	std	3.39e+08	1.04e+08	6.80e+06	1.18e+07	4.42e+07	1.07e+08	2.03e+08	5.64e+07	2.08e+07	9.38e+06
f_6	mean	5.94e+08 +	3.43e+09 +	**3.24e+05** ≈	2.45e+06 +	8.17e+07 +	5.82e+09 +	1.30e+10 +	2.21e+09 +	4.35e+05 ≈	3.42e+05
	std	7.68e+08	1.24e+09	1.15e+05	2.37e+06	8.80e+07	2.34e+09	6.34e+09	1.69e+09	2.83e+05	2.66e+05
f_7	mean	4.42e+09 +	5.21e+09 +	3.55e+07 +	4.64e+07 +	8.66e+08 +	4.98e+09 +	1.32e+10 +	1.90e+09 +	5.42e+07 +	2.32e+07
	std	2.85e+09	1.48e+09	1.10e+07	3.28e+07	3.07e+08	1.64e+09	5.35e+09	6.33e+08	2.02e+07	**7.00e+06**
f_8	mean	8.74e+03 +	4.51e+03 +	3.54e+03 +	3.39e+03 ≈	1.86e+04 +	1.43e+04 +	1.97e+04 +	4.19e+03 +	**2.70e+03** −	3.26e+03
	std	1.73e+03	3.77e+02	2.52e+02	3.06e+03	5.17e+03	3.23e+03	3.54e+03	6.11e+02	5.94e+01	2.69e+03
f_9	mean	1.02e+05 +	1.16e+05 +	6.75e+04 +	2.73e+04 +	9.41e+04 +	1.45e+05 +	1.68e+05 +	1.41e+05 +	1.34e+04 +	**4.65e+03**
	std	1.02e+04	1.27e+04	8.73e+03	6.67e+03	1.98e+04	6.47e+03	6.61e+03	9.10e+03	1.17e+03	6.25e+02
f_{10}	mean	1.24e+04 +	4.19e+04 +	8.62e+03 +	4.49e+03 +	9.26e+03 +	1.87e+04 +	2.86e+04 +	1.46e+04 +	4.14e+03 +	**3.55e+03**
	std	1.60e+03	7.87e+03	1.11e+03	2.42e+02	1.05e+03	1.80e+03	3.36e+03	2.02e+03	1.65e+02	6.87e+01
+/≈/− summary:		10/0/0	10/0/0	8/2/0	8/2/0	10/0/0	10/0/0	10/0/0	10/0/0	8/1/1	
Ave. rank:		6.4	9.0	3.6	2.9	5.1	9.4	8.1	6.7	2.4	**1.4**

Table 6.8: Experimental and statistical results on engineering problems. Mean and std: the mean and the standard deviation of 30 trial runs; best and worst: the best and the worst final solution found among 30 trial runs.

Func.		PSO	DE	CMA-ES	GWO	AEO	ZOA	WSO	EVO	RIME	SRIME
WBP	mean	1.9782e+00 +	1.7491e+00 ≈	**1.6835e+00** −	1.8507e+00 ≈	2.0093e+00 +	2.6751e+00 +	2.3641e+00 +	2.7706e+00 +	1.9293e+00 +	1.7855e+00
	std	1.0231e-01	2.5584e-02	3.8103e-04	1.8332e-01	3.7077e-01	3.6408e-01	3.7174e-01	5.3400e-01	2.2111e-01	1.5360e-01
	best	1.8004e+00	1.7205e+00	1.6829e+00	1.6958e+00	1.6845e+00	1.9634e+00	1.7475e+00	1.9008e+00	1.7036e+00	1.6876e+00
	worst	2.1512e+00	1.8252e+00	1.6844e+00	2.3770e+00	3.0495e+00	3.6005e+00	3.2922e+00	4.1217e+00	2.8182e+00	2.3626e+00
SRD	mean	3.1382e+03 +	2.9883e+03 +	2.9871e+03 +	3.0691e+03 +	3.1745e+03 +	4.3262e+03 +	5.7932e+05 +	3.4347e+03 +	2.9963e+03 +	**2.9869e+03**
	std	5.2582e+01	1.6237e+00	4.9198e-02	5.6899e+01	5.1034e+02	6.0602e+02	1.3695e+06	3.3585e+02	7.0867e+00	5.6256e-04
	best	3.0381e+03	2.9870e+03	2.9870e+03	2.9983e+03	2.9881e+03	3.2484e+03	3.0268e+03	3.0100e+03	2.9876e+03	2.9869e+03
	worst	3.2474e+03	2.9955e+03	2.9872e+03	3.1891e+03	5.1007e+03	6.0877e+03	5.7820e+06	4.2829e+03	3.0180e+03	2.9869e+03
CBD	mean	2.1032e+00 +	1.3615e+00 +	**1.3420e+00** −	1.3585e+00 ≈	1.4397e+00 +	1.4426e+00 +	1.4467e+00 +	2.1672e+00 +	1.3574e+00 ≈	1.3503e+00
	std	2.8919e-01	6.1573e-03	1.2834e-03	2.2945e-02	7.8800e-02	6.6320e-02	5.7351e-02	4.5218e-01	1.5191e-02	8.7103e-03
	best	1.6347e+00	1.3498e+00	1.3402e+00	1.3410e+00	1.3426e+00	1.3551e+00	1.3590e+00	1.4512e+00	1.3409e+00	1.3419e+00
	worst	2.8433e+00	1.3711e+00	1.3455e+00	1.4311e+00	1.6181e+00	1.6095e+00	1.6384e+00	3.2013e+00	1.4100e+00	1.3853e+00
IBD	mean	1.8405e-04 +	1.7458e-04 ≈	**1.7458e-04** ≈	1.7458e-04 ≈	**1.7458e-04** ≈	1.7588e-04 +	1.7531e-04 +	1.9790e-04 +	1.7458e-04 ≈	1.7458e-04
	std	3.7286e-06	3.9501e-15	1.3553e-19	1.6823e-11	1.3553e-19	4.0393e-06	5.5047e-07	1.9463e-05	1.1196e-09	1.4328e-10
	best	1.7779e-04	1.7458e-04	1.7458e-04	1.7458e-04	1.7458e-04	1.7458e-04	1.7462e-04	1.7459e-04	1.7458e-04	1.7458e-04
	worst	1.9246e-04	1.7458e-04	1.7458e-04	1.7458e-04	1.7458e-04	1.9339e-04	1.7610e-04	2.6354e-04	1.7459e-04	1.7458e-04
TCD	mean	3.0291e+01 +	3.0150e+01 +	3.0150e+01 +	3.0162e+01 +	3.0164e+01 +	3.0210e+01 +	3.0495e+01 +	3.0812e+01 +	3.0328e+01 +	**3.0150e+01**
	std	7.6171e-02	4.4080e-05	2.0787e-09	1.4541e-02	1.4648e-02	8.5008e-02	1.4293e-01	4.2575e-01	2.0913e-01	6.0611e-12
	best	3.0187e+01	3.0150e+01	3.0150e+01	3.0150e+01	3.0150e+01	3.0150e+01	3.0152e+01	3.0176e+01	3.0151e+01	3.0150e+01
	worst	3.0467e+01	3.0150e+01	3.0150e+01	3.0205e+01	3.0213e+01	3.0498e+01	3.0781e+01	3.1774e+01	3.0857e+01	3.0150e+01
PLD	mean	8.5114e+01 +	1.0574e+00 ≈	1.0575e+00 +	6.8752e+01 +	1.0698e+02 +	3.2786e+01 +	8.3341e+00 +	7.6292e+02 +	2.2975e+01 +	**1.0574e+00**
	std	3.7737e+01	8.1546e-07	6.4088e-05	1.0936e+02	1.1310e+02	7.5890e+01	6.4679e+00	7.1836e+02	6.5686e+01	6.0867e-07
	best	2.1552e+01	1.0574e+00	1.0574e+00	1.0579e+00	1.0577e+00	4.3765e+00	1.9064e+00	7.7311e+01	1.0623e+00	1.0574e+00
	worst	1.8491e+02	1.0574e+00	1.0576e+00	3.6459e+02	3.1880e+02	3.3513e+02	2.6399e+01	3.0660e+03	2.2998e+02	1.0574e+00
CBHD	mean	7.3600e+00 +	6.8632e+00 +	**6.8437e+00** −	7.0239e+00 +	6.9846e+00 +	7.1027e+00 +	7.3903e+00 +	8.9145e+00 +	6.8507e+00 ≈	6.8495e+00
	std	2.3485e-01	9.4051e-03	2.4452e-04	1.2506e-01	9.0480e-02	1.5523e-01	2.3715e-01	8.2938e-01	4.9921e-03	5.3934e-03
	best	6.9522e+00	6.8482e+00	6.8433e+00	6.8721e+00	6.8465e+00	6.9030e+00	7.0632e+00	7.2500e+00	6.8440e+00	6.8436e+00
	worst	7.8107e+00	6.8826e+00	6.8443e+00	7.3503e+00	7.1372e+00	7.4518e+00	8.1239e+00	1.0978e+01	6.8638e+00	6.8661e+00
RCB	mean	1.6174e+02 +	1.6662e+02 +	1.6662e+02 +	1.6693e+02 +	1.6636e+02 +	1.6701e+02 +	1.6750e+02 +	1.6869e+02 +	1.6550e+02 +	**1.5973e+02**
	std	2.0897e+00	6.0548e-01	6.0548e-01	9.9594e-01	1.2671e+00	1.0834e+00	1.3314e+00	2.0974e+00	1.3240e+00	3.6992e-01
	best	1.5936e+02	1.6356e+02	1.6356e+02	1.6356e+02	1.6055e+02	1.6356e+02	1.6356e+02	1.6356e+02	1.6386e+02	1.5937e+02
	worst	1.6677e+02	1.6677e+02	1.6677e+02	1.6844e+02	1.6677e+02	1.7021e+02	1.7142e+02	1.7306e+02	1.6713e+02	1.6115e+02
+/≈/− summary:		8/0/0	5/3/0	4/1/3	5/3/0	7/1/0	8/0/0	8/0/0	8/0/0	5/3/0	
Ave. rank		6.8	3.5	**2.0**	5.1	5.6	7.5	8.0	9.8	4.6	**2.0**

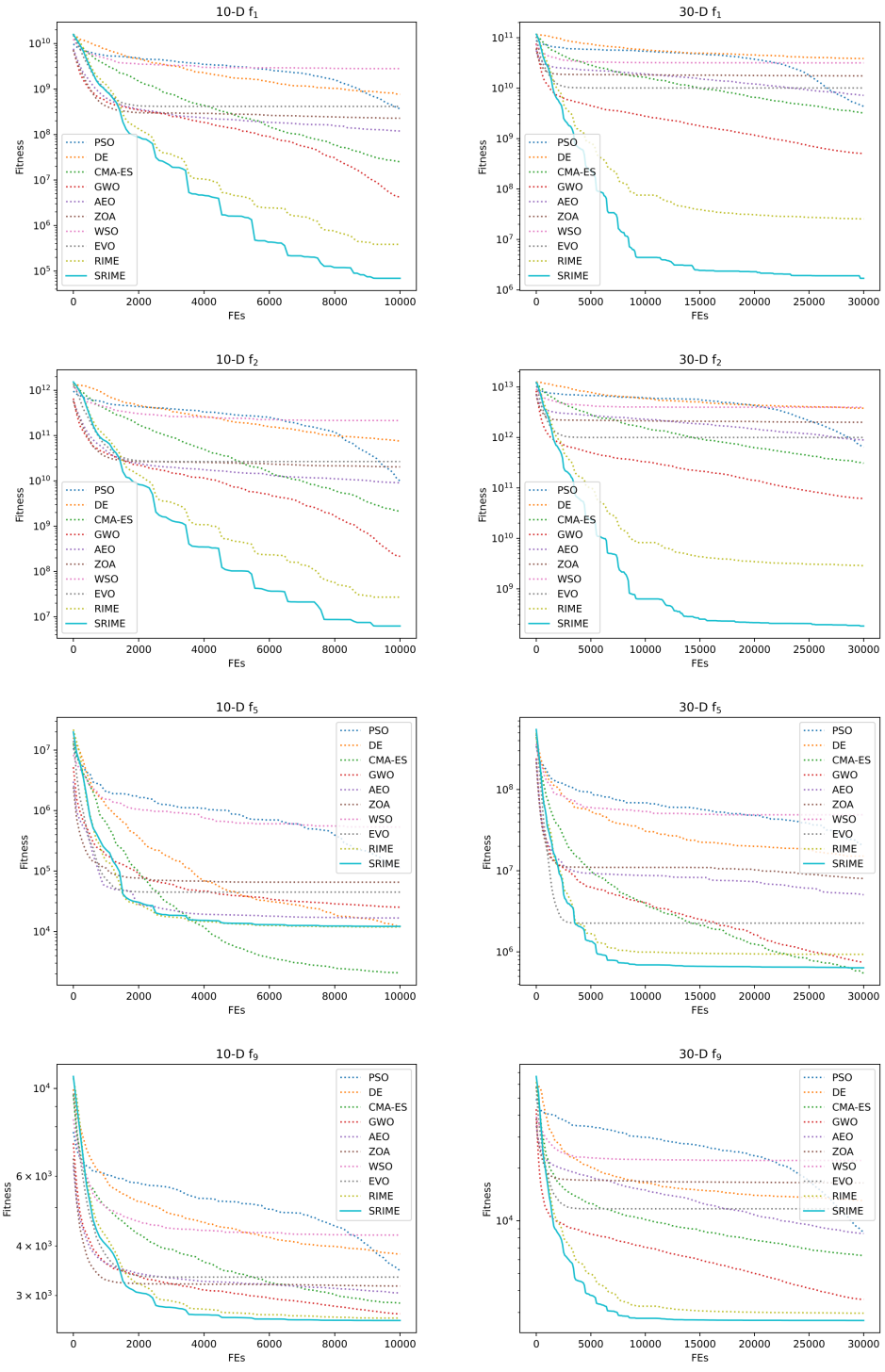


Figure 6.8: The convergence curve of 50-D and 100-D f_1, f_2, f_5, and f_9 in CEC2020 benchmark functions.

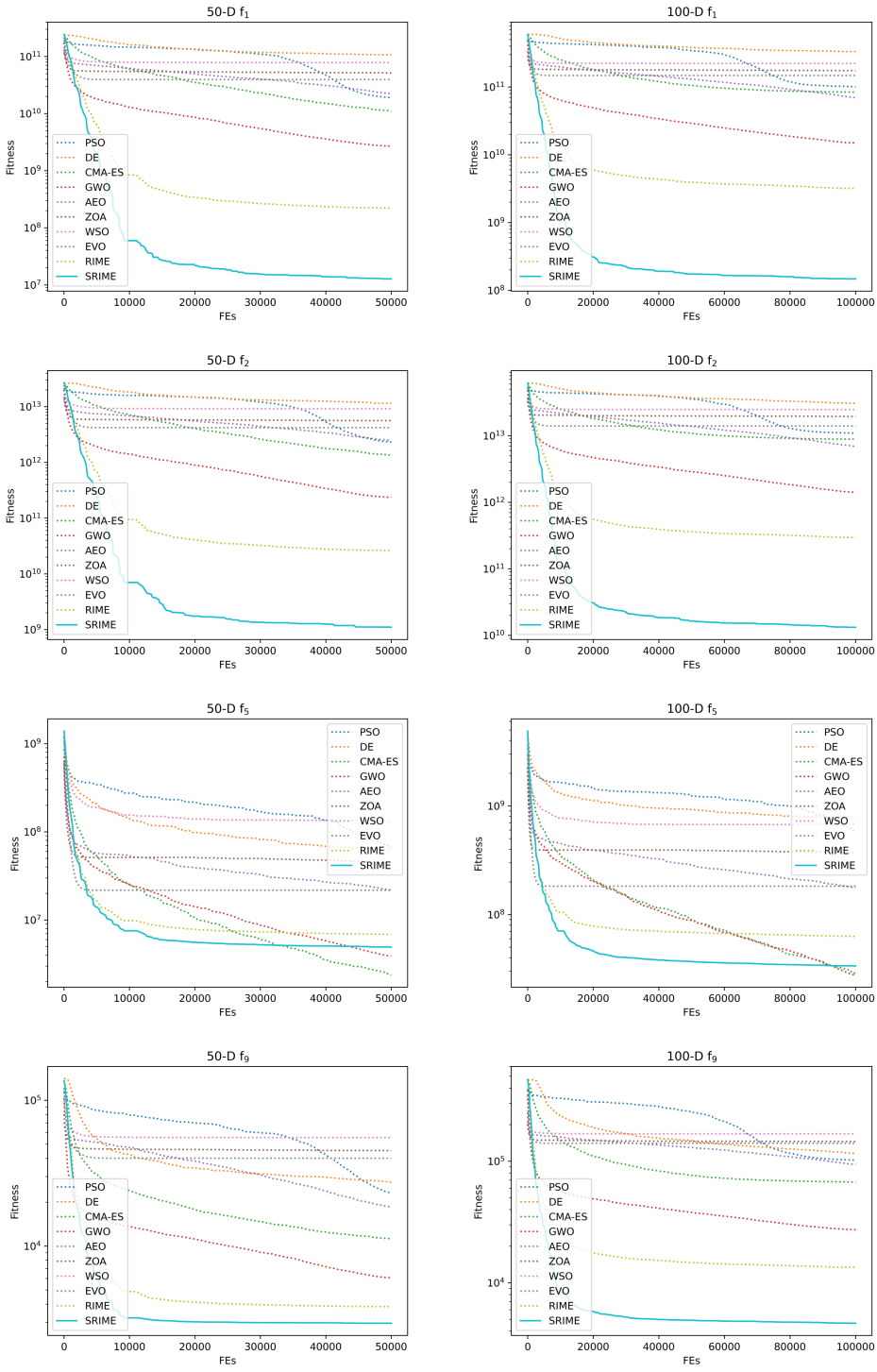


Figure 6.9: The convergence curve of 10-D and 30-D f_1, f_2, f_5, and f_9 in CEC2020 benchmark functions.

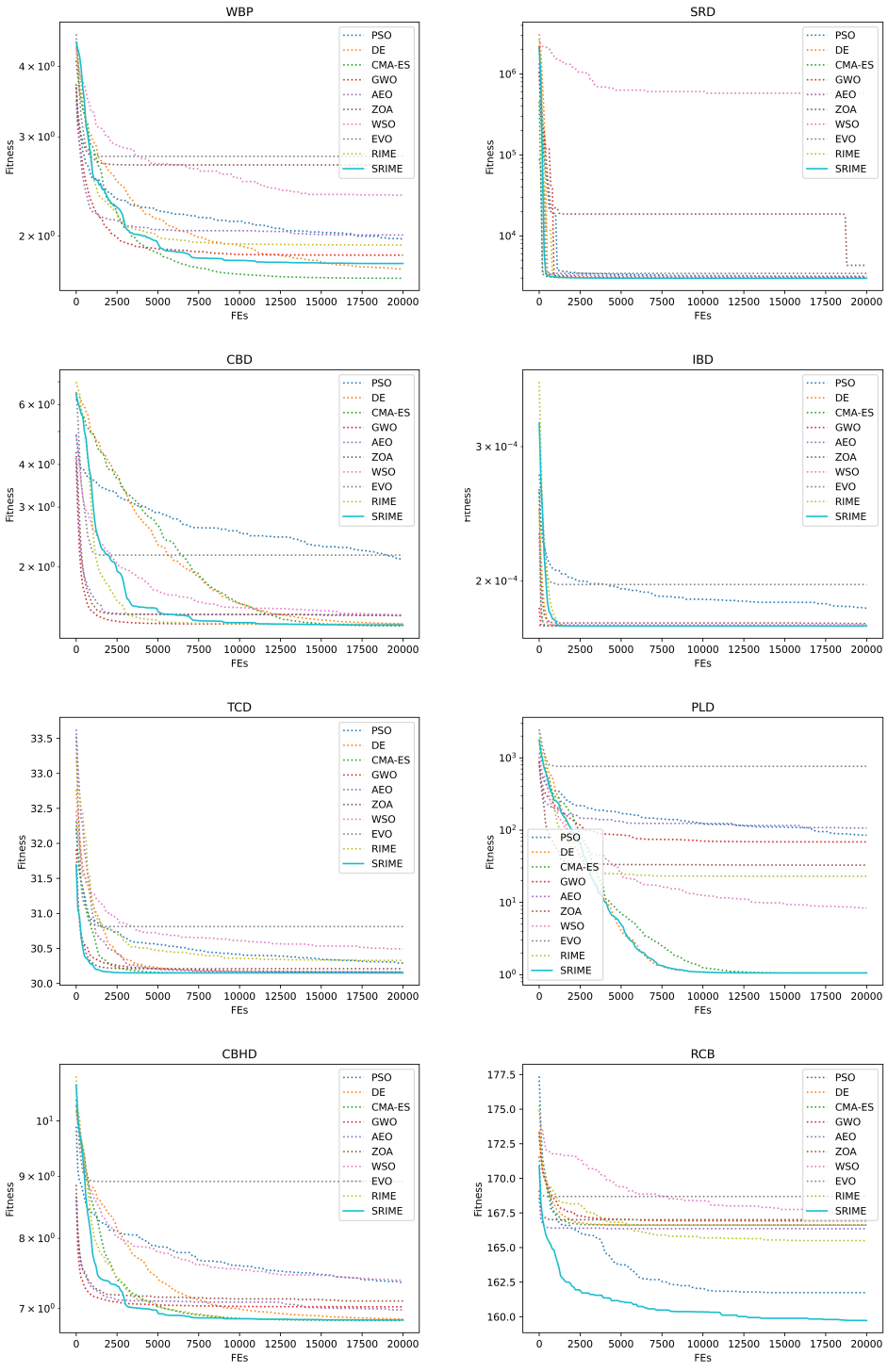


Figure 6.10: Convergence curves of eight engineering optimization problems.

AEO, ZOA, WSO, EVO, and the original RIME in f_1 with distinct dimensions. As the dimension of the problem increases, the superiority of SRIME is amplified, and the dom-

Table 6.9: Ablation experiments on 10-D CEC2020 benchmark functions.

Func.		RIME	RIME$_{lhs}$	RIME$_{mhs}$	RIME$_{edbs}$	SRIME
f_1	mean	3.86e+05 +	6.08e+05 +	6.31e+05 +	9.29e+04 ≈	**6.93e+04**
	std	4.57e+05	8.42e+05	9.81e+05	1.39e+05	1.17e+05
f_2	mean	2.72e+07 +	2.90e+07 +	3.97e+07 +	1.28e+07 +	**6.25e+06**
	std	2.44e+07	2.89e+07	4.93e+07	2.40e+07	1.01e+07
f_3	mean	1.18e+07 +	1.53e+07 +	1.43e+07 +	6.22e+06 +	**3.13e+06**
	std	1.23e+07	2.00e+07	1.54e+07	7.97e+06	3.53e+06
f_4	mean	1.90e+03 ≈	1.90e+03 ≈	1.90e+03 ≈	1.90e+03 ≈	**1.90e+03**
	std	9.00e-01	5.89e-01	5.71e-01	7.13e-01	6.29e-01
f_5	mean	**1.19e+04** ≈	1.37e+04 ≈	1.42e+04 ≈	1.58e+04 ≈	1.22e+04
	std	7.46e+03	7.73e+03	8.44e+03	8.71e+03	8.46e+03
f_6	mean	3.27e+03 ≈	2.72e+03 ≈	2.86e+03 ≈	2.72e+03 ≈	**2.25e+03**
	std	2.52e+03	1.91e+03	2.11e+03	1.58e+03	1.42e+03
f_7	mean	**1.83e+04** ≈	2.01e+04 ≈	2.10e+04 ≈	2.28e+04 ≈	2.13e+04
	std	1.04e+04	1.48e+04	1.28e+04	1.31e+04	1.61e+04
f_8	mean	2.30e+03 +	2.30e+03 +	2.30e+03 +	2.30e+03 +	**2.30e+03**
	std	2.15e+00	1.66e+00	2.46e+00	1.68e+00	1.43e+00
f_9	mean	2.63e+03 +	2.63e+03 +	2.62e+03 +	2.61e+03 +	**2.59e+03**
	std	4.07e+01	4.77e+01	2.85e+01	5.35e+01	3.38e+01
f_{10}	mean	2.99e+03 ≈	3.00e+03 ≈	**2.99e+03** ≈	3.00e+03 ≈	2.99e+03
	std	2.15e+01	2.72e+01	1.15e+01	2.48e+01	1.99e+01
+/≈/− summary:		5/5/0	5/5/0	5/5/0	4/6/0	
Ave. rank		3.3	3.5	3.6	3.0	**1.6**

ination between SRIME and other competitor algorithms is not changed, which can be observed from statistical analysis in Table 6.4, 6.5, 6.6, and 6.7. The curves of optimization in Figure 6.8 also confirm the excellent convergence speed of SRIME. Therefore, we can conclude that SRIME has a strong exploitation ability.

f_2 to f_{10} in CEC2020 benchmark functions are multi-modal, hybrid, and composite functions, these functions have multiple optima and complex fitness landscapes so that they are allowed to evaluate the exploration ability and the capacity of escaping from local optima. Experimental and statistical results in Table 6.4, 6.5, 6.6, and 6.7 show that our proposed SRIME has better performance on most of instances, and we can infer that the RIME structure-based optimization technique (i.e. RIME and SRIME) is competitive to

Table 6.10: Ablation experiments on 30-D CEC2020 benchmark functions.

Func.		RIME	RIME$_{lhs}$	RIME$_{mhs}$	RIME$_{edbs}$	SRIME
f_1	mean	2.56e+07 +	2.52e+07 +	2.54e+07 +	2.46e+06 ≈	**1.70e+06**
	std	1.06e+07	8.72e+06	1.12e+07	1.90e+06	1.55e+06
f_2	mean	2.88e+09 +	2.65e+09 +	2.75e+09 +	2.62e+08 ≈	**1.84e+08**
	std	1.16e+09	1.21e+09	1.12e+09	2.95e+08	2.12e+08
f_3	mean	8.73e+08 +	9.36e+08 +	1.00e+09 +	**7.85e+07** ≈	9.07e+07
	std	4.15e+08	5.00e+08	4.47e+08	6.70e+07	1.24e+08
f_4	mean	1.92e+03 +	1.92e+03 +	1.92e+03 +	1.92e+03 +	**1.91e+03**
	std	3.54e+00	4.67e+00	4.76e+00	4.74e+00	2.57e+00
f_5	mean	9.33e+05 ≈	7.11e+05 ≈	1.02e+06 ≈	7.61e+05 ≈	**6.37e+05**
	std	6.25e+05	4.44e+05	6.42e+05	4.35e+05	3.56e+05
f_6	mean	4.87e+04 ≈	5.80e+04 +	3.87e+04 ≈	4.52e+04 ≈	**3.82e+04**
	std	1.91e+04	2.48e+04	1.99e+04	2.32e+04	2.25e+04
f_7	mean	1.40e+06 ≈	1.50e+06 ≈	1.24e+06 ≈	1.26e+06 ≈	**1.07e+06**
	std	6.66e+05	8.47e+05	7.63e+05	6.78e+05	7.06e+05
f_8	mean	2.40e+03 +	2.40e+03 +	2.40e+03 +	2.39e+03 ≈	**2.38e+03**
	std	1.39e+01	1.06e+01	1.24e+01	8.83e+00	8.46e+00
f_9	mean	2.97e+03 +	2.98e+03 +	2.98e+03 +	2.70e+03 ≈	**2.70e+03**
	std	8.47e+01	9.23e+01	9.62e+01	9.31e+01	7.07e+01
f_{10}	mean	2.95e+03 +	2.94e+03 +	2.96e+03 +	2.93e+03 ≈	**2.93e+03**
	std	2.85e+01	2.17e+01	3.54e+01	1.07e+01	1.51e+01
+/≈/− summary:		7/3/0	8/2/0	7/3/0	1/9/0	
Ave. rank		4.0	3.8	3.9	2.2	**1.1**

the state-of-the-art EAs. Besides, the introduction of our proposed three improvements can at least approximately equal to or significantly accelerate the convergence of RIME in most cases. As the dimension of the problem increases, the superiority of SRIME is enhanced, which can be observed from the performance indicator of average rank. Consequently, we conclude that the incorporation of RIME and our proposed three strategies is suitable to deal with these optimization problems.

However, the significant deterioration situations between RIME and SRIME happen on 50-D and 100-D f_{10}, and we reasonably infer that our proposed three techniques are not proper for this specific task. In this case, the NFLT can be applied to explain this degeneration. NFLT states that every pair of optimization algorithms has identical average

Table 6.11: Ablation experiments on 50-D CEC2020 benchmark functions.

Func.		RIME	RIME$_{lhs}$	RIME$_{mhs}$	RIME$_{edbs}$	SRIME
f_1	mean	2.22e+08 +	2.02e+08 +	2.12e+08 +	1.76e+07 ≈	**1.28e+07**
	std	6.35e+07	5.67e+07	7.42e+07	1.64e+07	8.72e+06
f_2	mean	2.63e+10 +	2.63e+10 +	2.50e+10 +	2.17e+09 +	**1.10e+09**
	std	8.67e+09	8.48e+09	7.95e+09	1.13e+09	5.87e+08
f_3	mean	8.56e+09 +	8.76e+09 +	7.37e+09 +	5.71e+08 +	**3.69e+08**
	std	2.52e+09	2.32e+09	2.46e+09	3.62e+08	2.32e+08
f_4	mean	1.95e+03 +	1.96e+03 +	1.96e+03 +	1.94e+03 +	**1.93e+03**
	std	1.10e+01	1.21e+01	1.16e+01	8.01e+00	8.32e+00
f_5	mean	6.89e+06 +	7.09e+06 +	6.45e+06 ≈	4.98e+06 ≈	**4.93e+06**
	std	2.84e+06	2.88e+06	3.09e+06	1.95e+06	2.44e+06
f_6	mean	4.49e+05 +	3.87e+05 +	4.09e+05 +	**6.93e+04** ≈	7.91e+04
	std	4.49e+05	3.69e+05	3.98e+05	2.76e+04	5.97e+04
f_7	mean	8.38e+06 ≈	9.62e+06 +	9.54e+06 +	8.46e+06 ≈	**6.14e+06**
	std	4.46e+06	4.32e+06	3.87e+06	4.02e+06	2.69e+06
f_8	mean	2.52e+03 −	**2.52e+03** −	2.71e+03 +	2.62e+03 ≈	2.63e+03
	std	2.33e+01	2.61e+01	1.01e+03	7.77e+02	8.87e+02
f_9	mean	3.85e+03 +	3.83e+03 +	3.81e+03 +	**2.96e+03** ≈	2.96e+03
	std	2.26e+02	1.93e+02	2.34e+02	1.91e+02	2.41e+02
f_{10}	mean	3.71e+03 +	3.56e+03 +	3.70e+03 +	3.46e+03 +	**3.41e+03**
	std	1.85e+02	1.42e+02	1.87e+02	9.75e+01	1.00e+02
+/≈/− summary:		8/1/1	9/0/1	9/1/0	4/6/0	
Ave. rank		4.0	3.8	3.7	2.0	**1.5**

performance on all possible problems, and if an algorithm performs better on a certain category problem, it must deteriorate on the rest of the problems, since it is the only way to achieve the same average performance. Thus, the inferiority of SRIME in 50-D and 100-D f_{10} is acceptable.

6.5.2 Performance analysis based on engineering problems

In real-world applications, another characteristic of the optimization algorithm that we are concerned about is stability. If an algorithm has an outstanding average value of trial runs but the standard deviation is also large, it cannot be regarded as a successful approach in real-world scenarios, since the real-world optimization problem is usually computationally

Table 6.12: Ablation experiments on 100-D CEC2020 benchmark functions.

Func.		RIME	RIME$_{lhs}$	RIME$_{mhs}$	RIME$_{edbs}$	SRIME
f_1	mean	3.19e+09 +	2.97e+09 +	3.00e+09 +	1.85e+08 ≈	**1.48e+08**
	std	6.62e+08	6.92e+08	9.12e+08	9.44e+07	7.70e+07
f_2	mean	2.96e+11 +	2.97e+11 +	2.80e+11 +	2.08e+10 +	**1.32e+10**
	std	6.43e+10	5.92e+10	6.21e+10	8.75e+09	7.51e+09
f_3	mean	1.08e+11 +	1.06e+11 +	1.10e+11 +	8.41e+09 +	**5.66e+09**
	std	2.26e+10	1.92e+10	2.58e+10	4.21e+09	3.17e+09
f_4	mean	2.15e+03 +	2.16e+03 +	2.18e+03 +	**2.06e+03** ≈	2.08e+03
	std	8.49e+01	9.47e+01	1.21e+02	2.59e+01	4.84e+01
f_5	mean	6.28e+07 +	7.00e+07 +	7.58e+07 +	3.86e+07 ≈	**3.37e+07**
	std	2.08e+07	2.32e+07	2.40e+07	8.23e+06	9.38e+06
f_6	mean	4.35e+05 ≈	6.24e+05 ≈	7.27e+05 ≈	**2.33e+05** ≈	3.42e+05
	std	2.83e+05	7.00e+05	7.98e+05	8.90e+04	2.66e+05
f_7	mean	5.42e+07 +	5.96e+07 +	5.56e+07 +	2.39e+07 ≈	**2.32e+07**
	std	2.02e+07	2.43e+07	2.14e+07	1.09e+07	7.00e+06
f_8	mean	**2.70e+03** −	3.28e+03 +	4.37e+03 +	3.23e+03 ≈	3.26e+03
	std	5.94e+01	3.19e+03	5.02e+03	2.50e+03	2.69e+03
f_9	mean	1.34e+04 +	1.31e+04 +	1.26e+04 +	5.02e+03 +	**4.65e+03**
	std	1.17e+03	1.53e+03	1.36e+03	6.17e+02	6.25e+02
f_{10}	mean	4.14e+03 +	4.10e+03 +	4.14e+03 +	3.68e+03 +	**3.55e+03**
	std	1.65e+02	1.79e+02	1.98e+02	1.04e+02	6.87e+01
+/≈/− summary:		8/1/1	9/1/0	9/1/0	4/6/0	
Ave. rank		3.5	3.9	4.4	1.8	**1.4**

expensive, and multiple trial runs may not be afforded due to the limitation of computational resources. Table 6.8 shows the results of eight engineering optimization problems including mean, std, best, worst, and statistical tests, which can fully reflect the performance of our proposed SRIME and compared algorithm. From these results, we summarize that our proposed SRIME is competitive with competitor algorithms, and the optimization results in the Speed Reducer Problem (SRD), Tubular Column Problem (TCD), Piston Lever Problem (PLD), and Reinforced Concrete Beam Problem (RCB) reveal the excellent global search capacity and stability of SRIME. Additionally, the metric of average rank reveals the competitiveness of SRIME with state-of-the-art EAs such as CMA-ES. However, the inferiority of SRIME can be observed in some specific tasks such as in the Welded Beam

Problem (WBP) and Cantilever Beam Problem (CBD) compared with CMA-ES. Similarly, the NFLT can also be used to explain this degeneration reasonably.

6.5.3 Performance analysis based on ablation experiments

The ablation experiments are implemented to evaluate our proposed three strategies independently. From the experimental and statistical results in Table 6.9, 6.10, 6.11, and 6.12, the combination of three strategies can accelerate the convergence of optimization in most cases. Based on the summarized statistical results, the embedded distance-based selection plays the most important role in contributing to the improvement.

6.6 Conclusion

This chapter focuses on improving the overall optimization performance of RIME and proposes a strengthened RIME (SRIME). Three search strategies are introduced to the original RIME algorithm: (1). Latin hypercube initialization, (2). Modified hard rime search scheme, and (3). Embedded distance-based selection mechanism. To evaluate the performance of our proposed SRIME, we test it in CEC2020 benchmark functions and eight engineering optimization problems, the experimental and statistical results reveal the efficiency and effectiveness of our proposed SRIME. Ablation experiments are also provided to analyze the performance of each component in our proposed three strategies.

Chapter 7

Evolutionary multi-mode slime mold optimization

This chapter presents an evolutionary multi-mode slime mold optimization[1]. Section 7.1 presents the motivation of this research, Section 7.2 details our proposal: EMSMO, Section 7.3 presents the experiments on the CEC benchmark functions, and Section 7.4 discusses the performance of EMSMO. Finally, Section 7.5 concludes this chapter.

7.1 Motivation

As the highest-level component of the nervous system, the brain is commonly regarded as the foundation of intelligence. However, the slime mold, as a kind of single-cell organism without a brain, can perform a certain level of swarm intelligence: Nakagaki et al.[184] found that the physarum polycephalum, one category of slime mold, can find the minimum route in a complex maze with the nutrition guide. Tero et al.[185] adopted the slime mold to design the traffic network of the Tokyo area, food sources are placed in 36 geographical locations

[1]This chapter was previously published as Zhong, R., Zhang, E. & Munetomo, M. Evolutionary multi-mode slime mold optimization: a hyper-heuristic algorithm inspired by slime mold foraging behaviors. J Supercomput 80, 12186–12217 (2024). https://doi.org/10.1007/s11227-024-05909-0. Reproduced with permission from Springer Nature.[183]

of cities on the Tokyo map. As the slime mold grows gradually, the topology of the growth of slime mold is highly similar to the real traffic network. Adamatzky et al.[186] conducted similar experiments to approximate the real motorway networks in 14 countries by slime mold and got inspiring experimental results. Boussard et al.[187] considered that although the inner mechanism remains unknown, slime mold can utilize the information from previous experiences to modify its behaviors adaptively, and the oscillations may play an important role in learning and memory of the slime mold. These works reveal that even the simplest organismic structure of slime mold has incredible intelligence. Therefore, inspired by the collective intelligence of slime mold, we propose a novel hyper-heuristic algorithm named evolutionary multi-mode slime mold optimization. Figure 7.1 shows the foraging behavior of slime mold.

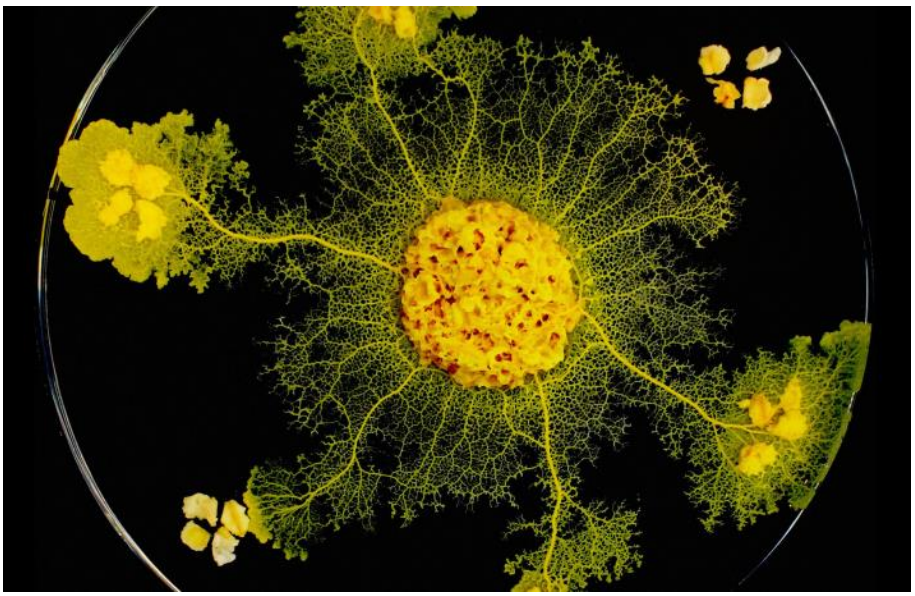


Figure 7.1: The foraging behavior of slime mold.[5]

7.2 Proposal: EMSMO

In the initialization stage, the physarum polycephalum slime mold is placed randomly so that it may be far away from the food source, and the search behavior to unexplored re-

gions is mainly adopted. When the slime mold approaches the food, the slight perturbation of movement allows careful exploitation to reach more nutritious locations. And when the slime mold is very close to the food source such as oat flakes, it will wrap the oat flakes and ingest them. However, when a certain direction of the slime mold cannot obtain sufficient nutrition to survive, the cells in this route will degenerate and die. Inspired by the slime mold behaviors, we design four search strategies to simplify the foraging behavior and realize the LLHs of our proposed EMSMO: the search for food, approach food, wrap food, and re-initialization. A simple demonstration of these strategies is visualized in Figure 7.2. Blue points denote the solutions, orange dotted lines with arrows represent the moving distances and directions, the red star means the optimal solution and the orange point is the estimated solution by wrap food operation. The detailed introduction is in the following sections. Noticing that all explanations are under the background of the minimization problem.

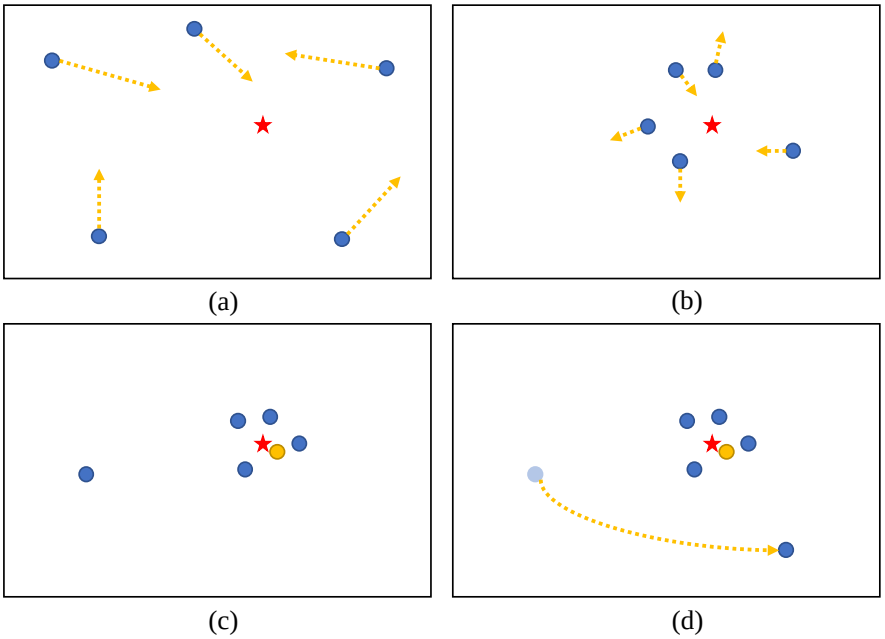


Figure 7.2: The simplified search strategies of EMSMO. (a). search for food. (b). approach food. (c). wrap food. (d). re-initialization.

7.2.1 Low-level heuristics

Search for food: Differential-based search strategy has been widely adopted to describe the foraging behaviors of the natural organism.[66,188–191] Similarly, we adopt the unbiased combination of strategies in Eq. (7.1) to realize the Search for food operator.

$$X_{i+1} = \begin{cases} X_i + a_1(X_{best} - X_i) + a_2(X_{r1} - X_{r2}) \\ X_{best} + a_3(X_{r1} - X_{r2}) \end{cases} \tag{7.1}$$

where X_i represents the i^{th} solution, X_{best} is the current best solution. $r1$ and $r2$ are two mutually different random values. a_1, a_2, and a_3 are randomly generated from a uniform distribution, which can be calculated by Eq. (7.2).

$$\begin{aligned} A &= \operatorname{arc}\tanh(1 - t/T) + 0.1 \\ a_i &\sim U(-A, A) \end{aligned} \tag{7.2}$$

t and T are the current and maximum generation of optimization respectively, and a_i is a self-adaptive parameter. Supposing the maximum iteration time is 100, and the range of a_i will decrease from 2.75 to 0.1 non-linearly. Figure 7.2 (a) demonstrates this Search for food operator. Furthermore, the benefit of this differential-based search strategy is that the differential vector between any pairwise solutions constructs a share of the population distribution, which contains a part of the information of distribution. Thus, a differential vector can represent the movement direction of the slime mold.

Approach food: When the slime mold approaches oatmeal, the bio-oscillator produces a propagating wave and the vein organism becomes veinless or network structures, and the growth speed also degenerates.[192] Therefore, we consider this behavior corresponding to the exploitation operator in algorithm design and formulate it by Eq. (7.3).

$$X_{i+1} = X_{mean} + R \cdot D \tag{7.3}$$

where X_{mean} represents the mean location of best-k solutions, and k is randomly sampled from 1 to $N/2$. R is the growth radius of the slime mold, D uniformly sampled from $[-1, 1]$ denotes the extension direction. An example is shown in Figure 7.3 (b).

Wrap food: The slime mold modifies the growth status when reaching the food source and wraps around the surfaces of the oat flakes. Since the contact of the slime mode with the oat flakes is large, it is advantageous for the intake of nutrition to diffuse from the surface of the food source.[184] Figure 7.2 (c) shows a demonstration of the wrap food operation in optimization. To further explain the inspiration for the Wrap food operator design, Figure 7.3 simulates the food intake process of slime mold.

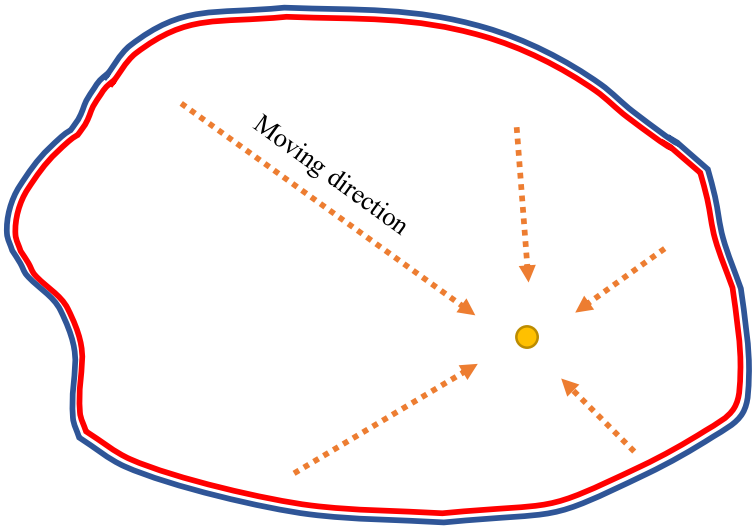


Figure 7.3: A demonstration of the wrap food operator in 2-D space.

In the real world, food sources such as oatmeal have a definite volume and cannot be regarded as a point of optimum simply, and the area limited by the red circle in Figure 7.3 represents the food source, the blue cycle denotes the slime mold which contacts the food source tightly. The distribution of ingredients in the food source is not uniform so the slime mold has the preference to absorb the food in a certain direction, which corresponds to the concept of fitness in the optimization. Therefore, slime mold can estimate the most nutritious region in the food source area with the biological mechanism, which contributes to the wrap food operator design, and this process can also show a certain level of swarm

intelligence of slime mold.

Based on the above explanation, the mathematical model of the Wrap food operator is formulated in Eq. (7.4).

$$X_{i+1} = \sum_{j=1}^{k} w_j X_j, \ s.t. \sum_{j=1}^{k} w_j = 1 \tag{7.4}$$

where w_j is the corresponding weight to solution X_j, which can be calculated by Eq. (7.5).

$$w_j = \frac{1/F_j}{\sum_{t=1}^{k} 1/F_t} \tag{7.5}$$

F_j is the objective value of solution j.

Re-initialization: In the real-world slime mold, the search direction that fails to find the food will shrink back and leave a trail to avoid re-entry. Similarly, as Figure 7.2 (d) shows a solution is in a poor search direction, which has a probability of being re-initialized. And the re-initialization operation can be described by Eq. (7.6).

$$X_{i+1}^{d} = LB^d + r(UB^d - LB^d) \tag{7.6}$$

where LB^d and UB^d represent the lower bound and upper bound of search space in d^{th} dimension, r is a random value from 0 to 1.

7.2.2 High-level component

The high-level component of EMSMO acts as the brain in the HHs algorithm to manipulate the LLHs. In our proposed EMSMO, we design an improvement-based probabilistic selection function to support the construction of the optimization sequence, which consists of two metrics: the probability of improvement (PI) and the normalized improvement (NI).

Probability of improvement (PI): PI is a metric to describe the probability of a specific search strategy to construct better solutions. Simply, for a search strategy A, PI_A

can be calculated by Eq. (7.7).

$$PI_A = \frac{Times\ of\ A\ achieving\ improved\ solution}{Times\ of\ A\ being\ adopted + \varepsilon} \tag{7.7}$$

ε is a tiny value to avoid the zero-division error. An example is that if search strategy A is chosen ten times and five improved solutions are obtained, PI_A is approximately equal to 0.5.

Normalized improvement (NI): NI is a quantitative metric to describe the certainty of improvement. Considering that the absolute improvement between the early and late stages of optimization will have a huge difference (i.e. Commonly the improvement of solutions in the early stage is large and in the late stage is relatively tiny), it is unfair to compare the improvement directly. To eliminate the effect of dimension, we normalize the improvement in every generation of optimization. Similarly, the NI of strategy A can be computed by Eq. (7.8).

$$NI_A = \frac{\sum_A \max(0, f(O_A) - f(P_A))}{\varepsilon + \sum_{i=1}^{|Pop|} \max(0, f(O_i) - f(P_i))} \tag{7.8}$$

where $|Pop|$ denotes the population size, $f(\cdot)$ means the objective function. O_i and P_i represent the i^{th} offspring and parent solution respectively. And the corresponding score of strategy A can be defined in Eq. (7.9).

$$Score_A = w \cdot PI_A + (1 - w) \cdot NI_A \tag{7.9}$$

where w is set to 0.5 in our experiments. Figure 7.4 shows the structure of the score matrix. A, B, C, and D correspond to our designed four LLHs. The values in a row represent the scores of a certain strategy in the continuous generation of optimization, and values in a column denote the scores of the corresponding strategies in a generation, each value in the score matrix is calculated by Eq. (7.9). We sum up all previous scores of a certain strategy

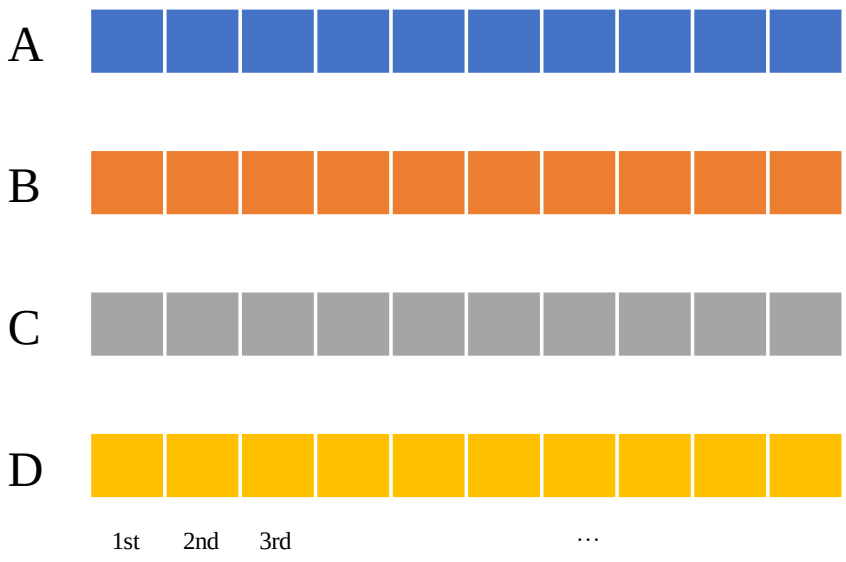


Figure 7.4: The structure of score matrix.

in the score matrix to contribute to the strategy selection in the next generation. Roulette wheel strategy (RWS)[193] is applied to choose the search strategy for each individual to construct the offspring, and the chosen probability of strategy A can be calculated by Eq. (7.10).

$$P_A = \frac{Score_A}{\sum_{i=\{A,B,C,D\}} Score_i} \tag{7.10}$$

In summary, the pseudocode of EMSMO is shown in Algorithm 7.5.

7.3 Numerical experiments

In this section, we implement a set of experiments to evaluate our proposed EMSMO. Section 7.3.1 introduces the experiment settings, and section 7.3.2 shows the experimental results.

Algorithm EMSMO

Require: Population size: S, Dimension: D, Maximum iteration: T, Lower and upper bound: LB, UB

Ensure: Optimum: O

$X \leftarrow$ **initialPop**(S, D, LB, UB) # Population initialization
$O \leftarrow$ **best**(X)
Initialize score matrix SM
while $t = 0$ and $t < T$ **do**
    Sum up the previous scores
    for $i = 0$ to S **do**
        Select a search strategy by RWS
        Construct offspring OS_i for X_i
        Update X by OS with greedy survival
    end for
    Update the score matrix
    $O \leftarrow$ **best**(X)
    $t \leftarrow t + 1$
end while
return O

Figure 7.5: The pseudocode of EMSMO.

7.3.1 Experiment settings

Benchmark functions: We evaluate the performance of EMSMO on the CEC2013 benchmark functions and three engineering optimization problems. The CEC2013 suite contains 29 functions with various characteristics such as multimodality, asymmetry, nonseparability, etc. We adopt 10-D, 30-D, and 50-D CEC2013 benchmark functions to evaluate EMSMO. Besides, the explanation of three engineering optimization problems is as follows:

Corrugated Bulkhead Design (CBD): This problem aims to minimize the weight of a corrugated bulkhead in a chemical tanker,[194] where the decision variables are the width (x_1), depth (x_2), length (x_3), and plate thickness (x_4). The mathematical model of this

optimization problem is given as follows.

$$
\begin{aligned}
&\text{minimize} \\
&f(X) = \frac{5.885x_4(x_1 + x_3)}{x_1 + \sqrt{|x_3^2 - x_2^2|}} \\
&\text{subject to} \\
&g_1(X) = -x_4x_2(0.4x_1 + \frac{x_3}{6}) + 8.94(x_1 + \sqrt{|x_3^2 - x_2^2|}) \leq 0 \\
&g_2(X) = -x_4x_2^2(0.2x_1 + \frac{x_3}{12}) + 2.2(8.94(x_1 + \sqrt{|x_3^2 - x_2^2|}))^{4/3} \leq 0 \\
&g_3(X) = -x_4 + 0.0156x_1 + 0.15 \leq 0 \\
&g_4(X) = -x_4 + 0.0156x_3 + 0.15 \leq 0 \\
&g_5(X) = -x_3 + x_2 \leq 0 \\
&\text{where} \\
&0 \leq x_1, x_2, x_3 \leq 100, 1.05 \leq x_4 \leq 5
\end{aligned}
\tag{7.11}
$$

Tension/Compression Spring Design (TCSD): Tension/Compression Spring Design is described in[194] as an optimization problem, the objective is to minimize the weight of a tension/compression spring under the constraints of minimum deflection, shear stress, surge frequency, and outside diameter limitation. Eq. (7.12) shows the mathematical model of

this problem.

$$
\begin{aligned}
&\text{minimize} \\
&f(X) = (x_3 + 2)x_2x_1^2 \\
&\text{subject to} \\
&g_1(X) = 1 - \frac{x_2^3 x_3}{71785x_1^4} \leq 0 \\
&g_2(X) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \leq 0 \\
&g_3(X) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0 \\
&g_4(X) = \frac{x_1 + x_2}{1.5} - 1 \leq 0 \\
&\text{where} \\
&0.05 \leq x_1 \leq 2, 0.25 \leq x_2 \leq 1.3, 2 \leq x_3 \leq 15
\end{aligned}
\tag{7.12}
$$

Pressure Vessel Design (PVD): This problem is a classic engineering optimization problem that has been adopted to evaluate the performance of many meta-heuristic algorithms,[177,195,196] and the objective is to minimize the cost of the pressure vessel with the thickness of the shell (x_1), the thickness of the head (x_2), the inner radius (x_3), and the

length of the cylindrical section of the vessel (x_4), which can be described by Eq. (7.13).

$$
\begin{aligned}
&\text{minimize} \\
&f(X) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \\
&\text{subject to} \\
&g_1(X) = -x_1 + 0.0193x_3 \leq 0 \\
&g_2(X) = -x_2 + 0.00954x_3 \leq 0 \\
&g_3(X) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \leq 0 \\
&g_4(X) = x_4 - 240 \leq 0 \\
&\text{where} \\
&0 \leq x_1, x_2 \leq 99, 10 \leq x_3, x_4 \leq 200
\end{aligned}
\tag{7.13}
$$

More details and visualization of these engineering problems can be found in.[196]

Compared methods and parameters: We first evaluate the overview optimization performance of EMSMO. Six classic or advanced EAs including differential evolution (DE),[49] particle swarm optimization (PSO),[52] whale optimization algorithm (WOA),[195] sea lion optimization algorithm (SLO),[52] slime mold algorithm (SMA),[189] and arithmetic optimization algorithm (AOA)[197] are employed to the optimization comparison, and the detailed information is listed in Table 7.1.

The second experiment is to evaluate the efficiency of our proposed improvement-based probabilistic selection function in optimization sequence construction. Three conventional HHs frameworks are employed as competitor algorithms which are summarized in Table 7.2. The LLHs for these compared HHs frameworks are consistent with EMSMO.

All compared EAs and CEC2013 benchmark functions are provided by MEALPY[166] and OpFuNu.[173] For all compared algorithms, the population size is 100, the maximum fitness evaluations for CEC2013 benchmark functions and engineering optimization problems are $500*D$ and 20000 respectively, and the independent trial run for each optimization

Table 7.1: The compared EAs and parameters

Algorithm	Parameters	Value
EMSMO	Growth radius R	2
DE	Scale factor F	0.7
	Crossover rate Cr	0.9
	Mutation strategy	DE/rand/1/bin
PSO	Constant b	1
	Random number l	[-1, 1]
	Coefficient vector $\vec{A}$ and $\vec{C}$	[0, 2]
WOA	Parameter-free	
SLO	Constant b	1
	Random number l	[-1, 1]
	Coefficient vector $\vec{A}$ and $\vec{C}$	[0, 2]
SMA	Re-initialization rate z	0.03
AOA	Exploitation accuracy α	5
	Control parameter μ	0.5
	MOA_{min} and MOA_{max}	0.2 and 0.9

Table 7.2: The compared hyper-heuristic algorithms and descriptions

HHs alg.	Description
Simple random (SR) [198,199]	Select the subsequent LLHs with uniform probability
Random descent (RD) [200]	Initially select a LLHs randomly and repeat it as long as the improvement is achieved
Random permutation (RP) [201]	Randomly construct a sequence of heuristics

method is 30. The Holm multiple comparison test[182] is applied to determine the statistical significance between every pair of algorithms, and we use $+$, $\approx$, and $-$ to denote that our proposed EMSMO is significantly better, with no significance, and significantly worse with the compared method. Win-Tie-Loss information is summarized based on the statistical results, and the average rank of each optimization technique is also calculated. The best value is in bold.

7.3.2 Experimental results

Experimental results on CEC2013: This section shows the experimental and statistical results of between EMSMO and compared EAs on CEC2013 benchmark functions. Table 7.3, 7.4, and 7.5 summarize the optimization results among EMSMO and compared EAs, and convergence curves of representative functions are visualized in Figure 7.6. Table 7.6, 7.7, and 7.8 show the results between EMSMO and other HHs frameworks.

Experimental results on engineering optimization problems: Table 7.9 summarizes the optimization results of compared EAs on three engineering optimization problems. Given all the engineering problems contain constraints, the original EMSMO and compared EAs cannot deal with the constrained optimization problem. For the sake of simplicity, we equipped the EMSMO and compared EAs with a death penalty function,[165] which allocates an extremely bad fitness value to infeasible solutions. And if more than 1 of 30 trial runs cannot find a feasible solution at the end of optimization, we manually fill the mean, std, and the worst with NaN in Table 7.9.

7.4 Discussion

In this section, we first provide the computational complexity analysis of EMSMO, and then the performance of EMSMO based on the experimental results is discussed. Then, we provide some open topics to further improve the performance of EMSMO at the end of this

Table 7.3: Experimental and statistical results between EMSMO and compared EAs on 10-D CEC2013 Suite

Func	DE		PSO		WOA		SLO		SMA		AOA		EMSMO	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
f_1	1.44e+03 +	5.91e+02	1.79e+03 +	2.71e+03	-1.14e+03 +	2.69e+02	-1.10e+03 +	3.33e+02	-7.20e+02 +	4.78e+02	6.78e+03 +	1.84e+03	**-1.40e+03**	**3.12e-01**
f_2	1.79e+07 +	6.55e+06	3.12e+07 +	3.14e+07	5.34e+06 +	3.56e+06	6.29e+06 +	5.59e+06	1.41e+07 +	7.70e+06	6.66e+07 +	3.16e+07	**2.45e+06**	**1.48e+06**
f_3	1.81e+09 +	5.48e+08	6.73e+09 +	3.22e+09	**7.44e+08** ≈	**1.01e+09**	8.10e+08 ≈	1.04e+09	3.90e+09 +	2.78e+09	1.59e+11 +	6.45e+11	7.79e+08	1.83e+09
f_4	4.07e+04 +	1.34e+04	6.78e+04 +	3.17e+04	3.56e+04 +	1.80e+04	4.00e+04 +	1.65e+04	4.24e+04 +	1.39e+04	8.34e+04 +	1.20e+05	**9.76e+03**	**3.74e+03**
f_5	-3.46e+02 +	1.48e+02	1.85e+03 +	3.49e+03	-6.12e+02 ≈	3.40e+02	-3.43e+02 +	5.19e+02	-4.80e+02 +	3.64e+02	8.75e+03 +	4.32e+03	**-7.52e+02**	**2.53e+02**
f_6	-7.46e+02 +	4.44e+01	-6.63e+02 +	1.90e+02	-8.06e+02 +	4.31e+01	-7.87e+02 +	5.60e+01	-8.01e+02 ≈	4.72e+01	-2.16e+02 +	2.40e+02	**-8.31e+02**	**2.61e+01**
f_7	6.75e+03 ≈	9.13e+02	9.00e+03 +	3.01e+03	8.25e+03 +	3.17e+03	8.69e+03 +	3.06e+03	**-6.85e+02** −	**3.82e+01**	2.60e+04 +	2.09e+04	5.84e+03	1.78e+03
f_8	**-6.79e+02** ≈	**1.12e-01**	-6.79e+02 ≈	1.11e-01	-6.79e+02 ≈	1.03e-01	-6.79e+02 ≈	8.92e-02	-6.79e+02 ≈	9.65e-02	-6.79e+02 ≈	1.55e-01	-6.79e+02	9.01e-02
f_9	-5.90e+02 +	8.89e-01	-5.90e+02 +	1.30e+00	-5.91e+02 +	1.43e+00	-5.91e+02 +	1.37e+00	-5.92e+02 +	1.58e+00	-5.88e+02 +	1.39e+00	**-5.93e+02**	**1.51e+00**
f_{10}	-1.29e+02 +	1.05e+02	1.07e+02 +	2.74e+02	-4.32e+02 +	4.89e+01	-4.17e+02 +	6.59e+01	-3.15e+02 +	1.23e+02	5.20e+02 +	3.03e+02	**-4.86e+02**	**1.37e+01**
f_{11}	-3.08e+02 +	1.16e+01	-2.91e+02 +	2.80e+01	-3.22e+02 +	2.85e+01	-3.20e+02 +	3.93e+01	-3.35e+02 +	1.93e+01	-2.19e+02 +	2.66e+01	**-3.52e+02**	**1.61e+01**
f_{12}	-1.92e+02 +	9.97e+00	-1.79e+02 +	3.29e+01	-2.12e+02 +	3.52e+01	-2.10e+02 +	3.70e+01	-2.32e+02 +	2.06e+01	-1.30e+02 +	2.39e+01	**-2.57e+02**	**1.10e+01**
f_{13}	-1.02e+02 +	9.76e+00	-1.01e+02 +	2.60e+01	-1.03e+02 +	2.82e+01	-1.01e+02 +	3.10e+01	-1.22e+02 ≈	2.63e+01	-4.05e+01 +	2.47e+01	**-1.38e+02**	**1.59e+01**
f_{14}	2.00e+03 +	1.90e+02	2.12e+03 +	2.49e+02	1.25e+03 ≈	4.23e+02	1.23e+03 ≈	2.90e+02	1.39e+03 +	2.74e+02	2.33e+03 +	3.00e+02	**1.03e+03**	**2.31e+02**
f_{15}	2.09e+03 +	2.00e+02	2.06e+03 +	2.44e+02	1.39e+03 +	3.46e+02	1.42e+03 +	3.37e+02	1.61e+03 +	2.49e+02	2.50e+03 +	2.03e+02	**9.68e+02**	**2.64e+02**
f_{16}	2.02e+02 ≈	3.28e-01	2.02e+02 ≈	4.78e-01	**2.01e+02** −	**4.07e-01**	2.01e+02 ≈	4.19e-01	2.02e+02 ≈	5.41e-01	2.03e+02 +	9.11e-01	2.02e+02	4.83e-01
f_{17}	5.20e+02 +	2.91e+01	4.17e+02 +	3.64e+01	4.09e+02 +	3.45e+01	4.07e+02 +	3.19e+01	3.88e+02 +	1.68e+01	4.62e+02 +	2.41e+01	**3.45e+02**	**8.48e+00**
f_{18}	6.14e+02 +	2.89e+01	6.10e+02 +	8.69e+01	5.75e+02 +	6.19e+01	5.77e+02 +	6.65e+01	**4.84e+02** ≈	**3.00e+01**	8.22e+02 +	7.47e+01	5.04e+02	4.25e+01
f_{19}	9.89e+02 +	4.81e+02	1.94e+03 +	4.77e+03	5.70e+02 +	1.02e+02	7.49e+02 +	7.89e+02	5.17e+02 +	1.64e+01	1.33e+04 +	2.05e+04	**5.07e+02**	**2.69e+00**
f_{20}	6.04e+02 ≈	1.39e-01	6.04e+02 ≈	2.94e-01	6.04e+02 ≈	2.91e-01	6.04e+02 ≈	2.80e-01	6.04e+02 ≈	3.65e-01	6.05e+02 +	2.41e-01	**6.04e+02**	**3.46e-01**
f_{21}	1.43e+03 +	6.45e+01	1.20e+03 +	1.33e+02	1.13e+03 +	7.82e+01	1.16e+03 +	9.63e+01	1.13e+03 +	5.68e+01	1.47e+03 +	8.99e+01	**1.10e+03**	**3.52e+01**
f_{22}	3.01e+03 +	2.31e+02	3.23e+03 +	2.55e+02	2.30e+03 ≈	3.51e+02	2.25e+03 ≈	4.38e+02	2.53e+03 +	3.15e+02	3.60e+03 +	1.78e+02	**2.13e+03**	**3.92e+02**
f_{23}	2.94e+03 +	2.04e+02	2.87e+03 +	3.08e+02	2.42e+03 ≈	3.69e+02	2.59e+03 ≈	2.75e+02	2.76e+03 +	3.28e+02	3.55e+03 +	2.77e+02	**2.37e+03**	**3.07e+02**
f_{24}	1.22e+03 ≈	2.38e+00	1.23e+03 ≈	1.05e+01	1.22e+03 +	2.59e+01	1.22e+03 ≈	2.06e+01	1.23e+03 ≈	3.44e+00	1.25e+03 +	1.32e+01	**1.22e+03**	**2.18e+01**
f_{25}	1.33e+03 ≈	1.41e+00	1.32e+03 ≈	1.34e+01	**1.32e+03** ≈	**2.61e+01**	1.32e+03 ≈	2.31e+01	1.32e+03 ≈	4.12e+00	1.35e+03 +	8.61e+00	1.32e+03	2.14e+01
f_{26}	1.40e+03 +	7.57e-01	1.40e+03 +	1.08e+01	1.40e+03 +	2.95e+01	1.41e+03 +	4.13e+01	1.41e+03 +	5.35e+01	1.44e+03 +	3.93e+01	**1.38e+03**	**2.82e+01**
f_{27}	1.89e+03 +	1.76e+01	1.89e+03 +	3.49e+01	1.90e+03 +	4.61e+01	1.89e+03 +	4.47e+01	1.91e+03 +	6.14e+01	2.01e+03 +	4.68e+01	**1.81e+03**	**4.00e+01**
f_{28}	2.27e+03 +	8.92e+01	2.08e+03 ≈	1.85e+02	1.98e+03 ≈	1.93e+02	**1.97e+03** ≈	**1.60e+02**	2.20e+03 +	1.27e+02	2.43e+03 +	1.39e+02	1.99e+03	1.73e+02
+/≈/−:	22/6/0		22/6/0		18/9/1		18/10/0		19/8/1		27/1/0			
Ave. rank	4.82		5.25		2.5		3.32		3.67		6.96		**1.46**	

Table 7.4: Experimental and statistical results between EMSMO and compared EAs on 30-D CEC2013 Suite

Func	DE		PSO		WOA		SLO		SMA		AOA		EMSMO	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
f_1	3.23e+04 +	4.47e+03	3.07e+04 +	1.07e+04	1.58e+04 +	5.45e+03	1.35e+04 +	6.06e+03	2.64e+04 +	6.73e+03	5.33e+04 +	3.58e+03	**-1.38e+03**	**4.34e+00**
f_2	3.91e+08 +	9.57e+07	3.91e+08 +	2.44e+08	3.31e+08 +	1.45e+08	2.11e+08 +	9.72e+07	3.88e+08 +	1.68e+08	1.18e+09 +	5.21e+08	**1.90e+07**	**8.24e+06**
f_3	1.30e+11 +	2.28e+10	6.98e+13 +	1.72e+14	7.62e+15 +	2.85e+16	6.58e+11 +	9.47e+11	3.09e+14 +	1.46e+15	4.62e+18 +	1.43e+19	**1.43e+10**	**1.55e+10**
f_4	1.73e+05 +	2.66e+04	1.98e+05 +	5.37e+04	1.28e+05 +	3.31e+04	1.06e+05 +	2.85e+04	1.45e+05 +	3.10e+04	1.26e+05 +	7.24e+04	**4.97e+04**	**8.14e+03**
f_5	1.20e+04 +	2.30e+03	5.11e+04 +	2.14e+04	3.25e+03 +	1.78e+03	9.18e+03 +	5.01e+03	5.92e+03 +	3.16e+03	9.04e+04 +	3.24e+04	**-6.61e+02**	**1.46e+02**
f_6	2.41e+03 +	6.79e+02	4.30e+03 +	3.26e+03	1.12e+03 +	8.02e+02	6.20e+02 +	7.39e+02	2.51e+03 +	1.39e+03	1.07e+04 +	1.84e+03	**-7.93e+02**	**3.40e+01**
f_7	2.45e+05 ≈	2.27e+04	2.84e+06 +	4.69e+06	4.14e+04 −	1.03e+05	2.45e+07 +	5.24e+07	**6.03e+03** −	**9.65e+03**	7.35e+08 +	8.43e+08	9.06e+05	1.93e+06
f_8	-6.79e+02 ≈	6.12e-02	-6.79e+02 ≈	5.69e-02	-6.79e+02 +	7.97e-02	-6.79e+02 ≈	5.94e-02	-6.79e+02 +	5.72e-02	-6.79e+02 +	7.52e-02	**-6.79e+02**	**5.50e-02**
f_9	-5.58e+02 +	1.19e+00	-5.62e+02 +	2.96e+00	-5.59e+02 +	2.60e+00	-5.61e+02 +	3.28e+00	-5.60e+02 +	2.45e+00	-5.55e+02 +	2.52e+00	**-5.66e+02**	**2.78e+00**
f_{10}	3.73e+03 +	7.49e+02	5.00e+03 +	1.76e+03	2.27e+03 +	6.35e+02	1.48e+03 +	6.52e+02	3.50e+03 +	9.16e+02	7.68e+03 +	1.17e+03	**-4.58e+02**	**1.80e+01**
f_{11}	2.13e+02 +	3.87e+01	3.74e+02 +	1.43e+02	2.53e+02 +	1.16e+02	3.02e+02 +	1.38e+02	2.11e+02 +	1.24e+02	4.76e+02 +	6.87e+01	**1.06e+02**	**6.37e+01**
f_{12}	3.68e+02 +	6.44e+01	4.44e+02 +	1.31e+02	3.68e+02 +	1.03e+02	3.78e+02 +	1.26e+02	2.80e+02 +	1.03e+02	5.44e+02 +	6.72e+01	**1.51e+02**	**8.39e+01**
f_{13}	4.22e+02 +	4.66e+01	5.02e+02 +	1.29e+02	5.06e+02 +	1.23e+02	5.51e+02 +	1.45e+02	4.16e+02 +	1.13e+02	6.21e+02 +	7.69e+01	**3.02e+02**	**5.89e+01**
f_{14}	6.63e+03 +	3.69e+02	8.22e+03 +	4.80e+02	7.52e+03 +	7.15e+02	6.27e+03 +	7.55e+02	7.43e+03 +	5.50e+02	9.40e+03 +	4.44e+02	**5.44e+03**	**8.22e+02**
f_{15}	8.33e+03 +	4.03e+02	8.65e+03 +	3.77e+02	7.83e+03 +	8.02e+02	6.53e+03 +	1.00e+03	7.67e+03 +	7.17e+02	9.20e+03 +	4.94e+02	**5.13e+03**	**6.74e+02**
f_{16}	2.03e+02 +	3.98e-01	2.03e+02 +	4.27e-01	2.04e+02 +	8.17e-01	2.04e+02 +	6.71e-01	2.04e+02 +	5.21e-01	2.05e+02 +	9.13e-01	**2.02e+02**	**5.42e-01**
f_{17}	2.16e+03 +	1.60e+02	1.50e+03 +	2.63e+02	1.13e+03 +	1.06e+02	1.23e+03 +	2.34e+02	1.11e+03 +	1.14e+02	1.36e+03 +	5.29e+01	**8.01e+02**	**7.34e+01**
f_{18}	2.03e+03 +	1.97e+02	2.09e+03 +	4.39e+02	1.19e+03 −	9.52e+01	2.27e+03 +	3.83e+02	**1.19e+03** −	**1.28e+02**	2.86e+03 +	1.60e+02	1.43e+03	1.95e+02
f_{19}	3.14e+05 +	1.33e+05	3.28e+05 +	3.88e+05	4.56e+04 +	4.07e+04	5.67e+04 +	7.87e+04	1.89e+05 +	1.59e+05	1.67e+06 +	9.57e+05	**5.42e+02**	**8.40e+00**
f_{20}	**6.14e+02** −	**2.67e-01**	6.15e+02 ≈	4.09e-01	6.15e+02 +	7.46e-02	6.15e+02 +	2.10e-01	6.15e+02 +	5.92e-02	6.15e+02 +	2.91e-08	6.15e+02	3.96e-01
f_{21}	4.60e+03 +	3.04e+02	3.62e+03 +	6.16e+02	2.78e+03 +	1.55e+02	2.65e+03 +	1.61e+02	3.20e+03 +	1.76e+02	3.49e+03 +	2.07e+02	**1.91e+03**	**6.43e+01**
f_{22}	7.72e+03 +	3.61e+02	1.00e+04 +	3.97e+02	8.85e+03 +	5.14e+02	8.02e+03 +	9.16e+02	9.02e+03 +	6.56e+02	1.06e+04 +	4.38e+02	**7.02e+03**	**8.54e+02**
f_{23}	9.39e+03 +	3.13e+02	9.49e+03 +	4.48e+02	9.25e+03 +	6.38e+02	8.23e+03 +	9.17e+02	9.06e+03 +	5.14e+02	1.05e+04 +	5.03e+02	**7.39e+03**	**9.79e+02**
f_{24}	1.31e+03 ≈	4.13e+00	1.31e+03 ≈	7.61e+00	1.36e+03 +	1.86e+01	1.32e+03 +	1.17e+01	1.33e+03 +	1.39e+01	1.42e+03 +	5.77e+01	**1.31e+03**	**1.06e+01**
f_{25}	**1.42e+03** −	**4.40e+00**	1.42e+03 −	8.56e+00	1.48e+03 +	2.24e+01	1.44e+03 ≈	1.38e+01	1.45e+03 +	1.39e+01	1.53e+03 +	1.65e+01	1.44e+03	1.61e+01
f_{26}	1.56e+03 ≈	5.22e+01	1.58e+03 ≈	4.65e+01	1.54e+03 ≈	9.27e+01	1.60e+03 +	3.73e+01	**1.53e+03** ≈	**8.73e+01**	1.68e+03 +	9.40e+01	1.58e+03	5.39e+01
f_{27}	2.75e+03 ≈	3.67e+01	2.71e+03 ≈	7.59e+01	2.80e+03 ≈	1.25e+02	2.86e+03 +	1.13e+02	**2.68e+03** ≈	**6.94e+01**	3.76e+03 +	3.25e+02	2.73e+03	1.20e+02
f_{28}	5.37e+03 +	2.77e+02	6.01e+03 +	6.14e+02	6.92e+03 +	7.44e+02	5.57e+03 +	5.58e+02	6.35e+03 +	6.90e+02	6.36e+03 +	4.51e+02	**4.86e+03**	**3.03e+02**
+/≈/−:	21/5/2		22/5/1		24/2/2		26/2/0		24/2/2		28/0/0			
Ave. rank	3.89		4.67		4.03		3.67		3.53		6.67		**1.5**	

Table 7.5: Experimental and statistical results between EMSMO and compared EAs on 50-D CEC2013 Suite

Func	DE		PSO		WOA		SLO		SMA		AOA		EMSMO	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
f_1	9.17e+04 +	7.36e+03	6.49e+04 +	1.64e+04	4.29e+04 +	7.50e+03	3.37e+04 +	9.52e+03	6.35e+04 +	7.66e+03	7.52e+04 +	3.34e+03	**-1.35e+03**	**1.05e+01**
f_2	1.30e+09 +	1.79e+08	1.57e+09 +	7.90e+08	5.33e+08 +	1.90e+08	3.83e+08 +	1.76e+08	1.00e+09 +	3.63e+08	3.12e+09 +	6.47e+08	**2.78e+07**	**6.21e+06**
f_3	5.79e+11 +	4.27e+11	7.58e+14 +	1.71e+15	5.76e+13 +	1.57e+14	1.19e+13 +	4.42e+13	8.17e+13 +	1.67e+14	1.48e+15 +	4.10e+15	**2.80e+09**	**5.11e+09**
f_4	2.91e+05 +	2.58e+04	3.02e+05 +	9.27e+04	1.58e+05 +	3.69e+04	1.16e+05 +	3.31e+04	2.20e+05 +	4.87e+04	1.72e+05 +	2.84e+05	**7.54e+04**	**1.10e+04**
f_5	6.59e+04 +	8.66e+03	8.69e+04 +	5.23e+04	6.48e+03 +	1.87e+03	1.58e+04 +	8.76e+03	1.11e+04 +	3.24e+03	6.19e+04 +	2.23e+04	**-8.51e+02**	**3.83e+01**
f_6	1.03e+04 +	1.85e+03	6.01e+03 +	2.54e+03	2.49e+03 +	7.47e+02	1.18e+03 +	6.94e+02	5.17e+03 +	1.36e+03	8.68e+03 +	1.37e+03	**-7.64e+02**	**5.09e+01**
f_7	1.00e+06 +	1.38e+05	2.24e+07 +	4.07e+07	2.65e+04 −	7.83e+04	6.35e+07 +	1.11e+08	**3.45e+03** −	**5.49e+03**	4.39e+07 +	4.44e+07	8.54e+05	7.13e+05
f_8	-6.79e+02 ≈	4.18e-02	**-6.79e+02** ≈	**5.32e-02**	-6.79e+02 +	3.34e-02	-6.79e+02 ≈	5.03e-02	-6.79e+02 +	5.68e-02	-6.79e+02 +	4.55e-02	-6.79e+02	3.57e-02
f_9	-5.24e+02 +	1.42e+00	-5.30e+02 ≈	5.24e+00	-5.25e+02 +	3.13e+00	-5.27e+02 +	3.01e+00	-5.26e+02 +	2.89e+00	-5.18e+02 +	2.42e+00	**-5.34e+02**	**4.02e+00**
f_{10}	1.05e+04 +	1.23e+03	9.22e+03 +	2.58e+03	4.72e+03 +	1.22e+03	3.59e+03 +	1.19e+03	7.95e+03 +	1.21e+03	1.20e+04 +	1.24e+03	**-4.18e+02**	**1.99e+01**
f_{11}	1.19e+03 +	1.04e+02	9.85e+02 +	1.94e+02	7.08e+02 +	1.09e+02	9.16e+02 +	2.17e+02	7.00e+02 +	1.02e+02	8.82e+02 +	7.51e+01	**3.51e+02**	**8.81e+01**
f_{12}	1.30e+03 +	1.13e+02	9.83e+02 +	2.51e+02	8.87e+02 +	1.07e+02	9.74e+02 +	1.63e+02	8.58e+02 +	1.06e+02	9.59e+02 +	7.66e+01	**4.78e+02**	**9.58e+01**
f_{13}	1.36e+03 +	1.18e+02	1.09e+03 +	1.90e+02	1.04e+03 +	1.39e+02	1.06e+03 +	1.84e+02	9.62e+02 +	1.14e+02	9.58e+02 +	6.30e+01	**5.78e+02**	**8.04e+01**
f_{14}	1.20e+04 +	4.13e+02	1.54e+04 +	7.50e+02	1.36e+04 +	9.29e+02	1.16e+04 +	1.14e+03	1.44e+04 +	6.99e+02	1.65e+04 +	4.71e+02	**1.03e+04**	**9.55e+02**
f_{15}	1.54e+04 +	2.88e+02	1.57e+04 +	6.13e+02	1.50e+04 +	7.47e+02	1.28e+04 +	1.26e+03	1.47e+04 +	7.66e+02	1.64e+04 +	7.32e+02	**1.05e+04**	**7.30e+02**
f_{16}	2.04e+02 +	3.87e-01	2.04e+02 +	3.18e-01	2.05e+02 +	8.17e-01	2.03e+02 +	4.87e-01	2.05e+02 +	6.30e-01	2.06e+02 +	9.20e-01	**2.03e+02**	**6.76e-01**
f_{17}	4.96e+03 +	3.22e+02	2.93e+03 +	5.86e+02	1.70e+03 +	1.21e+02	2.09e+03 +	3.51e+02	1.92e+03 +	1.47e+02	1.87e+03 +	5.74e+01	**1.22e+03**	**8.70e+01**
f_{18}	4.66e+03 +	2.55e+02	3.97e+03 +	7.00e+02	**1.78e+03** −	**1.11e+02**	3.75e+03 +	4.83e+02	2.02e+03 ≈	1.94e+02	3.94e+03 +	1.48e+02	1.98e+03	1.97e+02
f_{19}	4.63e+06 +	1.45e+06	7.48e+05 +	9.94e+05	1.71e+05 +	8.85e+04	5.52e+04 +	7.64e+04	7.64e+05 +	4.94e+05	5.63e+05 +	2.10e+05	**5.72e+02**	**1.13e+01**
f_{20}	**6.24e+02** ≈	**2.66e-01**	6.25e+02 ≈	3.35e-01	6.25e+02 +	2.81e-01	6.25e+02 ≈	2.76e-01	6.25e+02 +	1.68e-01	6.25e+02 +	4.65e-02	6.25e+02	3.98e-01
f_{21}	9.16e+03 +	7.93e+02	6.89e+03 +	1.23e+03	4.86e+03 +	1.24e+02	3.72e+03 +	3.62e+02	5.76e+03 +	3.45e+02	4.93e+03 +	2.48e+02	**1.17e+03**	**2.45e+00**
f_{22}	1.36e+04 ≈	4.27e+02	1.72e+04 +	6.45e+02	1.56e+04 +	7.06e+02	1.48e+04 +	1.35e+03	1.61e+04 +	7.21e+02	1.84e+04 +	5.62e+02	**1.35e+04**	**9.76e+02**
f_{23}	1.63e+04 +	4.24e+02	1.68e+04 +	4.54e+02	1.64e+04 +	8.43e+02	1.48e+04 +	1.08e+03	1.61e+04 +	6.22e+02	1.83e+04 +	4.56e+02	**1.33e+04**	**1.37e+03**
f_{24}	**1.41e+03** −	**6.88e+00**	1.41e+03 −	1.47e+01	1.51e+03 +	7.42e+01	1.45e+03 ≈	2.32e+01	1.47e+03 ≈	2.81e+01	2.12e+03 +	2.79e+02	1.45e+03	3.98e+01
f_{25}	**1.52e+03** −	**5.78e+00**	1.52e+03 −	1.43e+01	1.63e+03 +	3.70e+01	1.55e+03 −	2.47e+01	1.59e+03 ≈	2.42e+01	1.76e+03 +	5.27e+01	1.58e+03	3.33e+01
f_{26}	1.70e+03 +	4.67e+00	1.68e+03 ≈	1.03e+01	1.70e+03 +	1.11e+01	1.70e+03 +	8.55e+00	1.69e+03 ≈	8.73e+00	1.90e+03 +	2.33e+02	**1.68e+03**	**1.28e+01**
f_{27}	3.68e+03 ≈	5.76e+01	**3.64e+03** ≈	**1.23e+02**	3.93e+03 +	1.95e+02	3.93e+03 +	1.93e+02	3.72e+03 ≈	1.02e+02	5.77e+03 +	4.41e+02	3.71e+03	2.43e+02
f_{28}	1.06e+04 +	7.83e+02	1.25e+04 +	2.06e+03	1.08e+04 +	9.20e+02	1.18e+04 +	1.92e+03	1.02e+04 +	9.67e+02	1.13e+04 +	9.35e+02	**8.36e+03**	**6.11e+02**
+/≈/−:	22/4/2		21/5/2		26/0/2		24/3/1		22/5/1		28/0/0			
Ave. rank	4.67		4.82		3.89		3.42		3.92		5.82		**1.42**	

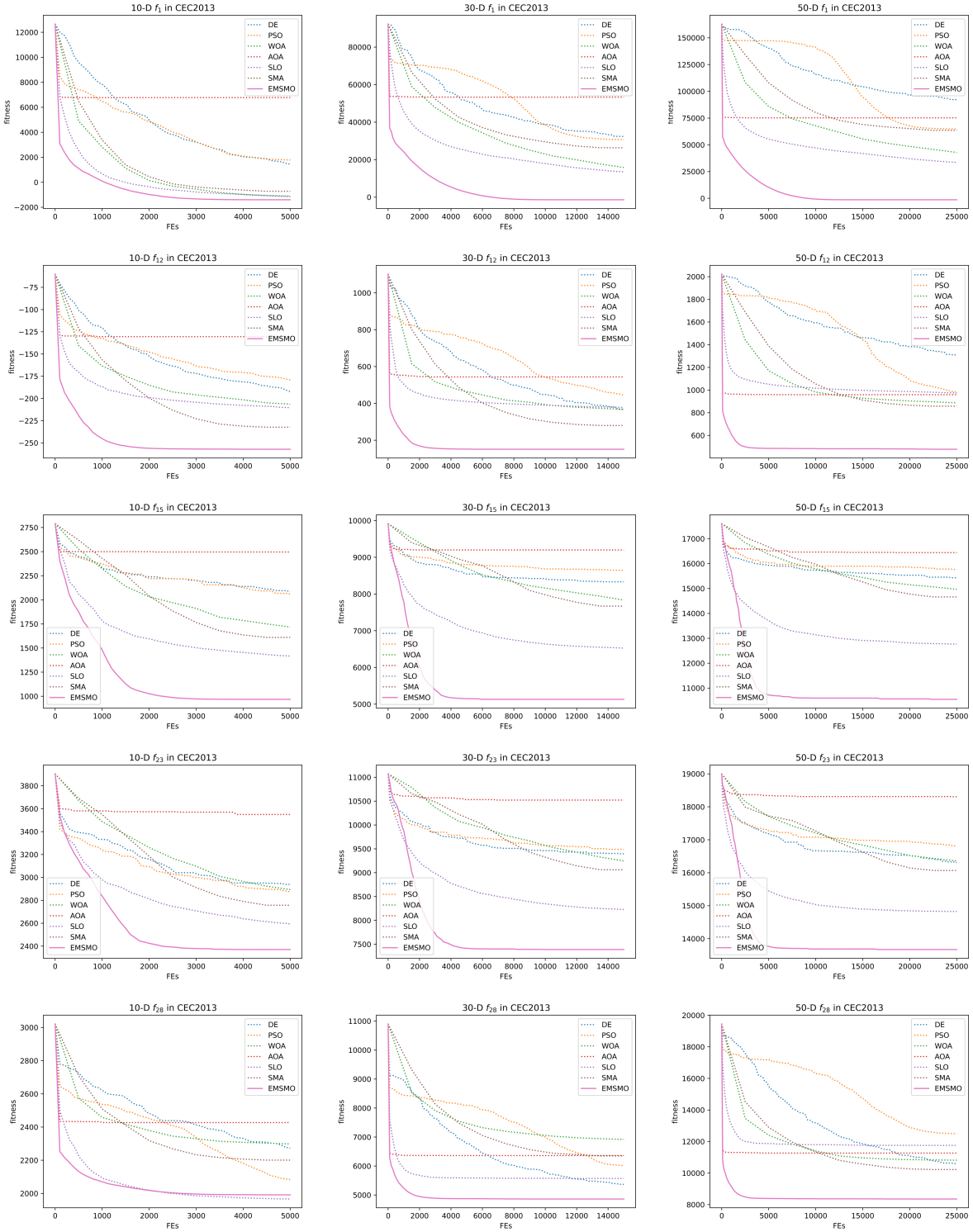


Figure 7.6: Convergence curves of representative functions (i.e., f_1: Unimodal; f_{12} and f_{15}: Multimodal; f_{23} and f_{28}: Composite).

Table 7.6: Experimental and statistical results between EMSMO and compared HHs algorithms on 10-D CEC2013 Suite

Func	SR		RD		RP		EMSMO	
	mean	std	mean	std	mean	std	mean	std
f_1	-9.08e+02 +	9.12e+02	-1.17e+03 +	6.76e+02	-9.31e+02 +	5.99e+02	**-1.40e+03**	**3.12e-01**
f_2	6.54e+06 +	3.69e+06	5.10e+06 +	3.77e+06	7.30e+06 +	3.65e+06	**2.45e+06**	**1.48e+06**
f_3	3.54e+09 +	3.00e+09	2.15e+09 +	3.17e+09	3.74e+09 +	3.31e+09	**7.79e+08**	**1.83e+09**
f_4	9.48e+03 ≈	4.61e+03	**7.41e+03** ≈	**4.80e+03**	1.08e+04 ≈	3.64e+03	9.76e+03	3.74e+03
f_5	-7.61e+01 +	7.56e+02	-6.31e+02 ≈	3.99e+02	-1.80e+02 +	6.04e+02	**-7.52e+02**	**2.53e+02**
f_6	-8.19e+02 ≈	4.76e+01	**-8.56e+02** −	**2.96e+01**	-8.01e+02 +	5.69e+01	-8.31e+02	2.61e+01
f_7	6.51e+03 ≈	2.11e+03	**5.40e+03** ≈	**2.74e+03**	5.82e+03 ≈	1.88e+03	5.84e+03	1.78e+03
f_8	-6.79e+02 ≈	9.80e-02	-6.79e+02 ≈	9.64e-02	**-6.79e+02** ≈	**1.08e-01**	-6.79e+02	9.01e-02
f_9	-5.93e+02 ≈	1.43e+00	-5.93e+02 ≈	1.86e+00	-5.93e+02 ≈	1.31e+00	**-5.93e+02**	**1.51e+00**
f_{10}	-4.03e+02 +	1.04e+02	-4.67e+02 ≈	5.89e+01	-3.95e+02 +	1.05e+02	**-4.86e+02**	**1.37e+01**
f_{11}	-3.50e+02 ≈	2.28e+01	-3.48e+02 ≈	2.26e+01	-3.47e+02 ≈	2.16e+01	**-3.52e+02**	**1.61e+01**
f_{12}	-2.52e+02 ≈	1.67e+01	-2.51e+02 ≈	1.88e+01	-2.50e+02 ≈	2.24e+01	**-2.57e+02**	**1.10e+01**
f_{13}	-1.37e+02 ≈	2.00e+01	**-1.44e+02** ≈	**2.29e+01**	-1.39e+02 ≈	2.00e+01	-1.38e+02	1.59e+01
f_{14}	1.05e+03 ≈	2.37e+02	1.12e+03 ≈	2.75e+02	1.14e+03 ≈	2.78e+02	**1.03e+03**	**2.31e+02**
f_{15}	1.08e+03 ≈	3.00e+02	1.04e+03 ≈	2.96e+02	**9.32e+02** ≈	**3.08e+02**	9.68e+02	2.64e+02
f_{16}	2.01e+02 ≈	5.17e-01	**2.01e+02** ≈	**5.75e-01**	2.02e+02 ≈	3.93e-01	2.02e+02	4.83e-01
f_{17}	3.62e+02 +	2.13e+01	3.64e+02 +	2.98e+01	3.62e+02 +	1.97e+01	**3.45e+02**	**8.48e+00**
f_{18}	5.47e+02 +	6.82e+01	5.29e+02 ≈	7.20e+01	5.51e+02 +	6.22e+01	**5.04e+02**	**4.25e+01**
f_{19}	5.24e+02 +	4.52e+01	5.43e+02 ≈	9.82e+01	5.18e+02 +	1.49e+01	**5.07e+02**	**2.69e+00**
f_{20}	6.04e+02 ≈	3.38e-01	6.04e+02 ≈	3.78e-01	**6.04e+02** ≈	**3.70e-01**	6.04e+02	3.46e-01
f_{21}	1.19e+03 +	9.93e+01	1.12e+03 +	5.87e+01	1.16e+03 +	1.12e+02	**1.10e+03**	**3.52e+01**
f_{22}	2.32e+03 ≈	3.68e+02	2.29e+03 ≈	3.20e+02	2.32e+03 ≈	3.28e+02	**2.13e+03**	**3.92e+02**
f_{23}	2.32e+03 ≈	3.35e+02	2.40e+03 ≈	3.15e+02	**2.30e+03** ≈	**3.38e+02**	2.37e+03	3.07e+02
f_{24}	1.21e+03 ≈	3.11e+01	1.21e+03 ≈	3.16e+01	**1.20e+03** ≈	**3.81e+01**	1.22e+03	2.18e+01
f_{25}	1.31e+03 ≈	3.38e+01	1.31e+03 ≈	3.40e+01	**1.31e+03** ≈	**3.57e+01**	1.32e+03	2.14e+01
f_{26}	1.39e+03 +	2.35e+01	1.40e+03 +	1.58e+01	1.39e+03 +	1.88e+01	**1.38e+03**	**2.82e+01**
f_{27}	1.84e+03 ≈	4.81e+01	1.83e+03 ≈	5.07e+01	1.84e+03 ≈	6.21e+01	**1.81e+03**	**4.00e+01**
f_{28}	2.02e+03 ≈	1.74e+02	2.01e+03 ≈	1.88e+02	**1.98e+03** ≈	**1.90e+02**	1.99e+03	1.73e+02
+/≈/−:	10/18/0		6/21/1		11/17/0			
Ave. rank	3.07		2.35		2.71		**1.85**	

section.

7.4.1 Computational complexity analysis of EMSMO

Similar to most EAs, EMSMO has three periods that need to be analyzed: initialization stage, iterative search stage, and parameter update stage. Supposing the population size is N, the dimension of the problem is D, the maximum iteration time is T, the number of the search strategy is a constant so that the complexity is $O(1)$, and the computational

Table 7.7: Experimental and statistical results between EMSMO and compared HHs algorithms on 30-D CEC2013 Suite

Func	SR		RD		RP		EMSMO	
	mean	std	mean	std	mean	std	mean	std
f_1	4.00e+03 +	3.23e+03	6.06e+03 +	9.53e+03	4.83e+03 +	3.76e+03	**-1.38e+03**	**4.34e+00**
f_2	5.75e+07 +	2.45e+07	6.96e+07 +	8.40e+07	6.08e+07 +	3.02e+07	**1.90e+07**	**8.24e+06**
f_3	1.96e+11 +	5.04e+11	4.75e+14 +	2.42e+15	8.34e+10 +	5.97e+10	**1.43e+10**	**1.55e+10**
f_4	**4.81e+04** ≈	**7.39e+03**	5.16e+04 ≈	8.27e+03	4.95e+04 ≈	8.53e+03	4.97e+04	8.14e+03
f_5	2.36e+03 +	1.52e+03	2.23e+03 +	4.45e+03	2.44e+03 +	2.08e+03	**-6.61e+02**	**1.46e+02**
f_6	-3.39e+02 +	2.85e+02	-2.03e+01 +	1.50e+03	-3.10e+02 +	3.82e+02	**-7.93e+02**	**3.40e+01**
f_7	1.74e+06 +	2.09e+06	9.15e+05 ≈	1.34e+06	2.46e+06 +	2.82e+06	**9.06e+05**	**1.93e+06**
f_8	-6.79e+02 ≈	5.91e-02	-6.79e+02 ≈	5.69e-02	-6.79e+02 ≈	5.48e-02	**-6.79e+02**	**5.50e-02**
f_9	-5.65e+02 ≈	3.25e+00	-5.66e+02 ≈	3.21e+00	-5.65e+02 ≈	2.89e+00	**-5.66e+02**	**2.78e+00**
f_{10}	4.67e+02 +	3.62e+02	1.86e+02 +	9.54e+02	5.28e+02 +	5.02e+02	**-4.58e+02**	**1.80e+01**
f_{11}	1.00e+02 ≈	8.85e+01	**8.13e+01** ≈	**1.23e+02**	8.78e+01 ≈	7.04e+01	1.06e+02	6.37e+01
f_{12}	2.00e+02 ≈	8.65e+01	1.88e+02 ≈	9.60e+01	1.95e+02 ≈	9.85e+01	**1.51e+02**	**8.39e+01**
f_{13}	3.35e+02 ≈	9.33e+01	3.13e+02 ≈	9.22e+01	3.32e+02 ≈	8.06e+01	**3.02e+02**	**5.89e+01**
f_{14}	5.74e+03 ≈	5.16e+02	**5.29e+03** ≈	**9.21e+02**	5.58e+03 ≈	5.69e+02	5.44e+03	8.22e+02
f_{15}	5.21e+03 ≈	6.09e+02	5.35e+03 ≈	9.92e+02	5.22e+03 ≈	6.63e+02	**5.13e+03**	**6.74e+02**
f_{16}	2.02e+02 ≈	5.96e-01	2.02e+02 ≈	6.19e-01	2.02e+02 ≈	5.01e-01	**2.02e+02**	**5.42e-01**
f_{17}	8.67e+02 ≈	1.13e+02	8.30e+02 ≈	1.15e+02	8.74e+02 ≈	1.16e+02	**8.01e+02**	**7.34e+01**
f_{18}	1.78e+03 +	3.38e+02	1.51e+03 ≈	2.65e+02	1.86e+03 +	2.94e+02	**1.43e+03**	**1.95e+02**
f_{19}	1.91e+03 +	2.89e+03	1.92e+04 +	4.66e+04	1.90e+03 +	2.13e+03	**5.42e+02**	**8.40e+00**
f_{20}	6.14e+02 ≈	1.28e-01	**6.14e+02** ≈	**1.81e-01**	6.14e+02 ≈	1.72e-01	6.15e+02	3.96e-01
f_{21}	2.15e+03 +	1.12e+02	2.30e+03 +	4.18e+02	2.17e+03 +	1.14e+02	**1.91e+03**	**6.43e+01**
f_{22}	7.32e+03 ≈	7.28e+02	7.19e+03 ≈	1.01e+03	7.51e+03 ≈	1.03e+03	**7.02e+03**	**8.54e+02**
f_{23}	**7.24e+03** ≈	**8.16e+02**	7.36e+03 ≈	9.56e+02	7.48e+03 ≈	9.09e+02	7.39e+03	9.79e+02
f_{24}	1.32e+03 ≈	1.59e+01	1.32e+03 ≈	2.67e+01	1.32e+03 +	1.71e+01	**1.31e+03**	**1.06e+01**
f_{25}	1.45e+03 +	1.74e+01	1.46e+03 +	2.67e+01	1.45e+03 +	1.72e+01	**1.44e+03**	**1.61e+01**
f_{26}	1.53e+03 ≈	8.74e+01	1.57e+03 ≈	7.40e+01	**1.51e+03** −	**8.74e+01**	1.58e+03	5.39e+01
f_{27}	**2.71e+03** ≈	**2.83e+02**	2.75e+03 ≈	2.77e+02	2.86e+03 +	1.75e+02	2.73e+03	1.20e+02
f_{28}	4.84e+03 ≈	4.49e+02	**4.77e+03** ≈	**4.41e+02**	4.92e+03 ≈	3.45e+02	4.86e+03	3.03e+02
+/≈/−:	11/17/0		9/19/0		13/14/1			
Ave. rank	2.60		2.60		3.17		**1.60**	

complexity of the initialization stage can be simply calculated to $O(ND)$. For one generation of the search, the complexity of each strategy is $O(D)$, and N individuals need to be updated, thus the computational complexity of one generation of the search is $O(ND)$. In the parameter update stage, the score matrix cooperates with the roulette wheel strategy to identify the search strategy for each individual, where the computational complexity is $O(N)$. After the evaluation, the necessary computation for updating the score matrix is

Table 7.8: Experimental and statistical results between EMSMO and compared HHs algorithms on 50-D CEC2013 Suite

Func	SR		RD		RP		EMSMO	
	mean	std	mean	std	mean	std	mean	std
f_1	1.68e+03 +	2.05e+03	1.74e+04 +	2.01e+04	1.02e+03 +	1.83e+03	**-1.35e+03**	**1.05e+01**
f_2	6.84e+07 +	2.38e+07	2.04e+08 +	1.58e+08	7.09e+07 +	3.38e+07	**2.78e+07**	**6.21e+06**
f_3	5.54e+10 +	2.14e+10	9.30e+11 +	3.77e+12	4.94e+10 +	2.18e+10	**2.80e+09**	**5.11e+09**
f_4	**6.10e+04** −	**1.06e+04**	7.35e+04 ≈	1.22e+04	6.14e+04 −	9.44e+03	7.54e+04	1.10e+04
f_5	6.68e+02 +	5.91e+02	3.75e+03 +	5.80e+03	5.92e+02 +	5.35e+02	**-8.51e+02**	**3.83e+01**
f_6	-5.69e+02 +	5.43e+01	1.59e+02 +	1.18e+03	-5.80e+02 +	6.93e+01	**-7.64e+02**	**5.09e+01**
f_7	9.95e+05 +	4.03e+05	8.62e+05 ≈	4.32e+05	9.68e+05 ≈	4.39e+05	**8.54e+05**	**7.13e+05**
f_8	-6.79e+02 ≈	3.82e-02	**-6.79e+02** ≈	**4.61e-02**	-6.79e+02 ≈	2.51e-02	-6.79e+02	3.57e-02
f_9	-5.34e+02 ≈	3.69e+00	**-5.34e+02** ≈	**4.68e+00**	-5.33e+02 ≈	3.30e+00	-5.34e+02	4.02e+00
f_{10}	1.69e+02 +	1.98e+02	1.45e+03 +	2.32e+03	1.70e+02 +	2.44e+02	**-4.18e+02**	**1.99e+01**
f_{11}	3.72e+02 ≈	1.16e+02	3.81e+02 ≈	1.27e+02	3.65e+02 ≈	9.22e+01	**3.51e+02**	**8.81e+01**
f_{12}	5.21e+02 ≈	9.35e+01	5.44e+02 ≈	1.17e+02	4.96e+02 ≈	1.11e+02	**4.78e+02**	**9.58e+01**
f_{13}	6.04e+02 ≈	9.30e+01	6.11e+02 ≈	1.11e+02	6.00e+02 ≈	7.70e+01	**5.78e+02**	**8.04e+01**
f_{14}	1.03e+04 ≈	9.97e+02	**1.02e+04** ≈	**1.58e+03**	1.08e+04 ≈	8.47e+02	1.03e+04	9.55e+02
f_{15}	1.10e+04 ≈	8.63e+02	**1.03e+04** ≈	**1.26e+03**	1.14e+04 +	9.65e+02	1.05e+04	7.30e+02
f_{16}	2.03e+02 ≈	4.92e-01	2.03e+02 ≈	5.30e-01	2.03e+02 +	4.98e-01	**2.03e+02**	**6.76e-01**
f_{17}	1.32e+03 +	1.36e+02	1.35e+03 +	1.90e+02	1.30e+03 +	1.31e+02	**1.22e+03**	**8.70e+01**
f_{18}	2.60e+03 +	3.58e+02	2.58e+03 +	3.46e+02	2.59e+03 +	3.45e+02	**1.98e+03**	**1.97e+02**
f_{19}	6.20e+02 +	5.41e+01	7.68e+03 +	2.83e+04	6.14e+02 +	3.82e+01	**5.72e+02**	**1.13e+01**
f_{20}	6.24e+02 ≈	2.08e-01	6.24e+02 ≈	3.09e-01	**6.24e+02** ≈	**3.84e-01**	6.24e+02	3.98e-01
f_{21}	1.70e+03 +	2.33e+02	2.38e+03 +	1.05e+03	1.79e+03 +	3.51e+02	**1.17e+03**	**2.45e+00**
f_{22}	1.39e+04 ≈	1.14e+03	**1.32e+04** ≈	**1.47e+03**	1.35e+04 ≈	1.15e+03	1.35e+04	9.76e+02
f_{23}	1.36e+04 ≈	1.26e+03	**1.33e+04** ≈	**1.26e+03**	1.38e+04 ≈	1.14e+03	1.33e+04	1.37e+03
f_{24}	1.47e+03 ≈	5.20e+01	1.48e+03 ≈	9.66e+01	1.46e+03 ≈	3.50e+01	**1.45e+03**	**3.98e+01**
f_{25}	1.61e+03 ≈	3.15e+01	**1.57e+03** ≈	**5.56e+01**	1.61e+03 +	3.35e+01	1.58e+03	3.33e+01
f_{26}	1.68e+03 ≈	1.43e+01	**1.67e+03** ≈	**5.16e+01**	1.68e+03 ≈	5.24e+01	1.68e+03	1.28e+01
f_{27}	3.83e+03 ≈	1.86e+02	3.75e+03 ≈	3.12e+02	3.85e+03 ≈	1.80e+02	**3.71e+03**	**2.43e+02**
f_{28}	8.37e+03 ≈	5.39e+02	8.33e+03 ≈	6.26e+02	**8.23e+03** ≈	**6.58e+02**	8.36e+03	6.11e+02
+/≈/−:	11/16/1		10/18/0		13/14/1			
Ave. rank	2.92		2.67		2.67		**1.71**	

$O(N)$ as well. In summary, the time complexity of EMSMO is

$$O(ND) + O(T(N + ND + N)) := O(ND) + O(TND) := O(TND) \tag{7.14}$$

7.4.2 Performance analysis on CEC2013 benchmark functions

In the realm of optimization algorithm design, the evaluation of an algorithm's exploitation ability, exploration ability, and the balance struck between these two aspects holds

Table 7.9: Experimental results on engineering optimization problems

Problem		DE	PSO	WOA	SLO	SMA	AOA	EMSMO
CBD	Mean	**6.8492e+00**	7.8896e+00	7.5919e+00	7.4157e+00	8.2067e+00	8.0645e+00	7.0027e+00
	Std	**1.6473e-03**	5.6763e-01	6.7792e-01	5.2653e-01	1.4708e+00	1.3656e+00	1.9991e-01
	Best	**6.8449e+00**	7.1050e+00	6.8884e+00	6.8511e+00	6.8462e+00	6.8923e+00	6.8529e+00
	Worst	**6.8523e+00**	9.2709e+00	9.5588e+00	9.0316e+00	1.1335e+01	1.0411e+01	7.7317e+00
TCSD	Mean	2.7368e-02	1.6177e-02	NaN	NaN	1.3449e-02	2.2017e-02	**1.2790e-02**
	Std	2.3459e-02	3.4204e-03	NaN	NaN	9.8589e-04	8.5871e-03	**9.1033e-05**
	Best	1.2681e-02	1.2863e-02	**1.2666e-02**	1.2671e-02	1.2710e-02	1.3062e-02	1.2677e-02
	Worst	9.3420e-02	2.5602e-02	NaN	NaN	1.6954e-02	3.9296e-02	**1.3121e-02**
PVD	Mean	7.5978e+03	4.1833e+04	7.3184e+04	3.5210e+04	6.3324e+03	7.8339e+03	**5.9900e+03**
	Std	6.5189e+03	2.2271e+04	8.5217e+04	4.8743e+04	4.1312e+02	1.7404e+03	**3.2718e+01**
	Best	**5.8862e+03**	1.0233e+04	6.2511e+03	6.3547e+03	5.9091e+03	6.1582e+03	5.9102e+03
	Worst	4.2620e+04	9.5789e+04	3.4844e+05	2.1892e+05	7.3116e+03	1.5397e+04	**6.0640e+03**

paramount importance. In our analysis, we commence by delving into the exploitation prowess of EMSMO.

Given that the functions f_1 to f_5 within the CEC2013 benchmark are unimodal, the optimization performance across these functions can directly reflect the exploitation capacity of the algorithm. From the comprehensive numerical experiments and statistical analyses, EMSMO outperforms the six compared optimization techniques on most functions. This notable superiority is maintained across the majority of functions, with the exception of 10-D f_3. Moreover, this competitiveness is amplified as the dimension of problems increases, which shows the excellent exploitation capacity of EMSMO even while facing the challenges of higher-dimensional problems.

f_6 to f_{28} are multimodal and composition functions in CEC2013 suite. These functions have multiple optima and complex fitness landscapes, which serve as challenging testbeds for evaluating the performance of algorithms.

Through the experimental and statistical results, our proposed EMSMO remains exceptionally competitive in comparison to the other six optimization techniques. This conclusion is verified through both statistical analyses and the average rank of performance.

However, it is important to note that there are instances of slight deterioration in EMSMO's performance, as observed in functions f_{24} and f_{25}.

To explain this phenomenon reasonably, the No Free Lunch Theorem[109] is responsible for this degeneration. This theory states that the average performance of any two stochastic optimization algorithms is equivalent across all possible problems. Consequently, if an algorithm outperforms in a specific class of problems, it must perform poorly in the rest of the problems in order to maintain the same average performance. In light of this theory, we can infer that EMSMO may exhibit reduced performance in problems that possess similar structural characteristics to f_{24} and f_{25}.

f_6 to f_{28} are multimodal and composition functions, and these benchmark functions contain multiple optima and complex fitness landscape characteristics so that they are allowed to evaluate the overall performance of algorithms. In general, our proposed EMSMO is quite competitive with the other six optimization techniques based on statistical analysis and the average rank. However, some deterioration of EMSMO can be observed such as in f_{24} and f_{25}, and we may use the No Free Lunch Theorem[109] to explain this deterioration: No Free Lunch Theorem states that any pair of algorithms have identical averaging performance in all possible problems, and if an algorithm performs better in a certain class of problems, it must degenerate in the other class of problems since it is the only way to achieve the same averaging performance. Therefore, we infer that our proposed EMSMO may not be good at dealing with the problem which has a similar structure to f_{24} and f_{25}.

7.4.3 Performance analysis on engineering optimization problems

In this study, we utilize three classical engineering optimization problems to comprehensively evaluate EMSMO: corrugated bulkhead design, tension/compression spring design, and pressure vessel design. The experimental results are summarized in Table 7.9.

An additional aspect we are concerned about is the stability of the algorithm, which is an important metric when addressing real-world optimization problems. The stability can

be reflected through metrics like standard deviation and the worst results, as presented in Table 7.9. Especially noteworthy are the results of the tension/compression spring design problem. In this scenario, the optimizers of WOA and SLO fail to find a feasible solution at least once. On the contrary, our proposed EMSMO consistently finds a feasible solution in all trial runs. This outcome primarily proves the stability of EMSMO in real-world applications.

7.4.4 Performance analysis with various HHs

To rigorously evaluate the efficiency of our novel improvement-based probabilistic selection function, we employ three baseline approaches: simple random, random descent, and random permutation. These baseline methods compare with EMSMO in comprehensive experiments. The experimental and statistical results practically prove the efficiency of our proposed selection function.

The results demonstrate that the improvement-based probabilistic selection function generally performs comparably or better than the three compared hyper-heuristics (HHs) frameworks in the majority of cases. In fact, it is significantly superior in nearly half of the instances. These results validate the efficacy of our proposal.

Nevertheless, we observe instances where EMSMO performs noticeably worse compared to the competing HHs on specific problems, such as 10-D f_6, 30-D f_{26}, and 50-D f_4. While the exact inner mechanism of the deterioration is not evident, we infer that these problems might present deceptive information for the score matrix update, which may mislead the construction of the optimization sequence.

7.5 Conclusion

This chapter proposes a novel hyper-heuristic algorithm named evolutionary multi-mode slime mold optimization (EMSMO). Inspired by the slime mold foraging behaviors, we

design four search schemes as the low-level heuristics. An improvement-based probabilistic selection function is designed as the High-Level component of EMSMO. To evaluate the performance of EMSMO, we implement comprehensive comparative experiments with six classic or advanced EAs and three HHs algorithms. Experimental and statistical results prove the competitiveness of EMSMO in practice.

Chapter 8

Surrogate ensemble assisted hyper-heuristic algorithm

This chapter presents an evolutionary multi-mode slime mold optimization[1]. Section 8.1 presents the motivation of this research, Section 8.2 details our proposal: SEA-HHA, Section 8.3 presents the experiments on the CEC benchmark functions, and Section 8.4 discusses the performance of EMSMO. Finally, Section 8.5 concludes this chapter.

8.1 Motivation

Evolutionary computation (EC) and swarm intelligence (SI) have achieved tremendous success in the research field[203–205] and industrial design[206–208] thanks to their superior characteristics such as robustness, flexibility, efficiency, and applicability. However, as a kind of stochastic optimization technique, an enormous number of fitness evaluation times (FEs) are necessary to find acceptable solutions, which severely limits the scalability of these methodologies in computationally expensive optimization problems (EOPs). Therefore,

[1]This chapter was previously published as This chapter was previously published as Zhong, R., Yu, J., Zhang, C., & Munetomo, M.: Surrogate Ensemble-Assisted Hyper-Heuristic Algorithm for Expensive Optimization Problems. Int J Comput Intell Syst 16, 169 (2023). https://doi.org/10.1007/s44196-023-00346-y.[202]

many researchers attempt to combine these algorithms with mathematical methods to further accelerate the convergence of optimization. A common approach is to adopt the surrogate model and the infill sampling criterion to predict potential solutions, and then combine them with conventional evolutionary algorithms (EA) to improve search efficiency. An optimization framework like this that combines the two belongs to the surrogate-assisted evolutionary algorithm (SAEA).[28,145,209]

Up to now, many effective SAEAs have been reported: Dong et al.[210] extended the surrogate-assisted approach to the popular and powerful grey wolf optimization (SAGWO). The radial basis function (RBF) is employed to assist the meta-heuristic exploration and fitness landscape knowledge mining. Nishihara et al.[211] noticed that not only the model settings but also the training data chosen will influence the estimation performance and designed an adaption scheme for training data selection, which includes four criteria: *All Data*, *Current Population*, *Recent Data*, and *Neighbor*. These schemes collaborate with differential evolution (DE) for computationally expensive optimization problems. Wang et al.[212] proposed a global and local surrogate-assisted DE (GL-SADE) to solve high-dimensional EOPs. The global RBF model is trained with all samples to approximate the tendency of the whole fitness landscape, and the local Kriging model trained with the local population prefers to select solutions with well-performed prediction and great uncertainty, which can prevent the search direction from getting trapped into local optima. And the unique reward search strategy in GL-SADE encourages the re-utilization of the Kriging model when the solution found by the local Kriging model is the best so far. Cai et al.[213] introduced the surrogate-assisted technique to multi-objective EOPs. Two strategies are proposed to balance the global and local search for multi-objective optimization: (1) Maximum angle-distance sequential sampling based improved surrogate-based multi-objective local search method. (2) Diversity-enhanced expected improvement matrix infill criterion-based pre-screening strategy.

The introduction of surrogate-assisted techniques has promoted the development of the

EC community rapidly and endowed the optimizers with a stronger ability to tackle more complex optimization problems at the cost of computational resources. However, No Free Lunch Theorem [109] states that any pair of black-box optimization algorithms have identical averaging performance on all possible problems, and if an algorithm performs well on a specific category of problems, then, it must degenerate on the remaining problems, which is the only way for all algorithms to have the same performance on average across all functions. Thus, many researchers try to develop a generic optimization framework, which can dynamically modify the structure of the algorithm to adapt to the characteristics of certain problems.

The hyper-heuristics framework provides a potential opportunity to realize that. From the perspective of the hyper-heuristic algorithm (HHA), an optimization algorithm can be regarded as a combination of search strategies (e.g. the genetic algorithm (GA) repeats the crossover and mutation operators and particle swarm optimization (PSO) iterates the velocity and location update.), and this sequence of heuristics can also be optimized. As an "off-the-peg" technique rather than "made-to-measure" meta-heuristics, [214] HHA is a high-level automatic methodology that manipulates a set of low-level heuristics (LLHs) to search for acceptable solutions. [198] Meanwhile, many HHAs have been reported to deal with combinatorial optimization problems [199–201,215] whilst only a few have dealt with continuous problems. [216]

In this chapter, we regard the surrogate-assisted technique as a kind of search strategy and propose a novel surrogate ensemble-assisted hyper-heuristic algorithm (SEA-HHA) for continuous and computational EOPs. In the low-level component of SEA-HHA, we design four search strategy archives as the low-level heuristics (LLHs): The exploration strategy archive, exploitation strategy archive, surrogate-assisted estimation archive, and mutation strategy archive, each archive contains several search strategies. In the high-level component design, we apply a probabilistic random selection function to construct the optimization sequence dynamically.

8.2 Proposal: SEA-HHA

The overall optimization framework of the proposed SEA-HHA can be summarized as Figure 8.1. Specifically, in the low-level component of SEA-HHA, we design four generation archives containing the various LLHs: Exploration strategy archive, exploitation strategy archive, surrogate-assisted estimation archive, and mutation strategy archive. Besides, each archive has one or more search strategies, and different strategies in the same archive have an unbiased probability of being chosen. In the high-level component of SEA-HHA, a stochastic selection function based on pre-defined probabilities is employed as the decision function to determine the optimization sequence dynamically.

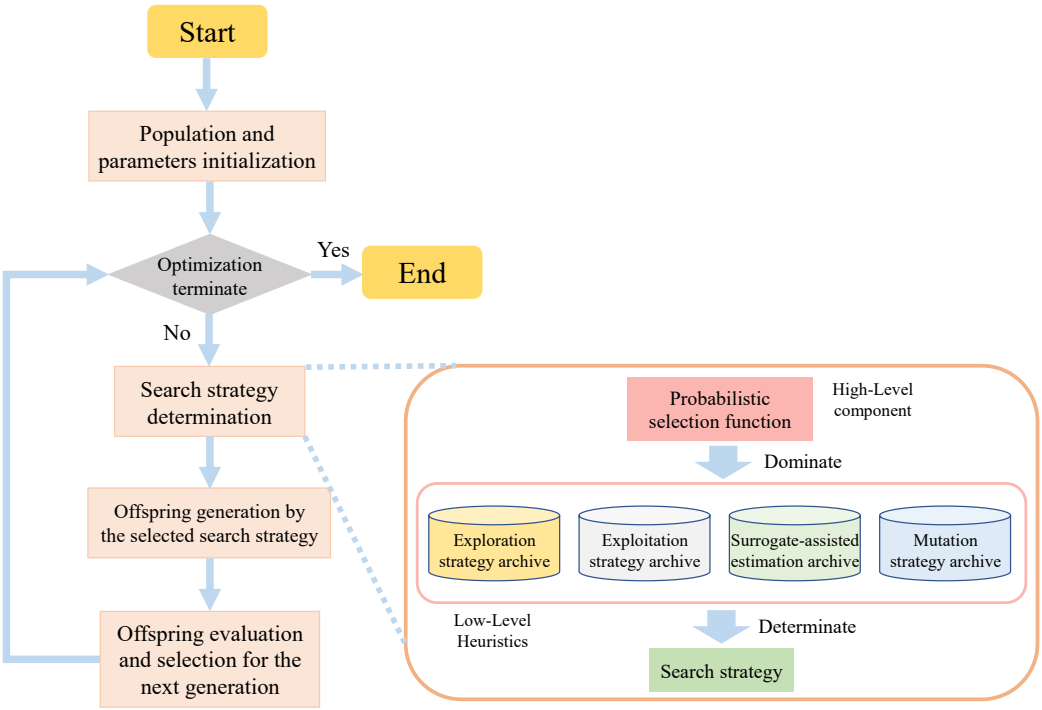


Figure 8.1: The main architecture of SEA-HHA, the flowchart is similar to most EAs, and our proposal focuses on the search strategy determination part.

8.2.1 Exploration strategy archive

The differential-based search strategy was first proposed in DE[49] and has been adopted in many bio-inspired EAs to describe the foraging behaviors of natural organisms.[179,188,217,218]

In this chapter, we also use the basic form of the differential-based search strategy in the exploration strategy archive and provide three different ways to select the base individual.

$$X_{i+1} = X_{base} + F \cdot (X_{r2} - X_{r3}) \tag{8.1}$$

where X_{base} is randomly selected from $\{X_i, X_{best}, X_{r1}\}$ with equal probability. X_i is the i^{th} individual, X_{best} represents the best solution in the current population, and X_{r1}, X_{r2} and X_{r3} are mutually different solutions which randomly sampled from the current population. F is a scaling vector and each element is randomly sampled from $[-0.8, 0.8]$.

The pseudocode of the exploration operation is shown in Figure 8.2.

Algorithm Exploration operation

Require: Dimension: D, Population: P, Current index: i
Ensure: Offspring: O
1: $r \leftarrow$ **rand**()
2: $r1, r2, r3 \leftarrow$ **rands**() # $r1, r2$ and $r3$ are mutually different
3: **if** $r < 1/3$ **then**
4: $\quad X_{base} \leftarrow X_i$
5: **else if** $r < 2/3$ **then**
6: $\quad X_{base} \leftarrow P[\arg\min(F)]$ # select the current best solution
7: **else**
8: $\quad X_{base} \leftarrow X_{r1}$
9: **end if**
10: Construct a D-dimensional scaling vector F
11: $O \leftarrow X_{base} + F \cdot (X_{r2} - X_{r3})$
12: Ensure the O is within the search space
13: **return** O

Figure 8.2: The pseudocode of the exploration operation.

8.2.2 Exploitation strategy archive

Rather than using the complex mechanism and parameters to realize the exploitation operation, we only adopt two parameters to determine the exploitation search strategy: The

search direction D and the exploitation radius R. And these operators can be described in Eq. (8.2)

$$X_{i+1} = X_{base} + D \cdot R \tag{8.2}$$

X_{base} is randomly selected from $\{X_i, X_{best}, X_{r1}\}$ as well. D is a random vector and R is a constant. Once these two parameters are specified, the location of X_{i+1} can be identified. In our experiment settings, each element in D is uniformly sampled from $[-1, 1]$ and $R = 2$.

The pseudocode of the exploitation operation is shown in Figure 8.3.

Algorithm Exploitation operation

Require: Dimension:D, Population:P, Fitness:F, Current index:i, Radius:R
Ensure: Offspring: O

$r \leftarrow$ **rand**()
$r1, r2, r3 \leftarrow$ **rands**() # $r1, r2$ and $r3$ are mutually different
if $r < 1/3$ **then**
    $X_{base} \leftarrow X_i$
else if $r < 2/3$ **then**
    $X_{base} \leftarrow P[\arg\min(F)]$
else
    $X_{base} \leftarrow X_{r1}$
end if
for $i = 0$ to D **do**
    $O^i \leftarrow X^i_{base} + R \cdot$ **rand**$(-1, 1)$
end for
Ensure the O is within the search space
return O

Figure 8.3: The pseudocode of the exploitation operation.

8.2.3 Surrogate-assisted estimation archive

We regard surrogate-assisted estimation as a kind of search strategy to generate high-quality solutions, and the basic process is as follows. We first choose the dataset selection fashion among $All\ Data, Recent\ Data$, and $Neighbor$ randomly, then, the dataset is randomly divided into the training dataset and the validation dataset with the proportion of 80% and 20% respectively. Three kinds of models of polynomial regression (PR), support vector

regression (SVR), and Gaussian process regression (GPR), are employed to construct the approximation model, and we use an extra DE to estimate the best solution in the surrogate model which has the highest accuracy on the validation dataset. The real objective function will evaluate this estimated solution and participate in the optimization as an offspring individual. Next, we will introduce the selection training dataset and the surrogate model selection principles in detail.

Inspired by the SADE-ATDSC,[211] three different strategies for selecting training datasets are applied in the archive: $All\ Data, Recent\ Data$, and $Neighbor$. $All\ Data$ utilizes all solutions from optimization beginning to approximate the overview of the fitness landscape. $Recent\ Data$ represents the recent k generated solutions which are selected as the dataset to describe the regularity of solution movements in the optimization. And $Neighbor$ denotes the nearest-k solutions of X_{best} determined by the Manhattan distance, which can depict the characteristics of the fitness landscape near the current best solution. A general demonstration of dataset selection is shown in Figure 8.4.

A subsequent problem is which model can approximate the fitness landscape better with these selected solutions. As we mentioned before, the selected dataset is randomly separated into two parts: The training dataset with the 80% proportion of the original data and the validation dataset with the rest 20% proportion. Then, three kinds of models are constructed based on the training dataset, and the model that has the lowest mean squared error (MSE) loss on the validation dataset is considered the most accurate in this regression task. The calculation of MSE is in Eq. (8.3).

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(E(y|x_i) - y_i)^2 \tag{8.3}$$

where n is the size of the dataset, $E(y|x_i)$ is the expectation of the model given the solution x_i, and y_i is the real fitness value of x_i, and the well-performed solution in the surrogate model is considered as a high-quality solution on the real fitness landscape and will be

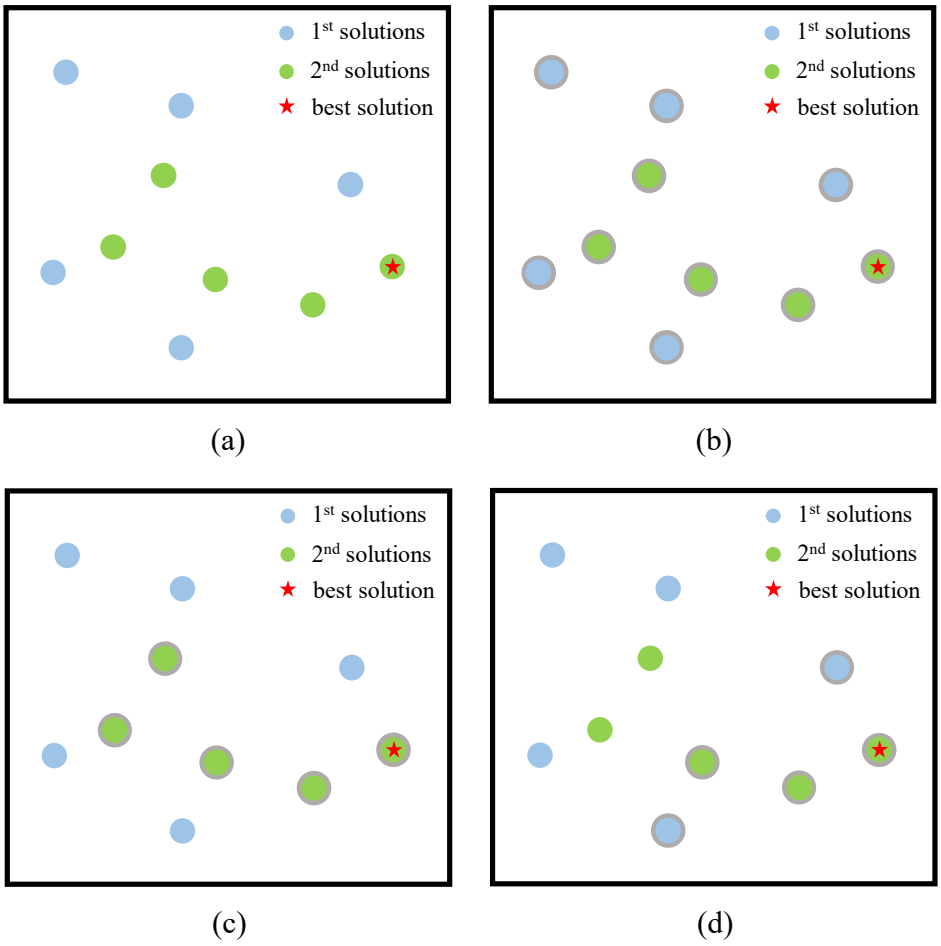


Figure 8.4: A demonstration of dataset selection criteria, blue points are solutions in the 1^{st} generation, green points are solutions in the 2^{nd} generation, the red star represents the current best solution, and the selected dataset scale k is five in this example. The point surrounding the grey circle represents the selected data for model construction. (a). The original distribution of solutions. (b). $All\ Data$ principle. (c). $Recent\ Data$ principle. (d). $Neighbor$ principle.

evaluated by the real objective function and participate in the optimization process.

The pseudocode of the surrogate-assisted estimation strategy is shown in Figure 8.5.

8.2.4 Mutation strategy archive

The Mutation strategy archive only contains one strategy that

$$X_{i+1} = X_{lb} + r \cdot (X_{ub} - X_{lb}) \tag{8.4}$$

Algorithm Surrogate-assisted estimation

Require: Data archive: DA, Model archive: MA
Ensure: Estimated solution: S

$r \leftarrow$ **rand**()
if $r < 1/3$ **then**
    $SD \leftarrow$ **All**(DA) # All Data
else if $r < 2/3$ **then**
    $SD \leftarrow$ **Recent**(DA) # Recent Data
else
    $SD \leftarrow$ **Neighbor**(DA) # Neighbor
end if
$TD, VD \leftarrow$ **split**(SD) # 80% Training data and 20% Validation data
Train the models in MA with TD
Validate the models in MA with VD
Estimate the best solution S on the most accurate model in terms of the DE
Ensure the S is within the search space
return S

Figure 8.5: The pseudocode of the surrogate-assisted estimation strategy.

r is a uniform random value from $[0, 1]$. X_{lb} and X_{ub} are the lower and upper bound of search space respectively. Simply, we randomly generate a new solution in the search space to endow an ability to SEA-HHA to get rid of the local optimum.

In summary, the pseudocode of SEA-HHA is shown in Figure 8.6.

Different from most EAs in which the search strategy is applied to the whole population, the object of the search strategy in our proposed SEA-HHA is applied to the individual. Each individual in the population has a high opportunity to generate offspring individual with various strategies, which is expected to enhance the diversity of the population and prevent premature convergence.

8.3 Numerical experiments

We implement a set of experiments to evaluate the performance of our proposed SEA-HHA. Section 8.3.1 introduces the experiment settings, and Section 8.3.2 shows the experimental results.

Algorithm SEA-HHA

Require: Dimension: D, Population size: PS, Maximum iteration: M, Predefined probability: PR, Search strategy archives: SSA,
Ensure: Global Optimum: GO

$P \leftarrow$ **LHS**(D, PS) # Latin hypercube sampling for population initialization
$GO \leftarrow$ **best**(P)
Data archive $DA \leftarrow P$
$g = 0$
while $g < M$ **do**
    for $i = 0$ to PS **do**
        Determine a certain search strategy from search strategy archives by pre-defined probability
        Construct the offspring O_i with sampled search strategy
        Save the O_i to DA
    end for
    Survive the best-PS solutions among population and offspring
    $GO \leftarrow$ **best**(P)
    $g = g + 1$
end while
return GO

Figure 8.6: The pseudocode of SEA-HHA.

8.3.1 Experiment settings

Experiment environment: The proposed SEA-HHA is programmed with Python 3.11 and implemented in Hokkaido Univerisity's high-performance intercloud supercomputer equipped with a CentOS operating system, Intel Xeon Gold 6148 CPU, and 384GB RAM.

Benchmark functions: We evaluate the performance of SEA-HHA on the 5-D, 10-D, and 30-D of 28 CEC2013 benchmark functions and three complex engineering problems including Cantilever Beam Design,[219] Tension/Compression Spring Design,[220] and Pressure Vessel Design.[177]

Cantilever Beam Design: This problem is a structural engineering optimization problem that is related to the weight optimization of a cantilever beam with a square cross-

section. Eq. (8.5) shows the mathematical model of this problem.

$$
\begin{aligned}
&\text{minimize} \\
&f(X) = 0.0624(x_1 + x_2 + x_3 + x_4 + x_5) \\
&\text{subject to} \\
&g(X) = \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \leq 0 \\
&\text{where} \\
&0.01 \leq x_i \leq 100, i = 1, 2, ..., 5
\end{aligned}
\tag{8.5}
$$

Tension/Compression Spring Design: The object of this problem is to minimize the weight of a tension/compression spring under the constraints of minimum deflection, shear stress, surge frequency, and outside diameter limitation. The formulation is presented in Eq. (8.6).

$$
\begin{aligned}
&\text{minimize} \\
&f(X) = (x_3 + 2)x_2x_1^2 \\
&\text{subject to} \\
&g_1(X) = 1 - \frac{x_2^3x_3}{71785x_1^4} \leq 0 \\
&g_2(X) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \leq 0 \\
&g_3(X) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0 \\
&g_4(X) = \frac{x_1 + x_2}{1.5} - 1 \leq 0 \\
&\text{where} \\
&0.05 \leq x_1 \leq 2 \\
&0.25 \leq x_2 \leq 1.3 \\
&2 \leq x_3 \leq 15
\end{aligned}
\tag{8.6}
$$

Pressure Vessel Design: This problem attempts to minimize the cost of the pressure

vessel including the cost of forming, material, and welding. And this optimization problem can be expressed in Eq. (8.7).

$$
\begin{aligned}
&\text{minimize} \\
&f(X) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \\
&\text{subject to} \\
&g_1(X) = -x_1 + 0.0193x_3 \leq 0 \\
&g_2(X) = -x_2 + 0.00954x_3 \leq 0 \\
&g_3(X) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \leq 0 \\
&g_4(X) = x_4 - 240 \leq 0 \\
&\text{where} \\
&0 \leq x_1 \leq 99 \\
&0 \leq x_2 \leq 99 \\
&10 \leq x_3 \leq 200 \\
&10 \leq x_4 \leq 200
\end{aligned}
\tag{8.7}
$$

More detailed explanations and visual demonstrations of these engineering optimization problems can be found in.[196]

Compared methods and parameters: We compare our proposal SEA-HHA with four EAs and two SAEAs that are listed in Table 8.1. The selected probability for each search strategy archive in SEA-HHA plays an important role in guiding the optimization sequence construction. However, the determination of these parameters is also a difficult task. In this research, we fix the exploration probability, exploitation probability, surrogate-assisted estimation probability, and mutation probability with 0.33, 0.33, 0.33, and 0.01 respectively, which are also corresponding to the intuition of optimization algorithm design.

For all compared algorithms, the population size is 100, the maximum FEs by the real

Table 8.1: The compared optimization techniques and their parameter configuration

Method	Parameters	Value
SEA-HHA	Radius R	2
	Population size for the local model k	100
	Exploration probability	0.33
	Exploitation probability	0.33
	Surrogate-assisted estimation probability	0.33
	Mutation probability	0.01
SFO [55]	Coefficients of power attack A and ε	4 and 0.001
DSIDE [221]	Mutation strategy	DE/rand/1/bin
ROA [190]	Condition parameter C	0.1
SCSO [66]	Sensitivity range r_G	[0, 2]
	Phases control range R	[-2r_G, 2r_G]
aRBF-NFO [63]	Surrogate model	RBF
	basic optimizer	NFO [222]
SHEALED [223]	Surrogate model	RBFNmv
	Scaling factor F	0.5
	Crossover rate Cr	0.8
	Probability of mutation Pm	0.3
	Probability of crossover Pc	0.8

objective function in both the CEC2013 suite and engineering optimization problems are 1000, the sample size of the random search for promising solutions in surrogate models is 1000, and the independent trial run for each method is 30.

8.3.2 Experimental results

This section shows the experimental and statistical results among seven compared optimization methods on CEC2013 benchmark functions and engineering optimization problems. Here, we apply the Friedman test and the Holm multiple comparison test[182] to check the statistical significance between every pair of algorithms. $+$, $\approx$, and $-$ are applied to represent that our proposed SEA-HHA is significantly better, with no significance, and significantly worse with the compared method, and the best fitness value is in bold.

Optimization on CEC2013 Suite: Table 8.2, 8.3, and 8.4 provide the experimental and statistical results on CEC2013 benchmark functions. The mean and standard deviation (std) are calculated at the end of the optimization within 30 trial runs.

Optimization on engineering optimization problems: The original SEA-HHA cannot solve the constrained optimization problems while the real-world engineering problems contain constraints. Therefore, we need to introduce a constraint-handling technique to SEA-HHA. Coello et al.[165] summarized the various penalty functions including static, dynamic, simulated annealing, adaptive, and death penalty. As one of the simplest methods, the death penalty assigns an enormous fitness value to the individual which violates the constraint in the minimization optimization. For the sake of simplicity, we equip the SEA-HHA and all compared algorithms with a death penalty function to deal with constrained optimization problems. Table 8.5 and 8.6 show the comparative results on the Cantilever Beam Design problem, Table 8.7 and 8.8 show the optimization results on the Tension/Compression Spring Design problem, and Table 8.9 and 8.10 show the results on the Pressure Vessel Design problem.

Table 8.2: The experimental and statistical results on 5-D CEC2013 Suite.

Func	SFO		DSIDE		ROA		SCSO		aRBF-NFO		SHEALED		SEA-HHA	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
f_1	-1.1e+03 +	2.4e+02	-8.7e+02 +	2.4e+02	2.3e+03 +	1.7e+03	-1.1e+03 +	3.3e+02	-1.4e+03 +	1.7e+01	**-1.4e+03 −**	**7.6e-04**	-1.4e+03	2.2e-01
f_2	2.0e+06 ≈	2.7e+06	1.9e+06 +	1.4e+06	2.6e+07 +	9.1e+07	3.7e+06 +	3.6e+06	4.4e+06 +	3.2e+06	**3.4e+05 ≈**	**4.1e+05**	8.1e+05	9.4e+05
f_3	1.0e+08 +	2.3e+08	4.1e+08 +	2.5e+08	5.9e+14 +	2.8e+15	2.0e+08 +	3.5e+08	3.9e+08 +	3.2e+08	1.3e+09 +	1.3e+09	**2.3e+07**	**5.6e+07**
f_4	2.0e+04 +	1.2e+04	1.1e+04 ≈	6.7e+03	2.8e+04 +	3.8e+04	1.5e+04 ≈	9.6e+03	2.3e+04 +	1.4e+04	**9.7e+03 ≈**	**6.0e+03**	9.8e+03	5.9e+03
f_5	-7.6e+02 +	2.2e+02	-7.2e+02 +	1.5e+02	1.7e+04 +	2.1e+04	-7.2e+02 +	2.0e+02	-7.4e+02 +	1.7e+02	-8.7e+02 +	1.4e+02	**-9.7e+02**	**2.1e+01**
f_6	-8.8e+02 +	2.6e+01	-8.7e+02 +	1.5e+01	-8.0e+02 +	3.5e+01	-8.8e+02 +	2.7e+01	-8.7e+02 +	1.6e+01	-8.1e+02 +	5.7e+01	**-8.9e+02**	**1.1e+01**
f_7	-7.8e+02 ≈	1.4e+01	-7.6e+02 +	1.3e+01	4.6e+08 +	2.4e+09	-7.8e+02 ≈	1.6e+01	-7.5e+02 +	2.1e+01	-7.2e+02 +	2.0e+01	**-7.8e+02**	**1.2e+01**
f_8	-6.8e+02 ≈	1.1e-01	-6.8e+02 ≈	1.1e-01	-6.8e+02 ≈	1.1e-01	**-6.8e+02 ≈**	**3.3e+00**	-6.8e+02 ≈	7.5e-01	-6.8e+02 ≈	1.7e-01	-6.8e+02	9.2e-02
f_9	-6.0e+02 +	9.1e-01	-6.0e+02 +	6.1e-01	-5.9e+02 +	4.6e-01	-6.0e+02 +	1.0e+00	-6.0e+02 +	5.8e-01	-6.0e+02 +	8.0e-01	**-6.0e+02**	**8.1e-01**
f_{10}	-4.7e+02 +	2.6e+01	-4.4e+02 +	2.9e+01	6.6e+01 +	6.1e+02	-4.4e+02 +	5.2e+01	-4.3e+02 +	3.3e+01	**-5.0e+02 −**	**1.9e-01**	-5.0e+02	1.7e-01
f_{11}	-3.8e+02 ≈	9.4e+00	-3.7e+02 +	7.7e+00	-2.7e+02 +	1.2e+01	-3.8e+02 ≈	1.1e+01	-3.6e+02 +	9.3e+00	-3.6e+02 +	9.0e+00	**-3.8e+02**	**7.3e+00**
f_{12}	**-2.8e+02 ≈**	**7.9e+00**	-2.7e+02 +	8.5e+00	-2.2e+02 +	9.6e+00	-2.8e+02 ≈	9.0e+00	-2.6e+02 +	9.1e+00	-2.5e+02 +	9.4e+00	-2.8e+02	6.5e+00
f_{13}	-1.8e+02 +	9.0e+00	-1.6e+02 +	6.9e+00	-1.2e+02 +	1.4e+01	-1.8e+02 +	1.4e+01	-1.6e+02 +	8.4e+00	-1.6e+02 +	1.1e+01	**-1.8e+02**	**6.9e+00**
f_{14}	4.8e+02 +	2.0e+02	3.5e+02 ≈	1.5e+02	7.9e+02 +	9.8e+01	4.5e+02 ≈	1.5e+02	5.5e+02 +	1.8e+02	5.3e+02 +	1.8e+02	**3.2e+02**	**1.8e+02**
f_{15}	6.7e+02 ≈	2.2e+02	7.3e+02 ≈	1.2e+02	1.0e+03 +	1.7e+02	7.1e+02 ≈	2.0e+02	8.8e+02 +	1.6e+02	9.1e+02 +	1.9e+02	**6.5e+02**	**1.4e+02**
f_{16}	2.0e+02 ≈	5.2e-01	2.0e+02 ≈	4.1e-01	2.0e+02 ≈	6.0e-01	**2.0e+02 −**	**5.0e-01**	2.0e+02 ≈	7.1e-01	2.0e+02 ≈	5.3e-01	2.0e+02	4.9e-01
f_{17}	3.2e+02 ≈	7.7e+00	3.3e+02 +	6.8e+00	3.5e+02 +	3.6e+00	3.3e+02 +	9.1e+00	3.3e+02 +	7.5e+00	3.4e+02 +	9.6e+00	**3.2e+02**	**5.1e+00**
f_{18}	4.2e+02 ≈	9.1e+00	4.3e+02 +	7.4e+00	4.5e+02 +	3.7e+00	4.3e+02 +	1.0e+01	4.3e+02 +	6.8e+00	4.4e+02 +	9.1e+00	**4.2e+02**	**3.3e+00**
f_{19}	5.0e+02 ≈	5.0e+00	5.1e+02 +	7.5e+00	6.3e+03 +	5.5e+03	5.0e+02 ≈	4.1e+00	5.4e+02 +	5.2e+01	5.5e+02 +	8.1e+01	**5.0e+02**	**1.0e+00**
f_{20}	**6.0e+02 ≈**	**4.3e-01**	6.0e+02 +	2.2e-01	6.0e+02 +	5.7e-02	6.0e+02 ≈	3.9e-01	6.0e+02 +	2.6e-01	6.0e+02 +	2.4e-01	6.0e+02	2.3e-01
f_{21}	1.1e+03 +	9.5e+01	1.1e+03 +	6.1e+01	1.3e+03 +	6.0e+01	1.1e+03 +	7.8e+01	1.2e+03 +	8.7e+01	1.0e+03 ≈	7.0e+01	**1.0e+03**	**8.0e+01**
f_{22}	1.6e+03 ≈	2.0e+02	1.6e+03 ≈	1.1e+02	1.8e+03 +	1.5e+02	1.6e+03 ≈	1.9e+02	1.7e+03 +	1.3e+02	1.6e+03 ≈	2.7e+02	**1.5e+03**	**1.8e+02**
f_{23}	1.7e+03 ≈	2.0e+02	1.8e+03 +	1.3e+02	2.1e+03 +	1.5e+02	1.7e+03 +	1.8e+02	1.9e+03 +	1.1e+02	1.8e+03 +	2.5e+02	**1.6e+03**	**1.6e+02**
f_{24}	1.2e+03 ≈	3.4e+01	1.2e+03 +	1.7e+01	1.2e+03 +	2.9e+01	1.2e+03 +	3.1e+01	1.2e+03 +	1.7e+01	1.2e+03 +	2.7e+01	**1.1e+03**	**3.3e+01**
f_{25}	1.2e+03 ≈	1.4e+01	1.2e+03 +	1.1e+01	1.3e+03 +	3.6e+00	1.2e+03 +	1.5e+01	1.2e+03 +	1.0e+01	1.2e+03 +	2.0e+01	**1.2e+03**	**1.6e+01**
f_{26}	1.3e+03 ≈	2.9e+01	1.3e+03 +	1.0e+01	1.5e+03 +	5.3e+01	1.3e+03 ≈	1.6e+01	1.4e+03 +	2.0e+01	1.3e+03 +	1.7e+01	**1.3e+03**	**2.6e+01**
f_{27}	1.7e+03 ≈	2.3e+01	1.7e+03 +	2.0e+01	1.7e+03 +	1.7e+01	1.7e+03 ≈	2.3e+01	1.7e+03 +	2.2e+01	1.7e+03 +	2.0e+01	**1.7e+03**	**2.7e+01**
f_{28}	1.8e+03 +	4.2e+01	1.8e+03 +	3.2e+01	2.0e+03 +	9.6e+01	1.8e+03 +	4.4e+01	1.8e+03 +	5.0e+01	1.7e+03 ≈	2.6e+01	**1.7e+03**	**4.1e+01**
	+/≈/−: 11/17/0		+/≈/−: 22/6/0		+/≈/−: 26/2/0		+/≈/−: 15/12/1		+/≈/−: 26/2/0		+/≈/−: 19/7/2			

Table 8.3: The experimental and statistical results on 10-D CEC2013 Suite.

Func	SFO		DSIDE		ROA		SCSO		aRBF-NFO		SHEALED		SEA-HHA	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
f_1	2.5e+03 +	1.8e+03	2.6e+03 +	1.1e+03	9.0e+03 +	2.4e+03	2.8e+03 +	2.4e+03	4.7e+02 +	1.5e+03	**-1.4e+03** −	**1.3e+01**	-9.3e+02	3.6e+02
f_2	2.8e+07 ≈	2.0e+07	3.2e+07 +	1.2e+07	7.7e+07 +	5.2e+07	2.1e+07 ≈	1.3e+07	6.0e+07 +	3.2e+07	**1.6e+07** ≈	**1.0e+07**	1.6e+07	9.9e+06
f_3	1.9e+10 +	1.8e+10	2.0e+10 +	9.6e+09	6.4e+11 +	1.3e+12	3.5e+10 +	1.3e+11	1.0e+10 +	5.5e+09	2.8e+10 +	2.5e+10	**6.2e+09**	**5.3e+09**
f_4	2.9e+04 ≈	1.3e+04	3.4e+04 ≈	1.4e+04	**2.2e+04** −	**7.9e+03**	2.4e+04 ≈	8.8e+03	5.3e+04 +	2.4e+04	4.9e+04 +	2.2e+04	3.3e+04	1.5e+04
f_5	5.2e+01 +	5.8e+02	4.2e+02 +	6.8e+02	1.0e+04 +	5.8e+03	4.9e+02 +	8.6e+02	7.7e+02 +	1.1e+03	**-7.1e+02** ≈	**9.9e+01**	-5.8e+02	3.4e+02
f_6	-6.1e+02 +	1.4e+02	-5.7e+02 +	1.2e+02	5.5e+01 +	2.8e+02	-6.4e+02 +	1.2e+02	-6.6e+02 +	8.7e+01	-5.9e+02 +	1.1e+02	**-8.2e+02**	**3.0e+01**
f_7	-5.9e+02 ≈	2.9e+02	-5.6e+02 +	7.1e+01	-6.7e+02 ≈	6.0e+01	-6.5e+02 ≈	1.6e+02	-6.3e+02 +	5.2e+01	-5.4e+02 +	1.1e+02	**-6.7e+02**	**4.8e+01**
f_8	-6.8e+02 ≈	9.5e-02	-6.8e+02 ≈	1.1e-01	**-6.8e+02** ≈	**9.5e-02**	-6.8e+02 ≈	8.3e-02	-6.8e+02 ≈	1.0e-01	-6.8e+02 ≈	1.3e-01	-6.8e+02	1.2e-01
f_9	-5.9e+02 ≈	1.8e+00	-5.9e+02 +	9.4e-01	-5.9e+02 +	7.4e-01	-5.9e+02 +	1.2e+00	-5.9e+02 +	1.0e+00	-5.9e+02 +	9.3e-01	**-5.9e+02**	**1.5e+00**
f_{10}	-5.0e+01 +	2.1e+02	5.7e+01 +	1.9e+02	1.1e+03 +	4.2e+02	-6.8e+01 +	2.1e+02	1.3e+02 +	3.0e+02	**-4.9e+02** −	**5.3e+00**	-3.4e+02	1.1e+02
f_{11}	-3.0e+02 +	2.9e+01	-2.6e+02 +	2.2e+01	-1.5e+02 +	2.8e+01	-2.9e+02 +	3.7e+01	-2.9e+02 +	3.1e+01	-2.9e+02 +	1.6e+01	**-3.2e+02**	**1.6e+01**
f_{12}	-2.0e+02 +	2.6e+01	-1.6e+02 +	2.1e+01	-6.9e+01 +	8.7e+00	-1.8e+02 +	2.8e+01	-1.7e+02 +	3.2e+01	-1.6e+02 +	2.3e+01	**-2.3e+02**	**1.6e+01**
f_{13}	-9.6e+01 +	2.9e+01	-7.3e+01 +	2.2e+01	2.1e+01 +	1.0e+01	-8.2e+01 +	3.2e+01	-7.2e+01 +	2.3e+01	-7.8e+01 +	1.7e+01	**-1.2e+02**	**1.4e+01**
f_{14}	1.8e+03 ≈	3.7e+02	1.7e+03 ≈	2.0e+02	2.0e+03 +	1.2e+02	1.7e+03 ≈	2.8e+02	1.8e+03 ≈	2.7e+02	2.1e+03 +	2.6e+02	**1.7e+03**	**2.8e+02**
f_{15}	2.0e+03 ≈	3.2e+02	2.1e+03 ≈	2.1e+02	2.2e+03 +	2.5e+02	**1.8e+03** ≈	**2.6e+02**	2.0e+03 ≈	3.1e+02	2.4e+03 +	2.1e+02	1.9e+03	2.7e+02
f_{16}	2.0e+02 ≈	4.1e-01	2.0e+02 ≈	4.2e-01	2.0e+02 ≈	4.5e-01	**2.0e+02** ≈	**6.2e-01**	2.0e+02 ≈	6.3e-01	2.0e+02 ≈	5.1e-01	2.0e+02	5.6e-01
f_{17}	4.2e+02 +	2.8e+01	4.4e+02 +	2.5e+01	4.9e+02 +	1.1e+01	4.1e+02 ≈	2.6e+01	4.0e+02 ≈	4.2e+01	4.0e+02 ≈	1.3e+01	**3.9e+02**	**2.2e+01**
f_{18}	5.2e+02 +	2.6e+01	5.4e+02 +	1.9e+01	5.8e+02 +	1.7e+01	5.3e+02 +	2.9e+01	**4.9e+02** ≈	**2.9e+01**	4.9e+02 ≈	9.6e+00	5.0e+02	2.4e+01
f_{19}	1.9e+03 +	3.1e+03	2.9e+03 +	2.4e+03	3.6e+04 +	1.3e+04	1.9e+03 +	2.6e+03	2.4e+03 +	2.6e+03	6.2e+03 +	5.6e+03	**5.3e+02**	**4.7e+01**
f_{20}	6.0e+02 ≈	4.2e-01	6.0e+02 +	2.0e-01	6.0e+02 +	2.1e-02	6.0e+02 ≈	3.7e-01	6.0e+02 +	2.1e-01	6.0e+02 +	1.9e-01	**6.0e+02**	**2.5e-01**
f_{21}	1.3e+03 +	7.3e+01	1.3e+03 +	5.0e+01	1.3e+03 +	5.0e+01	1.3e+03 +	6.1e+01	1.4e+03 +	1.1e+02	**1.2e+03** ≈	**4.1e+01**	1.2e+03	4.2e+01
f_{22}	2.9e+03 ≈	2.9e+02	2.8e+03 ≈	1.8e+02	3.4e+03 +	2.3e+02	2.9e+03 ≈	2.9e+02	2.8e+03 ≈	2.8e+02	3.1e+03 +	2.6e+02	**2.8e+03**	**2.9e+02**
f_{23}	3.1e+03 ≈	3.3e+02	3.3e+03 +	2.3e+02	3.7e+03 +	1.2e+02	3.2e+03 ≈	3.1e+02	3.2e+03 ≈	3.2e+02	3.4e+03 +	2.4e+02	**3.0e+03**	**2.6e+02**
f_{24}	1.2e+03 +	5.2e+00	1.2e+03 +	8.3e+00	1.3e+03 +	5.6e+00	1.2e+03 +	4.6e+00	1.2e+03 +	2.9e+00	1.2e+03 +	2.9e+00	**1.2e+03**	**4.3e+00**
f_{25}	1.3e+03 +	5.2e+00	1.3e+03 +	5.8e+00	1.3e+03 +	6.6e-01	1.3e+03 +	4.2e+00	1.3e+03 +	3.1e+00	1.3e+03 +	3.3e+00	**1.3e+03**	**4.6e+00**
f_{26}	1.4e+03 +	5.6e+01	1.4e+03 +	1.3e+01	1.5e+03 +	4.1e+00	1.4e+03 +	5.3e+01	1.4e+03 +	3.3e+01	1.5e+03 +	4.3e+01	**1.4e+03**	**5.0e+01**
f_{27}	2.0e+03 +	7.5e+01	1.9e+03 ≈	7.3e+01	2.2e+03 +	6.5e+01	2.0e+03 +	6.7e+01	2.0e+03 +	4.5e+01	2.0e+03 +	5.4e+01	**1.9e+03**	**7.4e+01**
f_{28}	2.4e+03 +	1.1e+02	2.5e+03 +	5.2e+01	2.6e+03 +	6.0e+01	2.4e+03 +	1.0e+02	2.5e+03 +	1.7e+02	2.4e+03 +	9.8e+01	**2.2e+03**	**1.6e+02**
	+/≈/−: 17/11/0		+/≈/−: 21/7/0		+/≈/−: 24/3/1		+/≈/−: 17/11/0		+/≈/−: 20/8/0		+/≈/−: 19/7/2			

Table 8.4: The experimental and statistical results on 30-D CEC2013 Suite.

Func	SFO		DSIDE		ROA		SCSO		aRBF-NFO		SHEALED		SEA-HHA	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
f_1	3.9e+04 +	5.2e+03	4.3e+04 +	7.1e+03	6.2e+04 +	4.1e+03	4.5e+04 +	6.6e+03	2.4e+04 +	5.9e+03	**8.0e+03 −**	**2.6e+03**	2.0e+04	6.3e+03
f_2	6.5e+08 +	2.1e+08	7.8e+08 +	1.6e+08	2.7e+09 +	8.4e+08	9.3e+08 +	4.0e+08	7.7e+08 +	2.3e+08	1.1e+09 +	3.2e+08	**3.7e+08**	**1.7e+08**
f_3	4.5e+16 +	1.3e+17	5.1e+17 +	1.4e+18	5.0e+21 +	1.0e+22	5.8e+17 +	2.0e+18	7.2e+14 +	3.1e+15	9.9e+16 +	2.5e+17	**1.8e+14**	**9.4e+14**
f_4	**6.7e+04 −**	**5.0e+03**	1.1e+05 ≈	3.0e+04	7.1e+04 −	8.3e+03	6.9e+04 −	3.1e+03	1.4e+05 ≈	3.0e+04	1.8e+05 +	9.4e+04	1.3e+05	2.9e+04
f_5	1.4e+04 +	4.8e+03	1.7e+04 +	5.2e+03	5.8e+04 +	1.5e+04	1.6e+04 +	7.4e+03	1.7e+04 +	7.3e+03	1.0e+04 +	2.2e+03	**7.3e+03**	**4.4e+03**
f_6	5.9e+03 +	2.2e+03	5.9e+03 +	1.5e+03	1.7e+04 +	3.7e+03	7.1e+03 +	2.4e+03	2.5e+03 +	1.7e+03	2.6e+03 +	8.7e+02	**9.1e+02**	**8.4e+02**
f_7	2.1e+05 +	4.6e+05	3.2e+05 +	4.1e+05	7.2e+07 +	8.3e+07	4.2e+05 +	8.8e+05	2.3e+04 +	8.4e+04	2.1e+05 +	3.7e+05	**4.0e+03**	**1.5e+04**
f_8	-6.8e+02 ≈	6.0e-02	-6.8e+02 ≈	6.2e-02	**-6.8e+02** ≈	**7.7e-02**	-6.8e+02 ≈	6.3e-02	-6.8e+02 ≈	6.8e-02	-6.8e+02 ≈	6.7e-02	-6.8e+02	7.0e-02
f_9	-5.6e+02 ≈	2.7e+00	-5.6e+02 +	1.5e+00	-5.5e+02 +	1.5e+00	-5.6e+02 ≈	3.5e+00	-5.6e+02 +	1.9e+00	-5.5e+02 +	1.7e+00	**-5.6e+02**	**3.4e+00**
f_{10}	5.3e+03 +	1.1e+03	5.4e+03 +	1.0e+03	1.2e+04 +	1.8e+03	6.4e+03 +	1.7e+03	5.6e+03 +	1.4e+03	5.3e+03 +	1.2e+03	**2.4e+03**	**9.5e+02**
f_{11}	3.2e+02 +	1.1e+02	3.6e+02 +	6.8e+01	6.6e+02 +	8.1e+01	3.8e+02 +	1.2e+02	3.5e+02 +	8.9e+01	3.5e+02 +	1.0e+02	**1.8e+02**	**1.0e+02**
f_{12}	4.1e+02 +	1.1e+02	4.5e+02 +	8.2e+01	8.4e+02 +	6.1e+01	4.8e+02 +	1.1e+02	4.6e+02 +	1.1e+02	4.8e+02 +	1.1e+02	**2.9e+02**	**1.1e+02**
f_{13}	5.2e+02 +	9.0e+01	5.9e+02 +	6.0e+01	9.3e+02 +	7.3e+01	6.3e+02 +	9.7e+01	6.0e+02 +	1.4e+02	5.5e+02 +	8.8e+01	**4.0e+02**	**9.1e+01**
f_{14}	8.3e+03 ≈	4.3e+02	8.3e+03 ≈	3.5e+02	8.5e+03 ≈	4.6e+02	8.2e+03 ≈	5.9e+02	**7.5e+03 −**	**7.1e+02**	8.7e+03 +	3.4e+02	8.3e+03	4.4e+02
f_{15}	**8.3e+03** ≈	**7.0e+02**	8.8e+03 ≈	4.3e+02	9.3e+03 +	5.5e+02	8.5e+03 ≈	4.2e+02	8.4e+03 ≈	7.8e+02	9.0e+03 +	3.9e+02	8.6e+03	3.7e+02
f_{16}	2.0e+02 ≈	7.2e-01	2.0e+02 ≈	5.8e-01	2.0e+02 ≈	6.3e-01	**2.0e+02 −**	**6.7e-01**	2.0e+02 ≈	7.5e-01	2.0e+02 ≈	8.5e-01	2.0e+02	5.7e-01
f_{17}	1.1e+03 ≈	1.0e+02	1.3e+03 +	1.0e+02	1.3e+03 +	2.1e+01	1.2e+03 +	8.7e+01	1.5e+03 +	2.9e+02	**7.7e+02 −**	**5.6e+01**	1.1e+03	1.6e+02
f_{18}	1.2e+03 ≈	7.3e+01	1.4e+03 +	9.1e+01	1.4e+03 +	2.2e+01	1.3e+03 +	7.4e+01	1.7e+03 +	2.4e+02	**8.7e+02 −**	**6.0e+01**	1.2e+03	1.9e+02
f_{19}	4.2e+05 +	2.8e+05	8.4e+05 +	3.3e+05	5.1e+05 +	1.0e+05	2.8e+05 +	1.2e+05	7.9e+05 +	7.5e+05	8.6e+05 +	5.3e+05	**1.8e+05**	**2.0e+05**
f_{20}	6.1e+02 +	1.5e-04	6.1e+02 ≈	2.8e-08	6.2e+02 +	2.7e-08	6.1e+02 ≈	1.5e-02	6.1e+02 ≈	2.3e-05	6.1e+02 +	1.7e-09	**6.1e+02**	**1.6e-02**
f_{21}	**3.2e+03 −**	**1.0e+02**	3.5e+03 ≈	1.6e+02	3.3e+03 ≈	6.5e+01	3.2e+03 −	7.4e+01	3.9e+03 +	3.6e+02	4.8e+03 +	4.6e+02	3.5e+03	2.5e+02
f_{22}	9.8e+03 +	5.7e+02	9.8e+03 +	4.1e+02	9.6e+03 ≈	3.7e+02	9.6e+03 ≈	6.1e+02	**9.3e+03** ≈	**5.2e+02**	1.0e+04 +	2.7e+02	9.4e+03	4.8e+02
f_{23}	9.6e+03 ≈	5.8e+02	1.0e+04 ≈	3.1e+02	1.1e+04 +	3.6e+02	1.0e+04 ≈	4.0e+02	**9.5e+03** ≈	**8.1e+02**	1.0e+04 +	4.1e+02	9.7e+03	5.2e+02
f_{24}	1.4e+03 +	2.3e+01	1.4e+03 +	3.5e+01	1.9e+03 +	1.1e+02	1.4e+03 +	3.7e+01	1.3e+03 +	1.1e+01	1.3e+03 +	9.8e+00	**1.3e+03**	**9.5e+00**
f_{25}	1.5e+03 +	1.7e+01	1.5e+03 +	1.8e+01	1.5e+03 +	1.1e+01	1.5e+03 +	2.1e+01	1.4e+03 +	6.6e+00	1.5e+03 +	7.3e+00	**1.4e+03**	**1.1e+01**
f_{26}	1.6e+03 +	6.5e+01	**1.5e+03 −**	**3.8e+01**	2.2e+03 +	1.9e+02	1.6e+03 ≈	7.6e+01	1.6e+03 ≈	5.9e+01	1.6e+03 ≈	4.6e+01	1.6e+03	7.3e+01
f_{27}	2.8e+03 +	8.3e+01	2.9e+03 +	5.9e+01	3.6e+03 +	2.7e+02	2.8e+03 +	1.1e+02	2.8e+03 +	4.6e+01	2.8e+03 +	5.0e+01	**2.7e+03**	**7.7e+01**
f_{28}	6.8e+03 +	6.8e+02	8.6e+03 +	7.0e+02	1.0e+04 +	7.7e+02	7.2e+03 +	7.0e+02	6.8e+03 +	7.3e+02	8.1e+03 +	7.0e+02	**6.4e+03**	**5.2e+02**
	+/≈/−: 18/8/2		+/≈/−: 19/8/1		+/≈/−: 22/5/1		+/≈/−: 17/8/3		+/≈/−: 19/8/1		+/≈/−: 22/3/3			

Table 8.5: The optimization results on the Cantilever Beam Design problem.

Alg.	worst	mean	best	std
SFO	2.6355	1.6819 ≈	1.3817	0.3057
DSIDE	2.2361	1.9216 +	1.5831	0.2121
ROA	**1.9586**	**1.6427** ≈	1.3872	**0.1710**
SCSO	2.3415	1.6918 ≈	1.4204	0.2177
aRBF-NFO	5.9947	4.1311 +	2.2430	0.9052
SHEALED	2.1880	1.8121 +	**1.3539**	0.2142
SEA-HHA	2.1469	1.6504	1.4036	0.2175

Table 8.6: The optimum found by optimization techniques on the Cantilever Beam Design problem.

Alg.	x_1	x_2	x_3	x_4	x_5	cons.	obj.
SFO	6.534790	4.653559	4.241771	4.377597	2.335527	-0.0033	1.3817
DSIDE	9.409405	4.737502	5.103161	3.361869	2.758319	-0.2039	1.5831
ROA	6.058160	6.415025	4.260142	3.553657	1.944015	-0.0476	1.3872
SCSO	5.745827	5.346359	4.685947	3.114073	3.872125	-0.0026	1.4204
aRBF-NFO	9.324028	7.978926	4.897215	8.857559	4.889085	-0.6715	2.2430
SHEALED	**5.752486**	**5.717425**	**4.170125**	**3.936549**	**2.121631**	**-0.0001**	**1.3539**
SEA-HHA	5.3868787	5.417096	5.144122	3.389277	3.157643	-0.0258	1.4036

Table 8.7: The optimization results on the Tension/Compression Spring Design problem. If more than 1 of 30 trial runs does not find a feasible solution, the worst, mean, and std cannot be calculated and we manually fill them with Nan. Statistical analysis is also meaningless.

Alg.	worst	mean	best	std
SFO	Nan	Nan	0.0132	Nan
DSIDE	0.1551	0.0433 ≈	0.0138	0.0286
ROA	0.1623	0.0483 ≈	0.0143	0.0379
SCSO	Nan	Nan	0.0131	Nan
aRBF-NFO	Nan	Nan	0.0204	Nan
SHEALED	**0.0420**	**0.0230** ≈	**0.0130**	**0.0085**
SEA-HHA	0.0548	0.0314	0.0135	0.0100

8.4 Discussion

8.4.1 Performance analysis of optimization on CEC2013

From the overview performance on CEC2013 benchmark functions among seven optimization techniques, our proposed SEA-HHA is competitive with these advanced algorithms,

Table 8.8: The optimum found by optimization techniques on the Tension/Compression Spring Design problem.

Alg.	x_1	x_2	x_3	cons.1	cons.2	cons.3	cons.4	obj.
SFO	0.053517	0.393260	9.745187	-0.0065	-0.0184	-3.9872	-0.7021	0.0132
DSIDE	0.056095	0.466966	7.392016	-0.0589	-0.0094	-3.8877	-0.6512	0.0138
ROA	0.058282	0.515254	6.227495	-0.0285	-0.0346	-3.9510	-0.6176	0.0143
SCSO	0.053946	0.412892	8.965602	-0.0380	-0.0011	-3.9571	-0.6887	0.0131
aRBF-NFO	0.066801	0.812510	3.645216	-0.3678	-0.0301	-2.8987	-0.4137	0.0204
SHEALED	**0.050000**	**0.315625**	**14.575956**	**-0.0214**	**-0.0044**	**-3.8362**	**-0.7562**	**0.0130**
SEA-HHA	0.058430	0.540130	5.358699	-0.0091	-0.0023	-4.2493	-0.6009	0.0135

Table 8.9: The optimization results on the Pressure Vessel Design problem.

Alg.	worst	mean	best	std
SFO	182308.88	37182.20 $\approx$	6811.07	53171.32
DSIDE	286301.67	71823.97 +	16446.54	49198.51
ROA	216645.45	69447.93 +	9316.22	47366.32
SCSO	240564.27	85641.49 +	21920.95	50317.78
aRBF-NFO	192761.70	25873.81 $\approx$	7117.16	38378.12
SHEALED	155058.38	42531.62 +	6811.27	31670.85
SEA-HHA	**29877.51**	**13509.01**	**6648.46**	**4690.01**

Table 8.10: The optimum found by optimization techniques on the Pressure Vessel Design problem.

Alg.	x_1	x_2	x_3	x_3	cons.1	cons.2	cons.3	cons.4	obj.
SFO	0.97065	0.56005	48.77169	108.81143	-0.02	-0.09	-3080.51	-131.18	6811.07
DSIDE	1.57724	1.31905	43.38856	195.95323	-0.73	-0.90	-205066.52	-44.04	16446.54
ROA	0.93220	0.82358	46.70452	178.11682	-0.03	-0.37	-351339.89	-61.88	9316.22
SCSO	1.42960	0.85228	72.49971	155.19587	-0.03	-0.16	-2862965.50	-84.80	21920.95
aRBF-NFO	1.14105	0.58046	56.86348	51.92381	-0.04	-0.03	-1626.53	-188.07	7117.10
SHEALED	1.00300	0.46802	47.03495	123.86394	-0.09	-0.01	-730.81	-116.13	6811.27
SEA-HHA	**1.00388**	**0.53724**	**51.45276**	**87.45941**	**-0.01**	**-0.04**	**-1978.28**	**-152.54**	**6648.46**

and we will analyze the performance of SEA-HHA from two perspectives: exploitation ability and exploration ability.

Exploitation ability of SEA-HHA: In the CEC2013 suite, f_1 to f_5 are unimodal functions so they can evaluate the exploitation ability of optimization algorithms. Meanwhile, we notice that SHEALED is significantly better than our proposed SEA-HHA in f_1 of three scales, which means this method is good at dealing with this kind of problem. Except for

f_1, SEA-HHA is at least approximately equal to or significantly better than SHEALED, which shows the superior exploitation ability of SEA-HHA. Compared with other algorithms in unimodal functions, our proposal performs significantly better than them in most cases, which further proves the exploitation capacity in practice. Besides, the deterioration of SEA-HHA on f_4 can be observed as the dimension of the problem increases, and we infer that one reason is due to the curse of dimensionality.[224] As the dimension of the problem increases, the search space will increase exponentially, and the presence of this phenomenon can degenerate the accuracy of the surrogate model rapidly and further affect the quality of estimated solutions.

Exploration ability of SEA-HHA: Since f_6 to f_{20} are multimodal functions and f_{21} to f_{28} are composition functions, these functions have complex fitness landscape characteristics and multiple local optima so that they are allowed to evaluate the exploration capacity of optimization techniques. Through the experimental and statistical results in Table 8.2, 8.3, and 8.4, the superior performance of SEA-HHA can be observed in practice, and we owe this excellent performance to the diverse search strategy and effective surrogate-assisted estimation. However, the slight degeneration caused by the curse of dimensionality exists in some benchmark functions such as f_{17}, f_{18}, and f_{21}, and how to overcome this issue will be considered in our future research.

8.4.2 Performance analysis of optimization on three engineering problems

These engineering optimization problems contain multiple constraints and complex fitness landscapes, the optimization performance on these problems can reflect the ability of the algorithm to deal with the real-world tasks. Besides, this research focuses on solving EOPs, and only 1000 FEs are assigned for each task optimization, which is a severe challenge for optimization techniques.

Statistical results in Table 8.6, 8.9, and 8.10 show that SEA-HHA at least is not inferior to any optimization method for any problem and can outperform in some problems (e.g. compared with DSIDE, aRBF-NFO, and SHEALED in Cantilever Beam Design). Another advantage of SEA-HHA is that the optimization process is stable even under the FEs limitation. In the Tension/Compression Spring Design problem, SEA-HHA can find a feasible solution in any independent trial run while SFO, SCSO, and aRBF-NFO can not at least once. In the Pressure Vessel Design, the worst solution found by SEA-HHA is apparently better than the compared methods and the standard deviation is also small. These experimental results reveal the excellent exploration and exploitation abilities of SEA-HHA in engineering optimization problems, which has great potential to deal with real-world applications.

8.5 Conclusion

This chapter proposes a novel surrogate ensemble-assisted hyper-heuristic algorithm (SEA-HHA) to solve EOPs. In the high-level component design, the random selection function based on the pre-defined probabilities is adopted to dominate the LLHs. In the low-level component, we design four search strategy archives as LLHs: exploration strategy archive, exploitation strategy archive, surrogate-assisted estimation archive, and mutation strategy archive, each search strategy is easily implemented. Besides, in the surrogate-assisted estimation archive, three different data selection principles are applied for model construction: $All\ Data$, $Recent\ Data$, and $Neighbor$, which correspond to the global and local concepts, and the most accurate model constructed by PR, SVR, and GPR is utilized to estimate the promising solutions.

In the numerical experiments, we compare our proposed SEA-HHA with six advanced optimization techniques on the CEC2013 benchmark functions and three popular engineering optimization problems. Experimental and statistical results show that SEA-HHA has

broad prospects for solving EOPs.

Chapter 9

Conclusion and Future Work

This dissertation solves LSOPs using EC approaches through two aspects: (1). Using the CC framework to separate decision variables and form median- and small-scale problems and (2). Developing high-quality EC technique.

In Chapter 3, we identify a limitation in existing DG-based decomposition methods when applied in noisy environments. Specifically, these methods are unable to accurately determine the separability between decision variables due to the influence of noise on fitness differences. We analyze this limitation both mathematically and through numerical experiments, leading us to propose a novel approach called linkage measurement minimization (LMM). In LMM, we introduce a linkage measurement function and treat the decomposition as a combinatorial optimization problem. To optimize this problem, we employ the elitism genetic algorithm (EGA), resulting in a comprehensive methodology for noisy environments. We provide a mathematical explanation of the feasibility of LMM in noisy environments, demonstrating its effectiveness in addressing the deficiencies of existing methods. Furthermore, by integrating the modified differential evolution with distance-based selection (MDE-DS), our proposed approach exhibits significant superiority over current decomposition methods in noisy environments. This approach not only presents a theoretical foundation but also offers a practically feasible solution for tackling LSOPs

in noise environments through the CC framework. The proposed LMM demonstrates its efficacy in decomposing noisy LSOPs by integrating theoretical insights with practical implementations. This integration between theory and practice empowers researchers to confidently address LSOPs in noisy environments.

In Chapter 4, we address the limitation of real-coded linkage identification by nonlinearity check (LINC-R), primarily its computational complexity, which hinders its application in large-scale expensive optimization problems (LSEOPs). Fortunately, the advent of recursive differential grouping (RDG) and efficient RDG (ERDG) has substantially reduced the computational complexity from $O(D^2)$ to $O(D \cdot \log D)$, making it feasible to extend these methods to LSEOPs. We incorporate the multiplicative interactions identification principle from dual differential grouping (DDG) into ERDG, leading to the development of a novel decomposition methodology termed efficient DDG (EDDG). Additionally, we propose an efficient surrogate ensemble-assisted differential evolution (DE), which leverages both global and local surrogate models constructed using generalized regression neural network (GRNN) and Gaussian process regression (GPR). Through comprehensive experimental results and statistical analyses, we demonstrate the competitiveness of our proposed methodology in effectively addressing LSEOPs. This research challenges the LSEOPs by proposing a complete methodology of SEADECC-EDDG, which provides a potential solution for solving academic and real-world tasks significantly.

Chapter 5 and Chapter 6 delve into two recently proposed meta-heuristic algorithms: Vegetation Evolution (VEGE) and the RIME algorithm. We introduce multiple efficient search strategies to enhance these algorithms' performance in specific optimization tasks. Additionally, Chapter 7 and Chapter 8 propose an evolutionary multi-mode slime mold optimization (EMSMO) and a surrogate ensemble-assisted hyper-heuristic algorithm (SEA-HHA), both of which adhere to the hyper-heuristic framework. Through comprehensive numerical experiments spanning various problem domains, we confirm the efficiency and effectiveness of our methodologies. These innovative optimization algorithms offer ad-

vanced methodologies tailored for solving academic benchmark functions and real-world applications, including scheduling problems, protein structure prediction, resource management, and so on.

In future research endeavors, we aim to develop high-quality methodologies and problem-specific algorithms further to tackle diverse optimization problems in both academic domains and real-world scenarios.

Author's Publications

1. Rui Zhong, Enzhi Zhang, and Masaharu Munetomo *Cooperative coevolutionary differential evolution with linkage measurement minimization for large-scale optimization problems in noisy environments*. Complex & Intelligent Systems, pp. 4439–4456, Springer (2023) (IF: 6.7)

2. Rui Zhong, Fei Peng, Enzhi Zhang, Jun Yu, and Masaharu Munetomo *Vegetation Evolution with Dynamic Maturity Strategy and Diverse Mutation Strategy for Solving Optimization Problems*. Biomimetics, MDPI (2023) (IF: 4.5)

3. Rui Zhong, Enzhi Zhang, and Masaharu Munetomo *Cooperative coevolutionary surrogate ensemble-assisted differential evolution with efficient dual differential grouping for large-scale expensive optimization problems*. Complex & Intelligent Systems, pp. 2129–2149, Springer (2023), (IF: 5.8)

4. Rui Zhong, Jun Yu, Chao Zhang, and Masaharu Munetomo *Surrogate Ensemble-Assisted Hyper-Heuristic Algorithm for Expensive Optimization Problems*. International Journal of Computational Intelligent Systems, Springer (2023) (IF: 2.9)

5. Rui Zhong, Fei Peng, Enzhi Zhang, Jun Yu, and Masaharu Munetomo *Q-learning based vegetation evolution for numerical optimization and wireless sensor network coverage optimization*. Alexandria Engineering Journal, pp. 148-163, Elsevier (2024) (IF: 6.8)

6. Rui Zhong, Jun Yu, Chao Zhang, and Masaharu Munetomo *SRIME: a strengthened RIME with Latin hypercube sampling and embedded distance-based selection for engineering optimization problems*. Neural Computing and Applications, pp. 6721–6740, Springer (2024) (IF: 6)

7. Rui Zhong, Enzhi Zhang, and Masaharu Munetomo *Evolutionary multi-mode slime*

mold optimization: a hyper-heuristic algorithm inspired by slime mold foraging behaviors. The Journal of Supercomputing, Springer (2024) (IF: 3.3)

8. Enzhi Zhang, Bochen Dong, Mohamed Wahib, Rui Zhong, and Masaharu Munetomo *Meta Generative Image and Text Data Augmentation Optimization*. The Journal of Supercomputing, Springer (2024) (IF: 3.3)

9. Enzhi Zhang, Mohamed Wahib, Rui Zhong, and Masaharu Munetomo *Learning from the Past Training Trajectories: Regularization by Validation*. Journal of Advanced Computational Intelligence and Intelligent Informatics, pp. 67-78, Fuji Technology Press (2024) (IF: 0.7)

10. Xuefeng Xu, Rui Zhong, Chao Zhang, and Jun Yu *Multiplayer Battle Game-Inspired Optimizer for Complex Optimization Problems*. Cluster Computing, Springer (2024) (IF: 4.4)

11. Rui Zhong, Enzhi Zhang, and Masaharu Munetomo *Cooperative Coevolutionary Differential Evolution with Adjacent Intensity Matrix with Linkage Identification for Large-scale Optimization Problems in Noisy Environments*. Evolutionary Intelligence, Springer (2024) (IF: 2.6)

12. Rui Zhong and Masaharu Munetomo *Random Population-based Decomposition Method by Linkage Identification with Non-linearity Minimization on Graph*. The 27th International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 27), Springer (2022), Accepted

13. Rui Zhong and Masaharu Munetomo *Cooperative Coevolutionary NSGA-II with Linkage Measurement Minimization for Large-Scale Multi-objective Optimization*. International Conference on Evolutionary Multi-Criterion Optimization (EMO 2023), pp 43–55, Springer (2023)

14. Rui Zhong, Enzhi Zhang, and Masaharu Munetomo *A Hierarchical Cooperative Coevolutionary Approach to Solve Very Large-Scale Traveling Salesman Problem*. Optimization and Learning (OLA 2023), pp. 74–84, Springer (2023)

15. Rui Zhong, Binnan Tu, Enzhi Zhang, and Masaharu Munetomo *Adjacent Intensity Matrix with Linkage Identification for Large-Scale Optimization in Noisy Environments*. 2023 IEEE Congress on Evolutionary Computation (CEC 2023), pp. 01-10, IEEE (2023)

16. Enzhi Zhang, Ruiqing Wang, Mohamed Wahib, Rui Zhong and Masaharu Munetomo *Training Knowledge Inheritance Through Deep Q-Net*. 2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Honolulu, Oahu, HI, USA, 2023, pp. 899-904.

17. Rui Zhong, Enzhi Zhang, and Masaharu Munetomo *Evolutionary multi-mode slime mold optimization: A hyper-heuristic algorithm inspired by slime mold foraging behaviors*. The 28th International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 28), Springer (2023), Accepted

18. Enzhi Zhang, Mohamed Wahib, Rui Zhong and Masaharu Munetomo *Validation Loss Landscape Exploration with Deep Q-Learning*. International Joint Conference on Neural Networks (IJCNN), Springer (2024), Accepted

19. Rui Zhong, Yang Cao, Enzhi Zhang, and Masaharu Munetomo *Introducing Competitive Mechanism to Differential Evolution for Numerical Optimization*. The 29th International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 29), Springer (2024), Accepted

20. Rui Zhong, Jun Yu, and Masaharu Munetomo *Hyper-heuristic Differential Evolution with Novel Boundary Repair for Numerical Optimization*. The 29th Interna-

tional Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 29), Springer (2024), Accepted

21. Rui Zhong, Yang Cao, Jun Yu, and Masaharu Munetomo *Hierarchical Adaptive Differential Evolution with Local Search for Extreme Learning Machine*. The Fifteenth International Conference on Swarm Intelligence (ICSI 2024), Springer (2024), Accepted

Bibliography

[1] Choong, S. S.; Wong, L.-P.; Lim, C. P. Automatic design of hyper-heuristic based on reinforcement learning. Information Sciences. 2018; pp 89–107.

[2] Chen, M.; Du, W.; Tang, Y.; Jin, Y.; Yen, G. G. A Decomposition Method for Both Additively and Non-additively Separable Problems. IEEE Transactions on Evolutionary Computation. 2022; pp 1–1.

[3] Liang, Y.; Niu, D.; Hong, W.-C. Short term load forecasting based on feature extraction and improved general regression neural network model. Energy. 2019; pp 653–663.

[4] Su, H.; Zhao, D.; Heidari, A. A.; Liu, L.; Zhang, X.; Mafarja, M.; Chen, H. RIME: A physics-based optimization. Neurocomputing. 2023; pp 183–214.

[5] Ternois, M.; Mougon, M.; Flahaut, E.; Dussutour, A. Slime molds response to carbon nanotubes exposure: from internalization to behavior. Nanotoxicology. 2021; pp 511–526.

[6] Sastry, K.; Goldberg, D. E.; Llora, X. Towards Billion-Bit Optimization via a Parallel Estimation of Distribution Algorithm. Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation. New York, NY, USA, 2007; p 577–584.

[7] Deb, K.; Myburgh, C. Breaking the Billion-Variable Barrier in Real-World Optimization Using a Customized Evolutionary Algorithm. Proceedings of the Genetic

and Evolutionary Computation Conference 2016. New York, NY, USA, 2016; p 653–660.

[8] Deb, K.; Myburgh, C. A population-based fast algorithm for a billion-dimensional resource allocation problem with integer variables. European Journal of Operational Research. 2017; pp 460–474.

[9] Omidvar, M. N.; Su, Y.; Li, X. Large-Scale Optimization and Learning. Proceedings of the Companion Conference on Genetic and Evolutionary Computation. New York, NY, USA, 2023; p 1477–1502.

[10] Dehghani, M.; Montazeri, Z.; Trojovská, E.; Trojovský, P. Coati Optimization Algorithm: A new bio-inspired metaheuristic algorithm for solving optimization problems. Knowledge-Based Systems. 2023; p 110011.

[11] Zitouni, F.; Harous, S.; Belkeram, A.; Hammou, L. The Archerfish Hunting Optimizer: A Novel Metaheuristic Algorithm for Global Optimization. Arabian Journal for Science and Engineering. 2022.

[12] Zhong, C.; Li, G.; Meng, Z. Beluga whale optimization: A novel nature-inspired metaheuristic algorithm. Knowledge-Based Systems. 2022; p 109215.

[13] Rahmani, A. M.; AliAbdi, I. Plant competition optimization: A novel metaheuristic algorithm. Expert Systems. 2022; p e12956.

[14] Daliri, A.; Asghari, A.; Azgomi, H.; Alimoradi, M. The water optimization algorithm: a novel metaheuristic for solving optimization problems. Applied Intelligence. 2022.

[15] Potter, M.; De Jong, K. A cooperative coevolutionary approach to function optimization. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 1994; pp 249–257.

[16] van den Bergh, F.; Engelbrecht, A. A Cooperative approach to particle swarm optimization. IEEE Transactions on Evolutionary Computation. 2004; pp 225–239.

[17] Omidvar, M. N.; Li, X.; Mei, Y.; Yao, X. Cooperative Co-Evolution With Differential Grouping for Large Scale Optimization. IEEE Transactions on Evolutionary Computation. 2014; pp 378–393.

[18] Chen, Z.; Francis, A.; Li, S.; Liao, B.; Xiao, D.; Ha, T. T.; Li, J.; Ding, L.; Cao, X. Egret Swarm Optimization Algorithm: An Evolutionary Computation Approach for Model Free Optimization. Biomimetics. 2022.

[19] Yang, Z.; Tang, K.; Yao, X. Differential evolution for high-dimensional function optimization. 2007 IEEE Congress on Evolutionary Computation. 2007; pp 3523–3530.

[20] Yang, Z.; Tang, K.; Yao, X. Large scale evolutionary optimization using cooperative coevolution. Information Sciences. 2008; pp 2985–2999.

[21] Li, X.; Yao, X. Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms. 2009 IEEE Congress on Evolutionary Computation. 2009; pp 1546–1553.

[22] Yang, Z.; Tang, K.; Yao, X. Multilevel cooperative coevolution for large scale optimization. 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence). 2008; pp 1663–1670.

[23] Omidvar, M. N.; Mei, Y.; Li, X. Effective decomposition of large-scale separable continuous functions for cooperative co-evolutionary algorithms. 2014 IEEE Congress on Evolutionary Computation (CEC). 2014; pp 1305–1312.

[24] Trunfio, G. A. A Cooperative Coevolutionary Differential Evolution Algorithm with

Adaptive Subcomponents. Procedia Computer Science. 2015; pp 834–844, International Conference On Computational Science, ICCS 2015.

[25] Trunfio, G. A.; Topa, P.; Was, J. A new algorithm for adapting the configuration of subcomponents in large-scale optimization with cooperative coevolution. Information Sciences. 2016; pp 773–795.

[26] Sun, M.; Sun, C.; Li, X.; Zhang, G.; Akhtar, F. Surrogate ensemble assisted large-scale expensive optimization with random grouping. Information Sciences. 2022; pp 226–237.

[27] Sun, M.; Sun, C.; Li, X.; Zhang, G.; Akhtar, F. Large-Scale Expensive Optimization with a Switching Strategy. Complex System Modeling and Simulation. 2022; pp 253–263.

[28] Gu, H.; Wang, H.; Jin, Y. Surrogate-Assisted Differential Evolution with Adaptive Multi-Subspace Search for Large-Scale Expensive Optimization. IEEE Transactions on Evolutionary Computation. 2022; pp 1–1.

[29] Tezuka, M.; Munetomo, M.; Akama, K. Linkage identification by nonlinearity check for real-coded genetic algorithms. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2004; pp 222–233.

[30] Munetomo, M.; Goldberg, D. E. Linkage Identification by Non-monotonicity Detection for Overlapping Functions. Evolutionary Computation. 1999; pp 377–398.

[31] Mei, Y.; Omidvar, M. N.; Li, X.; Yao, X. A Competitive Divide-and-Conquer Algorithm for Unconstrained Large-Scale Black-Box Optimization. ACM Trans. Math. Softw. New York, NY, USA, 2016.

[32] Ling, Y.; Li, H.; Cao, B. Cooperative co-evolution with graph-based differential grouping for large scale global optimization. 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD). 2016; pp 95–102.

[33] Cao, B.; Zhao, J.; Gu, Y.; Ling, Y.; Ma, X. Applying graph-based differential grouping for multiobjective large-scale optimization. Swarm and Evolutionary Computation. 2020; p 100626.

[34] Omidvar, M. N.; Yang, M.; Mei, Y.; Li, X.; Yao, X. DG2: A Faster and More Accurate Differential Grouping for Large-Scale Black-Box Optimization. IEEE Transactions on Evolutionary Computation. 2017; pp 929–942.

[35] Sun, Y.; Kirley, M.; Halgamuge, S. K. A Recursive Decomposition Method for Large Scale Continuous Optimization. IEEE Transactions on Evolutionary Computation. 2018; pp 647–661.

[36] Yang, M.; Zhou, A.; Li, C.; Yao, X. An Efficient Recursive Differential Grouping for Large-Scale Continuous Problems. IEEE Transactions on Evolutionary Computation. 2021; pp 159–171.

[37] Ma, X.; Huang, Z.; Li, X.; Wang, L.; Qi, Y.; Zhu, Z. Merged Differential Grouping for Large-Scale Global Optimization. IEEE Transactions on Evolutionary Computation. 2022; pp 1439–1451.

[38] Li, J.-Y.; Zhan, Z.-H.; Tan, K. C.; Zhang, J. Dual Differential Grouping: A More General Decomposition Method for Large-Scale Optimization. IEEE Transactions on Cybernetics. 2022; pp 1–15.

[39] Luenberger, D.; Ye, Y. *Linear and Nonlinear Programming*; Springer New York, NY, 2021.

[40] Qi, L.; Sun, J. A nonsmooth version of Newton's method. Math. Program. 1993; pp 353–367.

[41] Dai, Y.-H.; Yuan, Y.-x. A Nonlinear Conjugate Gradient Method With A Strong Global Convergence Property. SIAM Journal on Optimization. 1999.

[42] ichi Amari, S. Backpropagation and stochastic gradient descent method. Neurocomputing. 1993; pp 185–196.

[43] Gao, S.; Yu, Y.; Wang, Y.; Wang, J.; Cheng, J.; Zhou, M. Chaotic Local Search-Based Differential Evolution Algorithms for Optimization. IEEE Transactions on Systems, Man, and Cybernetics: Systems. 2021; pp 3954–3967.

[44] Gao, K.; Cao, Z.; Zhang, L.; Chen, Z.; Han, Y.; Pan, Q. A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems. IEEE/CAA Journal of Automatica Sinica. 2019; pp 904–916.

[45] Lu, Z.; Whalen, I.; Boddeti, V.; Dhebar, Y.; Deb, K.; Goodman, E.; Banzhaf, W. NSGA-Net: Neural Architecture Search Using Multi-Objective Genetic Algorithm. Proceedings of the Genetic and Evolutionary Computation Conference. New York, NY, USA, 2019; p 419–427.

[46] Ahmad, M.; Abdullah, M.; Moon, H.; Yoo, S. J.; Han, D. Image Classification Based on Automatic Neural Architecture Search Using Binary Crow Search Algorithm. IEEE Access. 2020; pp 189891–189912.

[47] Abdel-Basset, M.; Abdel-Fatah, L.; Sangaiah, A. K. In *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*; Sangaiah, A. K., Sheng, M., Zhang, Z., Eds.; Intelligent Data-Centric Systems; Academic Press, 2018; pp 185–231.

[48] De Jong, K. Learning with Genetic Algorithms: An Overview. Machine Learning. 1988; pp 121–138.

[49] Storn, R. On the usage of differential evolution for function optimization. Proceedings of North American Fuzzy Information Processing. 1996; pp 519–523.

[50] Koza, J. R. Genetic programming as a means for programming computers by natural selection. Statistics and computing. 1994; pp 87–112.

[51] Yao, X.; Liu, Y.; Lin, G. Evolutionary programming made faster. IEEE Transactions on Evolutionary Computation. 1999; pp 82–102.

[52] Kennedy, J.; Eberhart, R. Particle swarm optimization. Proceedings of ICNN'95 - International Conference on Neural Networks. 1995; pp 1942–1948 vol.4.

[53] Mirjalili, S.; Gandomi, A. H.; Mirjalili, S. Z.; Saremi, S.; Faris, H.; Mirjalili, S. M. Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. Advances in Engineering Software. 2017; pp 163–191.

[54] Zaldívar, D.; Morales, B.; Rodríguez, A.; Valdivia-G, A.; Cuevas, E.; Pérez-Cisneros, M. A novel bio-inspired optimization model based on Yellow Saddle Goatfish behavior. Biosystems. 2018; pp 1–21.

[55] Shadravan, S.; Naji, H.; Bardsiri, V. The Sailfish Optimizer: A novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems. Engineering Applications of Artificial Intelligence. 2019; pp 20–34.

[56] Alsattar, H.; Zaidan, A.; Bahaa, B. Novel meta-heuristic bald eagle search optimisation algorithm. Artificial Intelligence Review. 2020.

[57] Faramarzi, A.; Heidarinejad, M.; Mirjalili, S.; Gandomi, A. H. Marine Predators Algorithm: A nature-inspired metaheuristic. Expert Systems with Applications. 2020; p 113377.

[58] Hayyolalam, V.; Pourhaji Kazem, A. A. Black Widow Optimization Algorithm: A novel meta-heuristic approach for solving engineering optimization problems. Engineering Applications of Artificial Intelligence. 2020; p 103249.

[59] Fathollahi-Fard, A.; Hajiaghaei-Keshteli, M.; Tavakkoli-Moghaddam, R. Red deer algorithm (RDA): a new nature-inspired meta-heuristic. Soft Computing. 2020.

[60] Braik, M.; Sheta, A.; Al-Hiary, H. A Novel Meta-Heuristic Search Algorithm for Solving Optimization Problems: Capuchin Search Algorithm. Neural Comput. Appl. Berlin, Heidelberg, 2021; p 2515–2547.

[61] Xie, L.; Han, T.; Zhou, H.; Zhang, Z.-R.; Han, B.; Tang, A.; Khalil, A. M. Tuna Swarm Optimization: A Novel Swarm-Based Metaheuristic Algorithm for Global Optimization. Intell. Neuroscience. London, GBR, 2021.

[62] Hashim, F. A.; Hussien, A. G. Snake Optimizer: A Novel Meta-Heuristic Optimization Algorithm. Know.-Based Syst. NLD, 2022.

[63] Yu, J. Vegetation Evolution: An Optimization Algorithm Inspired by the Life Cycle of Plants. International Journal of Computational Intelligence and Applications. 2022.

[64] Azizi, M.; Talatahari, S.; Gandomi, A. Fire Hawk Optimizer: a novel metaheuristic algorithm. Artificial Intelligence Review. 2022; pp 1–77.

[65] Hashim, F. A.; Houssein, E. H.; Hussain, K.; Mabrouk, M. S.; Al-Atabany, W. Honey Badger Algorithm: New metaheuristic algorithm for solving optimization problems. Mathematics and Computers in Simulation. 2022; pp 84–110.

[66] Seyyedabbasi, A.; Kiani, F. Sand Cat swarm optimization: a nature-inspired algorithm to solve global optimization problems. Engineering with Computers. 2022; pp 1–25.

[67] Dehghani, M.; Montazeri, Z.; Trojovská, E.; Trojovský, P. Coati Optimization Algorithm: A new bio-inspired metaheuristic algorithm for solving optimization problems. Knowledge-Based Systems. 2023; p 110011.

[68] Jia, H.; Rao, H.; Wen, C.; Mirjalili, S. Crayfish optimization algorithm. Artificial Intelligence Review. 2023; p 1.

[69] Shi, Y. Brain Storm Optimization Algorithm. Advances in Swarm Intelligence. Berlin, Heidelberg, 2011; pp 303–309.

[70] Rao, R.; Savsani, V.; Vakharia, D. Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. Computer-Aided Design. 2011; pp 303–315.

[71] Bogar, E.; Beyhan, S. Adolescent Identity Search Algorithm (AISA): A novel metaheuristic approach for solving optimization problems. Applied Soft Computing. 2020; p 106503.

[72] Askari, Q.; Younas, I.; Saeed, M. Political Optimizer: A novel socio-inspired metaheuristic for global optimization. Knowledge-Based Systems. 2020; p 105709.

[73] Zhang, Y.; Jin, Z. Group teaching optimization algorithm: A novel metaheuristic method for solving global optimization problems. Expert Systems with Applications. 2020; p 113246.

[74] Shabani, A.; Asgarian, B.; Salido, M.; Asil Gharebaghi, S. Search and rescue optimization algorithm: A new optimization method for solving constrained engineering optimization problems. Expert Systems with Applications. 2020; p 113698.

[75] kai Feng, Z.; jing Niu, W.; Liu, S. Cooperation search algorithm: A novel metaheuristic evolutionary intelligence algorithm for numerical optimization and engineering optimization problems. Applied Soft Computing. 2021; p 106734.

[76] Abdulhameed, S.; Rashid, T. Child Drawing Development Optimization Algorithm Based on Child's Cognitive Development. ARABIAN JOURNAL FOR SCIENCE AND ENGINEERING. 2021.

[77] Acharya, D.; Das, D. A novel Human Conception Optimizer for solving optimization problems. Scientific Reports. 2022.

[78] Dehghani, M.; Trojovska, E.; Zuščák, T. A new human-inspired metaheuristic algorithm for solving optimization problems based on mimicking sewing training. Scientific Reports. 2022.

[79] Faridmehr, I.; Nehdi, M. L.; Davoudkhani, I. F.; Poolad, A. Mountaineering Team-Based Optimization: A Novel Human-Based Metaheuristic Algorithm. Mathematics. 2023.

[80] Matoušová, I.; Trojovsky, P.; Dehghani, M.; Trojovska, E.; Kostra, J. Mother optimization algorithm: a new human-based metaheuristic approach for solving engineering optimization. Scientific Reports. 2023.

[81] Wei, Z.; Huang, C.; Wang, X.; Han, T.; Li, Y. Nuclear Reaction Optimization: A Novel and Powerful Physics-Based Algorithm for Global Optimization. IEEE Access. 2019; pp 66084–66109.

[82] Houssein, E. H.; Saad, M. R.; Hashim, F. A.; Shaban, H.; Hassaballah, M. Lévy flight distribution: A new metaheuristic algorithm for solving engineering optimization problems. Engineering Applications of Artificial Intelligence. 2020; p 103731.

[83] Ahmadianfar, I.; Bozorg-Haddad, O.; Chu, X. Gradient-based optimizer: A new metaheuristic optimization algorithm. Information Sciences. 2020; pp 131–159.

[84] Dehghani, M.; Samet, H. Momentum search algorithm: a new meta-heuristic opti-

mization algorithm inspired by momentum conservation law. SN Applied Sciences. 2020.

[85] Talatahari, S.; Azizi, M.; Gandomi, A. H. Material Generation Algorithm: A Novel Metaheuristic Algorithm for Optimization of Engineering Problems. Processes. 2021.

[86] Ahmadianfar, I.; Heidari, A. A.; Gandomi, A. H.; Chu, X.; Chen, H. RUN beyond the metaphor: An efficient optimization algorithm based on Runge Kutta method. Expert Systems with Applications. 2021; p 115079.

[87] Shaqfa, M.; Beyer, K. Pareto-like Sequential Sampling Heuristic for Global Optimisation. Soft Computing. 2021; p 9077–9096.

[88] Ahmadianfar, I.; Heidari, A. A.; Noshadian, S.; Chen, H.; Gandomi, A. H. INFO: An efficient optimization algorithm based on weighted mean of vectors. Expert Systems with Applications. 2022; p 116516.

[89] Goodarzimehr, V.; Shojaee, S.; Hamzehei-Javaran, S.; Talatahari, S. Special Relativity Search: A novel metaheuristic method based on special relativity physics. Knowledge-Based Systems. 2022; p 109484.

[90] Deng, L.; Liu, S. Snow ablation optimizer: A novel metaheuristic technique for numerical optimization and engineering design. Expert Systems with Applications. 2023; p 120069.

[91] Cheng, M.-Y.; Sholeh, M. N. Optical microscope algorithm: A new metaheuristic inspired by microscope magnification for solving engineering optimization problems. Knowledge-Based Systems. 2023; p 110939.

[92] Shehadeh, H. Chernobyl disaster optimizer (CDO): a novel meta-heuristic method for global optimization. Neural Computing and Applications. 2023.

[93] Zhong, R.; Zhang, E.; Munetomo, M. *Complex & Intelligent System* **2023**, *9*, 4439–4456.

[94] Chen, W.; Weise, T.; Yang, Z.; Tang, K. Large-Scale Global Optimization Using Cooperative Coevolution with Variable Interaction Learning. Proc. of International Conference on Parallel Problem Solving from Nature, Ser. Lecture Notes in Computer Science. 2010; pp 300–309.

[95] Mei, Y.; Li, X.; Yao, X. Cooperative Coevolution With Route Distance Grouping for Large-Scale Capacitated Arc Routing Problems. IEEE Transactions on Evolutionary Computation. 2014; pp 435–449.

[96] Sayed, E.; Essam, D.; Sarker, R.; Elsayed, S. Decomposition-based evolutionary algorithm for large scale constrained problems. Information Sciences. 2015; pp 457–486, Nature-Inspired Algorithms for Large Scale Global Optimization.

[97] Omidvar, M. N.; Li, X.; Yao, X. A review of population-based metaheuristics for large-scale black-box global optimization: Part A. IEEE Transactions on Evolutionary Computation. 2021; pp 1–1.

[98] Omidvar, Mohammad Nabi and Li, Xiaodong and Yao, Xin A review of population-based metaheuristics for large-scale black-box global optimization: Part B. IEEE Transactions on Evolutionary Computation. 2021; pp 1–1.

[99] Sun, Y.; Kirley, M.; Halgamuge, S. K. Extended Differential Grouping for Large Scale Global Optimization with Direct and Indirect Variable Interactions. New York, NY, USA, 2015; p 313–320.

[100] Wu, Y.; Peng, X.; Wang, H.; Jin, Y.; Xu, D. Cooperative Coevolutionary CMA-ES with Landscape-Aware Grouping in Noisy Environments. IEEE Transactions on Evolutionary Computation. 2022; pp 1–1.

[101] Zhong, R.; Tu, B.; Zhang, E.; Munetomo, M. *Evolutionary Intelligence* **2024**, 1–21.

[102] Sun, Y.; Kirley, M.; Halgamuge, S. K. Extended Differential Grouping for Large Scale Global Optimization with Direct and Indirect Variable Interactions. Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation. New York, NY, USA, 2015; p 313–320.

[103] Munetomo, M. Linkage identification with epistasis measures considering monotonicity conditions. 2002.

[104] De Jong, K. A. An Analysis of the Behavior of a Class of Genetic Adaptive Systems. Ph.D. thesis, University of Michigan, USA, 1975; AAI7609381.

[105] Ghosh, A.; Das, S.; Mallipeddi, R.; Das, A. K.; Dash, S. S. A Modified Differential Evolution With Distance-based Selection for Continuous Optimization in Presence of Noise. IEEE Access. 2017; pp 26944–26964.

[106] Ghosh, A.; Das, S.; Mullick, S. S.; Mallipeddi, R.; Das, A. K. A switched parameter differential evolution with optional blending crossover for scalable numerical optimization. Applied Soft Computing. 2017; pp 329–352.

[107] Kundu, R.; Mukherjee, R.; Das, S.; Vasilakos, A. V. Adaptive differential evolution with difference mean based perturbation for dynamic economic dispatch problem. 2013 IEEE Symposium on Differential Evolution (SDE). 2013; pp 38–45.

[108] Omidvar, M.; Li, X.; Yao, X. Cooperative co-evolution with delta grouping for large scale non-separable function optimization. 2010 IEEE World Congress on Computational Intelligence, WCCI 2010 - 2010 IEEE Congress on Evolutionary Computation, CEC 2010. 2010; pp 1–8.

[109] Wolpert, D.; Macready, W. No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation. 1997; pp 67–82.

[110] Zhong, R.; Zhang, E.; Munetomo, M. *Complex & Intelligent System* **2024**, *10*, 2129–2149.

[111] Sun, Y.; Omidvar, M. N.; Kirley, M.; Li, X. Adaptive Threshold Parameter Estimation with Recursive Differential Grouping for Problem Decomposition. Proceedings of the Genetic and Evolutionary Computation Conference. New York, NY, USA, 2018; p 889–896.

[112] Jin, Y.; Sendhoff, B. A systems approach to evolutionary multiobjective structural optimization and beyond. IEEE Computational Intelligence Magazine. 2009; pp 62–76.

[113] Rashidi, S.; Ranjitkar, P. Bus Dwell Time Modeling Using Gene Expression Programming. Computer-Aided Civil and Infrastructure Engineering. 2015.

[114] Bidgoli, H., Ed. *Encyclopedia of Information Systems*; Elsevier: New York, 2003; pp 259–267.

[115] Chugh, T.; Sindhya, K.; Hakanen, J.; Miettinen, K. A Survey on Handling Computationally Expensive Multiobjective Optimization Problems with Evolutionary Algorithms. Soft Comput. Berlin, Heidelberg, 2019; p 3137–3166.

[116] Jin, Y. Surrogate-assisted evolutionary computation: Recent advances and future challenges. Swarm and Evolutionary Computation. 2011; pp 61–70.

[117] Amouzgar, K.; Bandaru, S.; Ng, A. H. Radial basis functions with a priori bias as surrogate models:A comparative study. Engineering Applications of Artificial Intelligence. 2018; pp 28–44.

[118] Zhou, Z.; Ong, Y. S.; Nguyen, M. H.; Lim, D. A study on polynomial regression and Gaussian process global surrogate model in hierarchical surrogate-assisted evo-

lutionary algorithm. 2005 IEEE Congress on Evolutionary Computation. 2005; pp 2832–2839 Vol. 3.

[119] Liu, B.; Zhang, Q.; Gielen, G. G. E. A Gaussian Process Surrogate Model Assisted Evolutionary Algorithm for Medium Scale Expensive Optimization Problems. IEEE Transactions on Evolutionary Computation. 2014; pp 180–192.

[120] Song, Z.; Wang, H.; He, C.; Jin, Y. A Kriging-Assisted Two-Archive Evolutionary Algorithm for Expensive Many-Objective Optimization. IEEE Transactions on Evolutionary Computation. 2021; pp 1013–1027.

[121] Atashkari, K.; Nariman-Zadeh, N.; Gölcü, M.; Khalkhali, A.; Jamali, A. Modelling and multi-objective optimization of a variable valve-timing spark-ignition engine using polynomial neural networks and evolutionary algorithms. Energy Conversion and Management. 2007; pp 1029–1041.

[122] Li, F.; Cai, X.; Gao, L.; Shen, W. A Surrogate-Assisted Multiswarm Optimization Algorithm for High-Dimensional Computationally Expensive Problems. IEEE Transactions on Cybernetics. 2021; pp 1390–1402.

[123] Cai, X.; Qiu, H.; Gao, L.; Jiang, C.; Shao, X. An efficient surrogate-assisted particle swarm optimization algorithm for high-dimensional expensive problems. Knowledge-Based Systems. 2019; p 104901.

[124] Cai, X.; Gao, L.; Li, X. Efficient Generalized Surrogate-Assisted Evolutionary Algorithm for High-Dimensional Expensive Problems. IEEE Transactions on Evolutionary Computation. 2020; pp 365–379.

[125] De Falco, I.; Della Cioppa, A.; Trunfio, G. A. Investigating surrogate-assisted cooperative coevolution for large-Scale global optimization. Information Sciences. 2019; pp 1–26.

[126] Ren, Z.; Pang, B.; Wang, M.; Feng, Z.; Liang, Y.; Chen, A.; Zhang, Y. Surrogate Model Assisted Cooperative Coevolution for Large Scale Optimization. Applied Intelligence. USA, 2019; p 513–531.

[127] Tian, Y.; Si, L.; Zhang, X.; Cheng, R.; He, C.; Tan, K. C.; Jin, Y. Evolutionary Large-Scale Multi-Objective Optimization: A Survey. ACM Comput. Surv. New York, NY, USA, 2021.

[128] Specht, D. A general regression neural network. IEEE Transactions on Neural Networks. 1991; pp 568–576.

[129] Hornik, K. Approximation capabilities of multilayer feedforward networks. Neural Networks. 1991; pp 251–257.

[130] 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. 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. 2016; pp 1–13.

[131] Leung, M.; Chen, A.-S.; Mancha, R. Making trading decisions for financial engineered derivatives: A novel ensemble of neural networks using information content. Int. Syst. in Accounting, Finance and Management. 2009; pp 257–277.

[132] Cheng, R.; Jin, Y.; Narukawa, K.; Sendhoff, B. A Multiobjective Evolutionary Algorithm Using Gaussian Process-Based Inverse Modeling. IEEE Transactions on Evolutionary Computation. 2015; pp 838–856.

[133] Buche, D.; Schraudolph, N.; Koumoutsakos, P. Accelerating evolutionary algorithms with Gaussian process fitness function models. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews). 2005; pp 183–194.

[134] Zhang, Q.; Liu, W.; Tsang, E.; Virginas, B. Expensive Multiobjective Optimization by MOEA/D With Gaussian Process Model. IEEE Transactions on Evolutionary Computation. 2010; pp 456–474.

[135] Yang, D.; Zhang, X.; Pan, R.; Wang, Y.; Chen, Z. A novel Gaussian process regression model for state-of-health estimation of lithium-ion battery using charging curve. Journal of Power Sources. 2018; pp 387–395.

[136] Liu, D.; Pang, J.; Zhou, J.; Peng, Y.; Pecht, M. Prognostics for state of health estimation of lithium-ion batteries based on combination Gaussian process functional regression. Microelectronics Reliability. 2013; pp 832–839.

[137] Rasmussen, C. E.; Nickisch, H. Gaussian Processes for Machine Learning (GPML) Toolbox. J. Mach. Learn. Res. 2010; p 3011–3015.

[138] Sun, Y.; Li, X.; Ernst, A.; Omidvar, M. N. Decomposition for Large-scale Optimization Problems with Overlapping Components. 2019 IEEE Congress on Evolutionary Computation (CEC). 2019; pp 326–333.

[139] Hu, X.-M.; He, F.-L.; Chen, W.-N.; Zhang, J. Cooperation coevolution with fast interdependency identification for large scale optimization. Information Sciences. 2017; pp 142–160.

[140] IEEE Standard for Floating-Point Arithmetic. IEEE Std 754-2019 (Revision of IEEE 754-2008). 2019; pp 1–84.

[141] Dushatskiy, A.; Mendrik, A. M.; Alderliesten, T.; Bosman, P. A. N. Convolutional Neural Network Surrogate-Assisted GOMEA. Proceedings of the Genetic and Evolutionary Computation Conference. New York, NY, USA, 2019; p 753–761.

[142] Wang, Y.; Yin, D.-Q.; Yang, S.; Sun, G. Global and Local Surrogate-Assisted Differ-

ential Evolution for Expensive Constrained Optimization Problems With Inequality Constraints. IEEE Transactions on Cybernetics. 2019; pp 1642–1656.

[143] Sarkari Khorrami, M.; Mianroodi, J.; Siboni, N.; Goyal, P.; Svendsen, B.; Benner, P.; Raabe, D. An artificial neural network for surrogate modeling of stress fields in viscoplastic polycrystalline materials. npj Comput Mater. 2023.

[144] Joy, E. J.; Menon, A. S.; Biju, N. Implementation of Kriging Surrogate Models for Delamination Detection in Composite Structures. Advanced Composites Letters. 2018; p 096369351802700604.

[145] Wang, Y.; Lin, J.; Liu, J.; Sun, G.; Pang, T. Surrogate-Assisted Differential Evolution With Region Division for Expensive Optimization Problems With Discontinuous Responses. IEEE Transactions on Evolutionary Computation. 2022; pp 780–792.

[146] Ren, C.; Aoues, Y.; Lemosse, D.; Souza De Cursi, E. Ensemble of surrogates combining Kriging and Artificial Neural Networks for reliability analysis with local goodness measurement. Structural Safety. 2022; p 102186.

[147] Santos, L. F.; Costa, C. B.; Caballero, J. A.; Ravagnani, M. A. Multi-objective simulation–optimization via kriging surrogate models applied to natural gas liquefaction process design. Energy. 2023; p 125271.

[148] Chen, C.; Liu, J.; Xu, P. Comparison of parallel infill sampling criteria based on Kriging surrogate model. Scientific Reports. 2022.

[149] Amato, F. pyGRNN. `https://github.com/federhub/pyGRNN`.

[150] Pedregosa, F. et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011; pp 2825–2830.

[151] Zhong, R.; Peng, F.; Zhang, E.; Yu, J.; Munetomo, M. *Biomimetics* **2023**, *8*.

[152] Zheng, S.; Janecek, A.; Tan, Y. Enhanced Fireworks Algorithm. 2013 IEEE Congress on Evolutionary Computation. 2013; pp 2069–2077.

[153] YU, J.; TAKAGI, H. Accelerating Vegetation Evolution with Mutation Strategy and Gbased Growth Strategy. 2019 IEEE Symposium Series on Computational Intelligence (SSCI). 2019; pp 3033–3039.

[154] YU, J.; TAKAGI, H. Multi-species Generation Strategy-Based Vegetation Evolution. 2020 IEEE Congress on Evolutionary Computation (CEC). 2020; pp 1–6.

[155] Yu, J.; Takagi, H. Performance Analysis of Vegetation Evolution. 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC). 2019; pp 2214–2219.

[156] Yaguchi, K.; Tamura, K.; Yasuda, K.; Ishigame, A. Basic study of proximate optimality principle based combinatorial optimization method. 2011 IEEE International Conference on Systems, Man, and Cybernetics. 2011; pp 1753–1758.

[157] Holland, J. H. Outline for a Logical Theory of Adaptive Systems. J. ACM. New York, NY, USA, 1962; p 297–314.

[158] Katoch, S.; Chauhan, S. S.; Kumar, V. A review on genetic algorithm: past, present, and future. Multimedia Tools and Applications. 2021; pp 8091–8126.

[159] Yang, X. S. Flower Pollination Algorithm for Global Optimization. Unconventional Computation and Natural Computation. Berlin, Heidelberg, 2012; pp 240–249.

[160] Zhao, S.; Zhang, T.; Ma, S.; Chen, M. Dandelion Optimizer: A Nature-Inspired Metaheuristic Algorithm for Engineering Applications. Eng. Appl. Artif. Intell. USA, 2022.

[161] Zhong, R.; Peng, F.; Yu, J.; Munetomo, M. *Alexandria Engineering Journal* **2024**, *87*, 148–163.

[162] Zhong, R.; Zhang, C.; Yu, J. *Evolutionary Intelligence* **2024**, 1–25.

[163] Liang, J.; Qu, B.; Suganthan, P.; Hernández-Díaz, A. Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization. Technical Report 201212, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China. 2013.

[164] Thieu, N. V. ENOPPY: A Python Library for Engineering Optimization Problems. 2023.

[165] Coello Coello, C. A. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Computer Methods in Applied Mechanics and Engineering. 2002; pp 1245–1287.

[166] Van Thieu, N.; Mirjalili, S. MEALPY: An open-source library for latest meta-heuristic algorithms in Python. Journal of Systems Architecture. 2023; p 102871.

[167] Teo, J. Exploring dynamic self-adaptive populations in differential evolution. Soft Comput. 2006; pp 673–686.

[168] Ghasemi, M.; Akbari, E.; Rahimnejad, A.; Razavi, E.; Ghavidel, S.; Li, L. Phasor particle swarm optimization: a simple and efficient variant of PSO. Soft Computing. 2019.

[169] Tharwat, A.; Gabel, T. Parameters optimization of support vector machines for imbalanced data using social ski driver algorithm. Neural Computing and Applications. 2020.

[170] Zhong, R.; Yu, J.; Zhang, C.; Munetomo, M. *Neural Computing and Applications* **2024**, *36*, 6721–6740.

[171] Stein, M. Large Sample Properties of Simulations Using Latin Hypercube Sampling. Technometrics. 1987; pp 143–151.

[172] Bouamama, S.; Jlifi, B.; Ghedira, K. D2G2A: A distributed double guided genetic algorithm for Max_CSPs. Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science). 2003; pp 422–429.

[173] Nguyen, T. A framework of Optimization Functions using Numpy (OpFuNu) for optimization problems. 2020.

[174] C. T. Yue, P. N. S., K. V. Price Problem Definitions and Evaluation Criteria for the CEC 2020 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization. Technical Report, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore. 2020.

[175] Ezugwu, A.; Agushaka, O.; Abualigah, L.; Mirjalili, S.; Gandomi, A. Prairie Dog Optimization Algorithm. Neural Computing and Applications. 2022; p 20017–20065.

[176] Hansen, N.; Ostermeier, A. Completely Derandomized Self-Adaptation in Evolution Strategies. Evolutionary Computation. 2001; pp 159–195.

[177] Mirjalili, S.; Mirjalili, S. M.; Lewis, A. Grey Wolf Optimizer. Advances in Engineering Software. 2014; pp 46–61.

[178] Zhao, W.; Wang, L.; Zhang, Z. Artificial ecosystem-based optimization: a novel nature-inspired meta-heuristic algorithm. Neural Computing and Applications. 2020; pp 1–43.

[179] Trojovská, E.; Dehghani, M.; Trojovský, P. Zebra Optimization Algorithm: A New Bio-Inspired Optimization Algorithm for Solving Optimization Algorithm. IEEE Access. 2022; pp 49445–49473.

[180] Ayyarao, T. S. L. V.; Ramakrishna, N. S. S.; Elavarasan, R. M.; Polumahanthi, N.; Rambabu, M.; Saini, G.; Khan, B.; Alatas, B. War Strategy Optimization Algorithm: A New Effective Metaheuristic Algorithm for Global Optimization. IEEE Access. 2022; pp 25073–25105.

[181] Azizi, M.; Aickelin, U.; Khorshidi, H.; Baghalzadeh Shishehgarkhaneh, M. Energy valley optimizer: a novel metaheuristic algorithm for global and engineering optimization. Scientific Reports. 2023; p 226.

[182] Holm, S. A Simple Sequentially Rejective Multiple Test Procedure. Scandinavian Journal of Statistics. 1979; pp 65–70.

[183] Zhong, R.; Zhang, E.; Munetomo, M. *J. Supercomput.* **2024**, *80*, 12186–12217.

[184] T., N.; H., Y.; Toth, A. Maze-solving by an amoeboid organism. Nature. 2000; p 470.

[185] Tero, A.; Takagi, S.; Saigusa, T.; Ito, K.; Bebber, D. P.; Fricker, M. D.; Yumiki, K.; Kobayashi, R.; Nakagaki, T. Rules for Biologically Inspired Adaptive Network Design. Science. 2010; pp 439–442.

[186] Adamatzky, A.; Akl, S.; Alonso-Sanz, R.; van Dessel, W.; Ibrahim, Z.; Ilachinski, A.; Jones, J.; Kayem, A. V.; Martínez, G. J.; de Oliveira, P.; Prokopenko, M.; Schubert, T.; Sloot, P.; Strano, E.; Yang, X.-S. Are motorways rational from slime mould's point of view? International Journal of Parallel, Emergent and Distributed Systems. 2013; pp 230–248.

[187] Boussard, A.; Fessel, A.; Oettmeier, C.; Briard, L.; Döbereiner, H.-G.; Dussutour, A. Adaptive behaviour and learning in slime moulds: the role of oscillations. Philosophical Transactions of the Royal Society B: Biological Sciences. 2021; p 20190757.

[188] Heidari, A. A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H. Harris

hawks optimization: Algorithm and applications. Future Generation Computer Systems. 2019; pp 849–872.

[189] Li, S.; Chen, H.; Wang, M.; Heidari, A. A.; Mirjalili, S. Slime mould algorithm: A new method for stochastic optimization. Future Generation Computer Systems. 2020; pp 300–323.

[190] Jia, H.; Peng, X.; Lang, C. Remora optimization algorithm. Expert Systems with Applications. 2021; p 115665.

[191] Chopra, N.; Mohsin Ansari, M. Golden jackal optimization: A novel nature-inspired optimizer for engineering applications. Expert Systems with Applications. 2022; p 116924.

[192] Nakagaki, T.; Yamada, H.; Ueda, T. Interaction between cell shape and contraction pattern in the Physarum plasmodium. Biophysical chemistry. 2000; pp 195–204.

[193] Lipowski, A.; Lipowska, D. Roulette-wheel selection via stochastic acceptance. Physica A: Statistical Mechanics and its Applications. 2012; pp 2193–2196.

[194] Arora, J. S., Ed. *Introduction to Optimum Design (Fourth Edition)*, fourth edition ed.; Academic Press: Boston, 2017; p iii.

[195] Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. Advances in Engineering Software. 2016; pp 51–67.

[196] Bayzidi, H.; Talatahari, S.; Saraee, M.; Lamarche, C.-P. Social Network Search for Solving Engineering Optimization Problems. Computational Intelligence and Neuroscience. 2021; pp 1–32.

[197] Abualigah, L.; Diabat, A.; Mirjalili, S.; Abd Elaziz, M.; Gandomi, A. H. The Arithmetic Optimization Algorithm. Computer Methods in Applied Mechanics and Engineering. 2021; p 113609.

[198] Cowling, P.; Kendall, G.; Soubeiga, E. A Hyperheuristic Approach to Scheduling a Sales Summit. Practice and Theory of Automated Timetabling III. Berlin, Heidelberg, 2001; pp 176–190.

[199] Jackson, W. G.; Özcan, E.; Drake, J. H. Late acceptance-based selection hyper-heuristics for cross-domain heuristic search. 2013 13th UK Workshop on Computational Intelligence (UKCI). 2013; pp 228–235.

[200] Özcan, E.; Kheiri, A. A Hyper-Heuristic Based on Random Gradient, Greedy and Dominance. Computer and Information Sciences II. London, 2012; pp 557–563.

[201] Cowling, P.; Kendall, G.; Soubeiga, E. Hyperheuristics: A Tool for Rapid Prototyping in Scheduling and Optimisation. Applications of Evolutionary Computing. Berlin, Heidelberg, 2002; pp 1–10.

[202] Zhong, R.; Yu, J.; Zhang, C.; Munetomo, M. *International Journal of Computational Intelligence Systems* **2023**, *16*, 169.

[203] Al-Sahaf, H.; Bi, Y.; Chen, Q.; Lensen, A.; Mei, Y.; Sun, Y.; Tran, B.; Xue, B.; Zhang, M. A survey on evolutionary machine learning. Journal of the Royal Society of New Zealand. 2019; pp 205–228.

[204] Wang, Z.; Sobey, A. A comparative review between Genetic Algorithm use in composite optimisation and the state-of-the-art in evolutionary computation. Composite Structures. 2020; p 111739.

[205] Tan, K. C.; Feng, L.; Jiang, M. Evolutionary Transfer Optimization - A New Frontier in Evolutionary Computation Research. IEEE Computational Intelligence Magazine. 2021; pp 22–33.

[206] Fernandes Junior, F. E.; Yen, G. G. Particle swarm optimization of deep neural net-

works architectures for image classification. Swarm and Evolutionary Computation. 2019; pp 62–74.

[207] Telikani, A.; Gandomi, A. H.; Shahbahrami, A. A survey of evolutionary computation for association rule mining. Information Sciences. 2020; pp 318–352.

[208] Zhao, F.; He, X.; Wang, L. A Two-Stage Cooperative Evolutionary Algorithm With Problem-Specific Knowledge for Energy-Efficient Scheduling of No-Wait Flow-Shop Problem. IEEE Transactions on Cybernetics. 2021; pp 5291–5303.

[209] Chatterjee, T.; Chakraborty, S.; Chowdhury, R. A critical review of surrogate assisted robust design optimization. Archives of Computational Methods in Engineering. 2019; pp 245–274.

[210] Dong, H.; Dong, Z. Surrogate-assisted grey wolf optimization for high-dimensional, computationally expensive black-box problems. Swarm and Evolutionary Computation. 2020; p 100713.

[211] Nishihara, K.; Nakata, M. Surrogate-assisted Differential Evolution with Adaptation of Training Data Selection Criterion. 2022 IEEE Symposium Series on Computational Intelligence (SSCI). 2022; pp 1675–1682.

[212] Wang, W.; Liu, H.-L.; Tan, K. C. A Surrogate-Assisted Differential Evolution Algorithm for High-Dimensional Expensive Optimization Problems. IEEE Transactions on Cybernetics. 2023; pp 2685–2697.

[213] Cai, X.; Ruan, G.; Yuan, B.; Gao, L. Complementary surrogate-assisted differential evolution algorithm for expensive multi-objective problems under a limited computational budget. Information Sciences. 2023; pp 791–814.

[214] Dowsland, K. A. Off-the-peg or made-to-measure? timetabling and scheduling with

SA and TS. Practice and Theory of Automated Timetabling II. Berlin, Heidelberg, 1998; pp 37–52.

[215] Kheiri, A.; Keedwell, E. Selection Hyper-Heuristics. Proceedings of the Genetic and Evolutionary Computation Conference Companion. New York, NY, USA, 2022; p 983–996.

[216] Cruz-Duarte, J. M.; Amaya, I.; Ortiz-Bayliss, J. C.; Conant-Pablos, S. E.; Terashima-Marín, H. A Primary Study on Hyper-Heuristics to Customise Metaheuristics for Continuous optimisation. 2020 IEEE Congress on Evolutionary Computation (CEC). 2020; pp 1–8.

[217] Braik, M. S. Chameleon Swarm Algorithm: A bio-inspired optimizer for solving engineering design problems. Expert Systems with Applications. 2021; p 114685.

[218] Abdollahzadeh, B.; Soleimanian Gharehchopogh, F.; Mirjalili, S. Artificial gorilla troops optimizer: A new nature-inspired metaheuristic algorithm for global optimization problems. International Journal of Intelligent Systems. 2021.

[219] CHICKERMANE, H.; GEA, H. C. STRUCTURAL OPTIMIZATION USING A NEW LOCAL APPROXIMATION METHOD. International Journal for Numerical Methods in Engineering. 1996; pp 829–846.

[220] Arora, J. S. In *Introduction to Optimum Design (Fourth Edition)*, fourth edition ed.; Arora, J. S., Ed.; Academic Press: Boston, 2017; p iv.

[221] Zhong, X.; Cheng, P. An Improved Differential Evolution Algorithm Based on Dual-Strategy. Mathematical Problems in Engineering. 2020; pp 1–14.

[222] Wu, Z.; Chow, T. W. Neighborhood Field for Cooperative Optimization. Soft Comput. Berlin, Heidelberg, 2013; p 819–834.

[223] Liu, Y.; Wang, H. Surrogate-assisted hybrid evolutionary algorithm with local estimation of distribution for expensive mixed-variable optimization problems. Applied Soft Computing. 2023; p 109957.

[224] Köppen, M. The curse of dimensionality. 5th online world conference on soft computing in industrial applications (WSC5). 2000; pp 4–8.