Swarm and Evolutionary Computation 98 (2025) 102100

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

Dimensional window method: A plug-in-style large-scale handling technique for evolutionary algorithm[☆]

Yafeng Sun [a], Xingwang Wang [a,b,*], Junhong Huang [a], Bo Sun [c], Peng Liang [d]

[a] College of Computer Science and Technology, Jilin University, Changchun 130012, China
[b] Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China
[c] School of Intelligent Systems Engineering, Shenzhen Campus of Sun Yat-sen University, Shenzhen 518000, China
[d] School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan 430074, China

ARTICLE INFO

Keywords:
Large-scale optimization
Evolutionary algorithm
Dimensional window method
Neural network
Network pruning

ABSTRACT

Large-scale optimization constitutes a pivotal characteristic of numerous real-world problems, where large-scale evolutionary algorithms emerge as a potent instrument for addressing such intricacies. However, existing methods are typically tailored to address only a particular class of problems and lack the versatility to be readily adapted to other evolutionary algorithms or generalized across diverse problem domains. To address the issue above, this paper proposes the dimensional window method, a simple yet effective enhancement that can be seamlessly integrated into low-dimensional evolutionary algorithms to bolster their performance in large-scale optimization. Specifically, the dimensional window method involves grouping a subset of randomly selected dimensions into a window during each iteration, restricting the population's evolution to the dimensions within this window. Furthermore, the effectiveness of the dimensional window method is analyzed, and the window is improved based on the insights gained, including the isometric segmentation individual-level window length and the neural network-guided window element. Extensive experiments on single-objective, multi-objective, constrained multi-objective, and discrete test problems with large-scale attributes demonstrate that the proposed method significantly mitigates the curse of dimensionality and enhances the performance of evolutionary algorithms in large-scale settings. A more significant advantage lies in the fact that the proposed plug-ins not only demonstrate remarkable performance when tackling real-world challenges, such as ratio error estimation problems, but also offer easily integration into existing evolutionary algorithm platforms, all while being highly user-friendly for evolutionary algorithm users.

1. Introduction

Many real-world optimization problems exhibit large-scale properties, encompassing diverse areas such as offloading policy optimization of edge servers [1], waste management [2], and power control optimization [3]. A defining trait of these large-scale optimization problems is the substantial number of decision variables, or high dimensionality, which can manifest across a spectrum of problem types—ranging from single-objective to multi-objective, constrained, and discrete problems [4]. Evolutionary algorithms (EAs), including differential evolution, genetic algorithm, particle swarm optimization (PSO), greylag goose optimization [5], and others, have garnered significant attention in the optimization domain over the past few decades. Their robust global search capabilities and scalability have made them indispensable in various complex applications with dynamic, large-scale properties, such as game design [6], job shop scheduling [7], and disease prediction [8].

However, EAs encounter the curse of dimensionality when tackling large-scale problems, resulting in a significant decline in performance and posing practical difficulties [9]. A prevalent solution to this challenge is the cooperative coevolution framework (CC), rooted in the divide-and-conquer principle. CC involves partitioning the decision variables into multiple subsets, each optimized by a separate EA [10]. Decision space reduction represents a burgeoning approach for EAs to address high-dimensional characteristics. While CC decomposes a complex, high-dimensional problem into multiple simpler, low-dimensional

☆ This research was supported in part by the National Natural Science Foundation of China (NSFC) (Grants No. U19A2061, No. 61902143), in part by National Key Research and Development Program of China (No. 2023YFB4502304).

* Corresponding author at: College of Computer Science and Technology, Jilin University, Changchun 130012, China.
E-mail addresses: yfsun22@mails.jlu.edu.cn (Y. Sun), xww@jlu.edu.cn (X. Wang), huangjh23@mails.jlu.edu.cn (J. Huang), sunb37@mail2.sysu.edu.cn (B. Sun), liangpung@hust.edu.cn (P. Liang).

https://doi.org/10.1016/j.swevo.2025.102100
Received 5 January 2025; Received in revised form 24 June 2025; Accepted 20 July 2025
Available online 13 August 2025

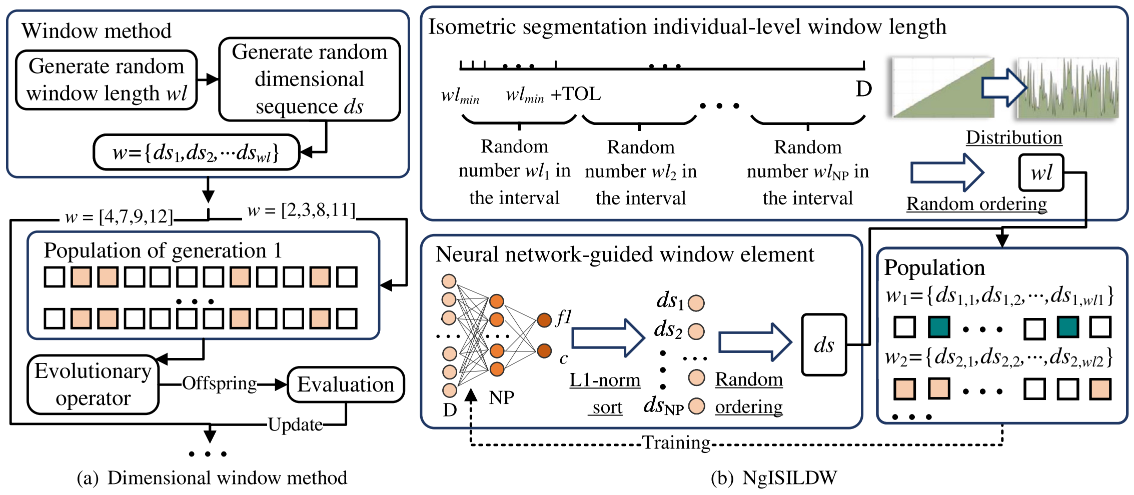

Window method
Generate random window length wl
Generate random dimensional sequence ds
$w=\{ds_1,ds_2,\cdots ds_{wl}\}$
w = [4,7,9,12]
w = [2,3,8,11]
Population of generation 1
Evolutionary operator
Offspring
Evaluation
Update
(a) Dimensional window method
Isometric segmentation individual-level window length
wl_{min}
wl_{min} +TOL
D
Random number wl_1 in the interval
Random number wl_2 in the interval
Random number wl_{NP} in the interval
Distribution
wl
Random ordering
Neural network-guided window element
D
NP
$f1$
c
ds_1
ds_2
ds_{NP}
L1-norm sort
Random ordering
ds
Population
$w_1=\{ds_{1,1},ds_{1,2},\cdots,ds_{1,wl1}\}$
$w_2=\{ds_{2,1},ds_{2,2},\cdots,ds_{2,wl2}\}$
Training
(b) NgISILDW

Fig. 1. Overview of the proposed dimensional window method and its improved version.

problems, EAs leveraging decision space reduction utilize techniques like problem transformations [11] and dimensionality reduction [12] to condense a complex problem into a solitary low-dimensional one. Furthermore, some methods have devised specialized operators for EAs tailored to large-scale attributes, such as expressive, fast explosion [13] and the random contrastive interaction strategy, to sustain rapid convergence.

While the methods above do mitigate the curse of dimensionality faced by EAs, they present certain limitations. On the one hand, integrating these methods with multi-objective EAs and constrained EAs is not as straightforward, hindering their cross-domain applicability. For instance, certain approaches rely on quantifying the magnitude of generational improvement to steer evolutionary processes. However, this metric cannot be as easily quantified on multi-objective problems as it is on single-objective problems. On the other hand, some problem models feature a variable number of decision variables, exhibiting high-dimensional attributes only in specific scenarios [14]. In such cases, EA users seek straightforward methods to enhance EA accuracy on large-scale problems, the existing large-scale evolutionary algorithms (LSEAs) may be overly complex for EA users, impeding the generalization of EA. Consequently, it may be a promising direction to develop plug-in-like large-scale handling techniques so that EA users can easily embed the technique into the EA they are using.

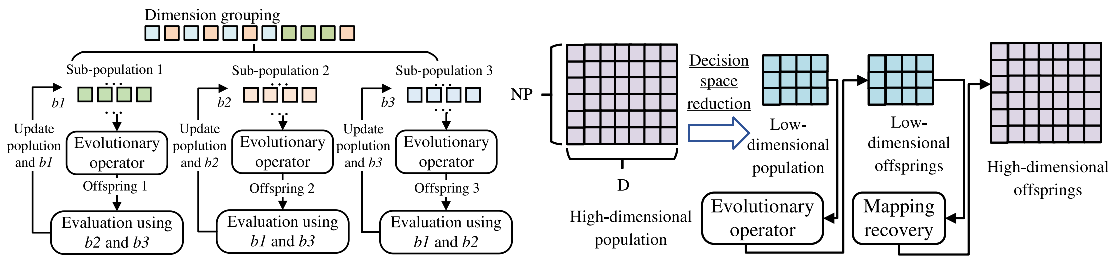
In light of the preceding analysis, this paper presents a lug-in-style large-scale handling technique, the dimensional window (DW) method, which can be incorporated into a majority of EAs. As illustrated in Fig. 1(a), the dimensional window method organizes several random dimensions into a window during each generation, permitting only the evolution of dimensions confined within this window. Additionally, taking into account the characteristics of the dimensional window method, this paper also proposes the isometric segmentation individual-level window length mechanism and the neural network-guided window element strategy. The refined dimensional window method, incorporating these strategies, is termed the neural network-guided isometric segmentation individual-level dimensional window method (NgISILDW), as depicted in Fig. 1(b). This isometric segmentation individual-level window length mechanism generates a window length for each individual that satisfies a particular distribution, thus allowing the population to explore more windows. Concurrently, the neural network-guided window element strategy prioritizes high-value windows through data-driven inference, guided by node importance metric. In contrast to existing methods, the approach proposed in this paper offers a modular design that facilitates seamless integration into EAs. Besides, it eschews population and evolutionary operations, thereby eliminating the need for specialized adaptations tailored to various problem types. To the best of our knowledge, this paper marks the inaugural effort to develop a plug-in-style large-scale handling technique. The main contributions of this paper are summarized as follows:

- The dimensional window method is proposed. In light of the practical application scenarios and the needs of EA users, this paper develops the dimensional window method. The core idea of the dimensional window method is to randomly constrain the evolution of specific dimensions in each generation, thereby avoiding the curse of dimensionality. The significance of the dimensional window method lies in its ability to operate across various problem domains and its ease of integration into existing EAs.
- The effectiveness of the dimensional window method is analyzed. This paper analyzes the effectiveness of the dimensional window method from a perturbation perspective and derives two key guidelines for improving the method: (1) increasing the population's exposure to diverse windows, and (2) ensuring that the population experiences valuable windows.
- Two key strategies are designed. Based on the analysis of the dimensional window method's effectiveness, this paper proposes isometric segmentation individual-level window lengths to enhance the population's exploration of diverse windows. Additionally, neural network-guided window elements are designed to help the population develop more valuable windows.
- Comprehensive experimental validation is conducted. Embedded within 12 distinct EAs, the plugin consistently enhanced performance, showcasing its robust generalization at the algorithm level. Furthermore, rigorous testing across three continuous and three discrete optimization problems confirmed its superior generalization at the problem domain level.

The remainder of this paper is structured as follows: Section 2 presents an overview of the prevalent LSEAs. Section 3 outlines the motivation for this work. Section 4 delves into the specifics of the proposed method, encompassing the windowing technique and its efficacy, the isometric segmentation individual-level window length, and the neural network-guided window element. Section 5 demonstrates the effectiveness of the proposed method through comprehensive experimental analysis. Section 6 acknowledges the limitations of the proposed method. Lastly, Section 7 concludes this paper.

2. Related work

Many critical real-world problems involve optimizing a vast number of decision variables, such problems are defined as large-scale optimization problems. This large-scale attribute transcends specific problem



(a) CC-based (b) Decision space reduction-based

Fig. 2. General process for CC-based and decision space reduction-based LSEAs.

types: Eq. (1) is a constrained multi-objective optimization problem, where $X = x_1, x_2, \ldots, x_D$, with m objectives, C_g inequality constraints, and C_h equality constraints. If both C_g and C_h are 0, this problem is a multi-objective optimization problem. If both C_g and C_h are 0 and m is 1, then this problem is a single-objective optimization problem. Large-scale attributes can be manifested in optimization problems including these three types of problems and the defining characteristic remains the high number of decision variables (D). Historically, problems with tens of variables were considered large-scale. As EAs have advanced, and the threshold has risen; contemporary large-scale problems typically involve hundreds, thousands, or even higher dimensions [4].

$$\begin{aligned} &\min F(X) = [f_1(x), f_2(x), \ldots, f_m(x)],\\ &\text{s.t}\\ &\quad g_i(x) \le 0, i = 1, 2, \ldots, C_g\\ &\quad h_i(x) = 0, i = 1, 2, \ldots, C_h \end{aligned} \tag{1}$$

The immense search space of large-scale problems contains countless local optima, posing significant challenges for many conventional EAs. This inherent difficulty directly motivated the development of specialized LSEAs. Without loss of generality, LSEAs can be categorized into three distinct groups: CC-based, decision space reduction-based, and special operators. CC-based LSEAs leverage the divide-and-conquer strategy to decompose high-dimensional problems into multiple low-dimensional sub-problems [15]. Meanwhile, decision space reduction-based methods facilitate EA searches in simplified, low-dimensional spaces [11]. Lastly, special operators are designed to ensure rapid convergence and mitigate the curse of dimensionality [13].

As illustrated in Fig. 2(a), CC-based EAs initially categorize the problem dimensions into distinct groups, each forming a sub-problem. These sub-problems are then optimized by their respective EAs. The evaluation of individuals within a subpopulation involves integrating them with the optimal individuals from other subpopulations [10]. For instance, subpopulation 1 tackles a sub-problem comprising the 1st, 3rd, 5th, and 7th dimensions of a high-dimensional problem. At the algorithm's inception, each subpopulation randomly selects one individual to serve as the initial optimal individual (denoted as $b1, b2, b3$ for simplicity). When subpopulation 1 generates its child population, each individual is paired with $b2, b3$, and their combined fitness value is assessed. Subsequently, the optimal individual from subpopulation 1 replaces the previous $b1$.

Grouping strategies significantly influence the performance of CC-based methods and are prevalent in the realm of large-scale optimization. Early grouping strategies primarily involved simple clustering of dimensions, such as random grouping and k-means grouping [16]. However, recent research on grouping strategies has increasingly emphasized the interactions among variables. For instance, MDG initially classifies variables into separable and non-separable categories and then employs binary trees to categorize non-separable variables into multiple subsets further [17]. EDG, on the other hand, leverages historical interrelationship information among variable groups to assess interactions between the remaining groups, enabling precise grouping [18].

Decision space reduction-based LSEAs harness dimensionality reduction techniques to transform the original high-dimensional problem into a simplified, low-dimensional one, aiming to reduce its complexity [4]. As depicted in Fig. 2(b), these LSEAs typically downscale the population suitable for the original high-dimensional problem into a low-dimensional population. Evolutionary algorithms are then employed to find the optimal solution within this reduced-dimensional space. Unlike CC-based LSEAs, which revert the low-dimensional population back to a high-dimensional population before evaluating the individuals [11]. From the perspective of decision space reduction techniques, notable methods include those utilizing weighting [19], PCA [20], and SVD [11]. Based on the problem type, single-objective and multi-objective LSEAs exist that leverage decision space reduction. Large multi-objective optimization has witnessed a surge of advanced LSEAs in recent years. For instance, KGEA clusters and aggregates the correlations of objective functions, effectively lowering the dimensionality of the high-dimensional problem. The simplified objective function variables exhibit reduced correlations, allowing for a flexible representation of preferences within the Pareto solution set [21]. On the other hand, SLMEA introduces a rapid clustering method to decrease the dimensionality of the search space and estimates the sparse distribution of optimal solutions by optimizing the binary vectors of individuals [22]. Decision space reduction-based LSEAs offer a more intuitive approach than CC frameworks and are easier to integrate with mature machine learning techniques, making it a promising direction.

Furthermore, rather than striving for more generalized large-scale techniques or frameworks, certain approaches concentrate on enhancing the large-scale search capabilities of specific evolutionary algorithms. For instance, LSMaOEA introduces a spatial sampling method designed to augment existing algorithms' diversity that relies on spatial sampling [23]. SFEA employs sigmoid functions to adjust the fuzzy phase and introduces an advanced fuzzy operator to bolster the performance of evolutionary algorithms in addressing constrained multi-objective optimization problems [24]. SDLSO provides each particle in the PSO with an opportunity to progress directly to the next generation, and each updated particle learns exclusively from its dominator, thereby achieving excellent convergence [25].

3. Motivation

The LSEAs detailed in Section 2 have demonstrated remarkable outcomes across numerous large-scale problem domains. However, due to their potential for excessive complexity and challenges in generalization, they may not always be user-friendly for EA users in certain scenarios. Consider the example of the UAV communication sum-rate optimization problem, where an LED-equipped UAV is deployed in the air to offer both lighting and communication services to N ground users (GUs). The UAV-mounted LED transmitter efficiently conveys data to these N GUs by dynamically adjusting its transmit power, utilizing the same time–frequency resources. The optimization problem

Table 1
Results of constrained EAs for solving the UAV communication sum-rate optimization problem.

Number of GUs	10	20	50	100	200	500
	Sum rate			CV		
CCFE — ECHT	**1.91E+01**	1.41E+01	8.59E+00	**1.54E+00**	1.26E+02	5.44E+03
CEDE — DRL	1.86E+01	**1.46E+01**	**8.70E+00**	1.64E+00	**2.10E+01**	**3.75E+03**
DeCODE	1.90E+01	1.34E+01	8.52E+00	1.91E+00	4.14E+02	4.87E+04
VMCH	1.90E+01	1.43E+01	8.57E+00	3.35E+01	5.92E+04	3.95E+06

is formulated as follows:

$$\begin{aligned}
&\max_{\{l,p\}} \sum_{i\in\mathbb{N}} \log_2(1+\frac{h_i p_i}{n_0+\sum_{j\in\ :h_j>h_i} h_i p_j}) \\
&s.t.\quad p_i \geq 0, \forall i \in \mathbb{N} \\
&\sum_{i\in\mathbb{N}} p_i \leq P_{\max} \\
&\sum_{i\in\mathbb{N}} \sqrt{p_i} \leq C \\
&\overline{h}_{i+1}p_i - \sum_{j=i+1}^{N} \overline{h}_{i+1}p_j \geq 0, i=1,\dots,N-1 \\
&SR_i \geq SR_i^{th}, \forall i \in \mathbb{N} \\
&x_u^2 + y_u^2 \leq R^2
\end{aligned} \tag{2}$$

Where p is the power allocation sequence, p_i i.e., the power allocated by the UAV to the GN_i, $l = x_u, y_u, h$ denotes the positional coordinates of the UAV, h is the signal gain, computed from the position of l and the GN, and SR_i denotes the data rate of the GN_i. n_0, $P_{\max}$, C, and R denote the noise, the total power threshold, coefficients, and the coverage radius, respectively, all of which are fixed values. The problem model is not the work of this paper, and the specific principles and sub-models are described in Ref. [26].

The decision variables for the problem above consist of p and l, totaling N+2 in number, which are directly tied to the count of GUs. Typically, the number of GUs ranges between 10 and 20 in general scenarios, but in specific situations, such as emergency communication, this number can surge significantly. The constraints of this problem exhibit a certain degree of coupling with the objective function. As the number of decision variables increases, the feasible domain diminishes notably, posing a significant optimization challenge. So, in this problem, the number of GUs can be greater than 20, which challenges the optimization performance of a normal EA, and the EA user has two viable options:

- Option 1: Continue using plain EAs that are not specifically tailored for large-scale attributes.
- Option 2: Transition to LSEAs or employ LSEAs when in the higher dimensional.

Option 1 may not be feasible in practice. To illustrate this, Table 1 presents the results of solving this problem using four advanced constrained EAs—CCFE - ECHT [27], CEDE - DRL [28], DeCODE [29], and VMCH [30]. When the number of GUs reaches or exceeds 100, EAs that lack large-scale capabilities cannot search for a feasible solution. Therefore, we present their constraint violation (CV) in Table 1. As the number of GUs increases, the performance of these algorithms declines rapidly. When the dimensionality is excessively high, no algorithm can even locate a feasible solution, and the CV becomes intolerable at dimensions greater than 200. Therefore, option 1 may yield poor optimization results or make identifying any feasible solution impossible.

Option 2, switching to LSEAs or using them when appropriate, is also not entirely satisfactory for EA users. On the one hand, adopting LSEAs in low-dimensional scenarios may yield inferior optimization results. On the other hand, this solution necessitates substantial modifications to the algorithm. EA users tend to use highly integrated evolutionary computing platforms such as PlatEMO [31] and MTO [32], and in-depth design of algorithms is not an easy task for EA users and is not a job they want to do work.

In summary, the demands of practical applications necessitate developing a user-friendly large-scale handling technique for EA users, aiming to boost the performance of EAs in large-scale real-world problems. This technique should possess the following characteristics:

- Cross-problem domain adaptability. The UAV communication sum-rate optimization problem is a constrained single-objective problem. Still, more scenarios require large-scale attributes, including discrete [33] and multi-objective [34] scenarios. Therefore, large-scale handling techniques must be adapted to multiple problem domains.
- Easy to embed. EA users do not want or care about the design of the algorithm and may use more than one EA, so it is desirable that the large-scale handling techniques can be integrated with existing EAs that are not designed for large-scale characterization with a simple operation.

The works presented in Section 2 are only compatible with some of these criteria. The CC-based LSEA necessitates the selection of an optimal individual b from each sub-population to form the comprehensive optimal solution. However, when addressing multi-objective or constrained problems, the selection of b must be redefined, which hinders its generalization across various problem domains. Furthermore, integrating CC with existing EAs is not straightforward and can be challenging for EA users to manipulate. Additionally, the grouping strategy may not apply to other problem types and requires redesign. Similarly, decision space reduction-based LSEAs encounter comparable challenges, as many reduction techniques hinge on population fitness or specialized algorithms. More critically, both CC-based LSEAs and decision space reduction-based LSEAs are operationally intricate and challenging to embed. While special operators may offer simpler implementation, they are tailored for specific algorithms and scale poorly. Consequently, there is room for improvement in user-friendliness in the existing work, motivating us to develop the dimensional window method.

4. Methodology

Driven by the actual needs of EA users and inspired by CC-based LSEAs, this paper proposes the dimensional window method and its improved version as follows.

4.1. Dimensional window method

Assume that the EAs initialize the population as X, undergo an evolutionary operation to produce the offspring O, and then update X by a selection operation This cycle is repeated for g generations, ultimately outputting the optimal individual or solution set. In simpler terms, the process can be summarized as:

$$O^i = \text{EvolutionaryOperation}(X^i), 1 \leq i \leq g \tag{3}$$

$$X^{i+1} = \text{SelectionOperation}(X^i, O^i), 1 \leq i \leq g-1 \tag{4}$$

where X and O denote population and offspring, respectively.NP represents the population size and D denotes the problem dimension. When D becomes excessively large, the problem's complexity skyrockets, effectively causing the EAs to suffer from the curse of dimensionality. To tackle this issue, CC-based LSEAs are devised by partitioning D into some sub-problems, each optimized by EA. In this context, the population structure is as follows:

$$SX = [SX_1, \dots SX_i, \dots, SX_k], SXi \in \mathbb{R}^{NP\times SD_i} \tag{5}$$

Where k denotes the number of sub-problems, SD_i and SX_i represent the dimension size and population for the ith sub-problem, respectively. The combined population of all sub-populations forms $SX \in$

$\mathbb{R}^{NP\times D}$. CC-based LSEAs have demonstrated promising results on large-scale problems by allowing sub-populations to evolve independently. However, limitations in individual evaluation and grouping operations hinder their generalization capabilities, as Section 3 discusses.

The dimensional window method proposed in this paper is inspired by the divide-and-conquer idea of CC-based LSEAs and deals with large-scale problems in the form of serial optimization of the evolutionary process, as shown in Fig. 1(a). Before the start of each iteration, the dimensional window method first randomly generates a window w that contains a partial index of dimensions:

$$w = \{ds_j \mid j \in \{1, 2, \ldots, wl\}\} \tag{6}$$

The window is generated based on a window length wl, and a sequence of dimensions ds, where wl is a random integer:

$$wl \sim \mathrm{Uniform}(wl_{min}, D) \tag{7}$$

Where wl_{min} is the hyper-parameter. ds is then a uniform random order of the sequence of dimensions:

$$ds = \mathrm{RandomOrder}(D) \tag{8}$$

Then, after the algorithm generates the offspring and has not evaluated them, it uses w to update O:

$$\widetilde{O}^i_{j,m} = \begin{cases} O^i_{j,m}, & if\ m \in w \\ X^i_{j,m}, & otherwise \end{cases},\ 1 \leqslant i \leqslant g,\ 1 \leqslant j \leqslant NP,\ 1 \leqslant m \leqslant D \tag{9}$$

$\widetilde{O}$ is the population of new offspring affected by the window, and the value of the jth column of $\widetilde{O}$ is the value of the jth column of O if the jth column is indexed in the window w, and the value of the jth column of X otherwise.

Finally, the new offspring are evaluated and participate in the next evolution:

$$X^i = \mathrm{SelectionOperation}(X^{i-1}, \widetilde{O}^{i-1}), 1 \leqslant i \leqslant g \tag{10}$$

As depicted in Fig. 1(a), the core of the dimensional window method lies in randomly generating a window that encompasses a set of dimension indexes during each iteration, thereby limiting the evolutionary process solely to the dimensions within this window. A notable distinction between the proposed dimensional window method and CC-based EAs can be observed by comparing Figs. 1(a) and 2(a). The dimensional window method maintains the original population structure and algorithmic flow, eliminating the need to merge individuals within sub-populations. Consequently, unlike CC-based EAs, the dimensional window method can be applied to multi-objective, constrained, or discrete EAs without requiring any specialized modifications.

The core difference between the dimension window method and CC can be outlined as the former randomly selects dimensions to optimize at each iteration, whereas the latter requires the division of populations before the algorithm starts, with each population optimizing a different dimension. This is the reason why our approach is more general than CC. In terms of convergence and computational efficiency, the dimension window method emerges as the clear frontrunner, outperforming CC. To illustrate these advantages, we briefly validate the classical multi-objective EA GDE3, CC-based GDE3 (CC-GDE3), and DW-GDE3 embedded with the dimensional window method on the BT1 and DTLZ1 problems, and the results are shown in Table 2.[1]

While CC-GDE3 demonstrates superior IGD and GD performance compared to GDE3, it incurs substantially increased computational time. GDE3 exhibits a per-generation time complexity of approximately $O(NP \cdot D)$. In contrast, CC-GDE3 partitions the population into two subpopulations, each optimizing distinct dimensions (D_1 and D_2 respectively) before merging solutions during evaluation, rendering it slower than GDE3. DW-GDE3, however, demonstrates superior performance across all metrics, notably achieving shorter runtimes than GDE3. Its per-generation cost comprises a constant $O(1)$ overhead for generating wl and ds, followed by evolutionary operations on only wl dimensions with complexity $O(NP \cdot wl)$. Combined with the condition $wl < D$, ensures DW-GDE3 operates faster than GDE3.

Regarding convergence performance, as illustrated in Fig. 3, GDE3 exhibits the poorest inverted IGD convergence, while CC-GDE3 demonstrates the weakest GD convergence. Notably, DW-GDE3's IGD convergence trajectory initially trails CC-GDE3 only marginally during the early evolutionary stages. However, it rapidly surpasses both competitors, maintaining a dominant performance advantage thereafter. Although its GD convergence is less pronounced than its IGD results, DW-GDE3 still achieves superior GD outcomes relative to the other algorithms.

In summary, the dimensional window method stands out as a novel paradigm among large-scale optimization techniques. It can improve the solution performance at a smaller time cost, and also does not require any a priori knowledge about the offspring. Even the subsequent enhanced version can be integrated into EA with minimal cost, making it more friendly to EA practitioners.

4.2. Why the dimensional window method works

To assess the effectiveness of the windowing method, we examined the curse of dimensionality phenomenon through a perturbation lens. The standard differential evolutionary Algorithm (DE) achieves a result of approximately 6.6242E+06 when solving f1 in CEC2013 (D=1000), which lags significantly behind the outcomes of leading LSEA and converges at a glacial pace, succumbing to the curse of dimensionality.

Subsequently, we employed the optimized population yielding the optimal value of 6.6242E+06 as the initial population X. We conducted 10,000 generations of evolutionary operations with the crossover rate CR=[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9], respectively. CR dictates the proportion of content from the offspring individuals that the parent individuals retain. Generally, as CR increases, so does the divergence between the offspring and parent populations, indicating a more significant perturbation of individuals during the evolutionary process. Conversely, a lower CR results in a lesser degree of perturbation.

Fig. 4(a) depicts the convergence over 10,000 iterations for various CR. Notably, the slowest convergence rates are observed at CR values of 0.8 and 0.9, whereas faster convergence is attained at CRs of 0.1, 0.2, and 0.5. Fig. 4(c) presents the difference between the current and previous generation populations for different CR values, computed using the following formula:

$$diff^g = \sum_i^{NP} \sum_j^{D} |X^g_{i,j} - X^{g-1}_{i,j}| \tag{11}$$

It is observed that as the CR increases, the magnitude of the perturbation applied to the population also increases. For the DE tailored to small-scale problems, a CR of 0.9 is typically used. However, when dealing with high-dimensional problems (D = 1000), this value is clearly not optimal, and better CRs exhibit more minor perturbations to the solution. Furthermore, it is not necessarily true that more minor perturbations lead to better performance in large-scale problems. For instance, at CR = 0.5, the perturbation is significantly larger than at CR = 0.1 and 0.2, yet the results are comparable. This suggests that the perturbation at CR = 0.5 is more valuable than those at the lower CRs.

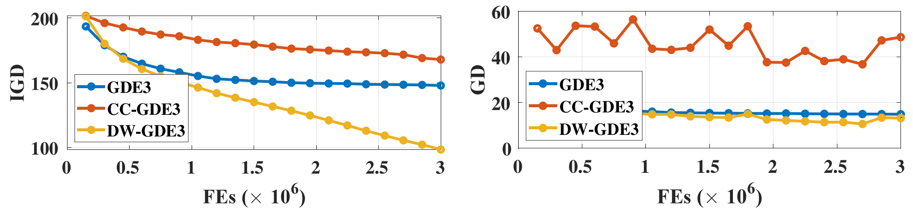
Hence, the challenges EAs face in solving large-scale problems lie in the fact that the evolutionary operations can either perturb the solution excessively or be too coarse-grained. To achieve satisfactory results on large-scale issues, EAs must employ perturbation schemes that are both (1) cannot be overpowered and (2) valuable.

The dimensional window method allows only partial dimension updates at each iteration, resulting in a smaller disruption to the solution

[1] The dimensions of the two problems are 1000. The maximum number of function evaluations is 3E6. CC-GDE3 divided two populations.

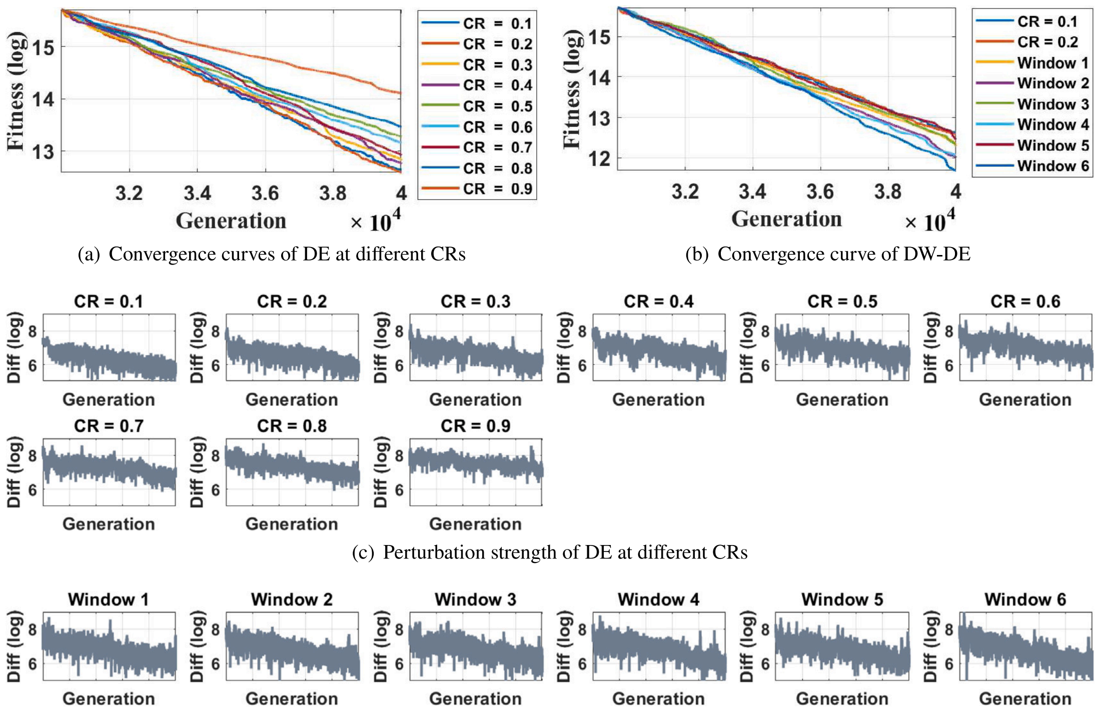
Table 2
Mean(variance) results of GDE3, CC-GDE3, and DW-GDE3(ours).

Problem	Method	GD	IGD	Runtime(s)	Time complexity
BT1	GDE3	3.81E+1 (9.82E+0)	1.73E+2 (1.44E+1)	3.46E+2 (4.20E+0)	$O(NP \cdot D)$
	CC-GDE3	1.47E+1 (7.68E−2)	1.46E+2 (1.02E+0)	4.31E+2 (3.45E+0)	$O(NP \cdot D_1) + O(NP \cdot D_2) + O(NP)$
	DW-GDE3	**1.40E+1 (2.23E+0)**	**1.19E+2 (7.80E+0)**	**2.96E+2 (3.46E+0)**	$O(1) + O(NP \cdot wl)$
DTLZ1	CC-GDE3	2.12E+3 (1.40E+2)	1.64E+4 (1.47E+3)	4.37E+2 (9.42E+0)	$O(NP \cdot D)$
	GDE3	7.66E+2 (2.15E+2)	3.98E+3 (2.41E+3)	4.15E+2 (6.81E+0)	$O(NP \cdot D_1) + O(NP \cdot D_2) + O(NP)$
	DW-GDE3	**4.06E+2 (2.77E+1)**	**3.15E+3 (2.43E+2)**	**2.96E+2 (1.44E+0)**	$O(1) + O(NP \cdot wl)$



(a) convergence curves of IGD (b) convergence curves of GD

Fig. 3. Convergence curves of GDE3, CC-GDE3, and DW-GDE3 on BT1 problem.



(a) Convergence curves of DE at different CRs (b) Convergence curve of DW-DE

(c) Perturbation strength of DE at different CRs

(d) Perturbation strength of DW-DE

Fig. 4. Convergence curves and perturbation strength of DE and dimensional window-DE.

compared to the original algorithm. This window can be viewed as a perturbation scheme that attenuates the original evolutionary operation. As shown in Fig. 4(b), we embedded the dimensional window method into DE and ran it independently 6 times with CR = 0.9. The dimensional window method works better than the cases with CR = 0.1 and 0.2. Regarding perturbation, Fig. 4(d) reveals that the dimensional window method exhibits a stronger impact in the initial stages than CR values of 0.1 and 0.2 while maintaining comparable performance in later stages. This indicates that the perturbation introduced by the dimensional window method holds value.

Table 3 lists the average perturbation strengths and optimal fitness values for running DE under different CRs and 6 independent runs of dimensional window-DE using the same initial population (maximum number of evaluations of the function is 3E+06, NP=100, D=1000). The average perturbation strength increases with increasing CR, and the fitness becomes significantly worse when CR > 0.4, indicating that coarse-grained perturbations are not suitable for solving large-scale problems. For the dimensional window method, the results of all six runs were better than the best CR (i.e., CR = 0.2), but the average perturbations were all between CR = 0.4 and CR = 0.5, suggesting that the dimensional window method performs a more valuable perturbation without performing too weak a perturbation, which we surmise is the reason why the dimensional window method is effective.

In summary, the effectiveness of the dimensional window method lies in attenuating the perturbation intensity of the evolutionary operation and making it more valuable for perturbing the population. So keys

Table 3
Results for DE with different CR values and window DE with 6 independent runs on CEC2013 f1.

	CR = 0.1	CR = 0.2	CR = 0.3	CR = 0.4	CR = 0.5	CR = 0.6	CR = 0.7	CR = 0.8	CR = 0.9
Average perturbation	**6.41E+02**	7.69E+02	9.09E+02	1.07E+03	1.18E+03	1.36E+03	1.59E+03	1.76E+03	2.03E+03
Best fitness	3.06E+05	2.91E+05	3.67E+05	3.45E+05	5.77E+05	5.19E+05	4.07E+05	7.01E+05	1.33E+06
	Window1	Window2	Window3	Window4	Window5	Window6			
Average perturbation	1.11E+03	1.07E+03	1.12E+03	1.09E+03	1.09E+03	1.08E+03		–	
Best fitness	2.17E+05	1.63E+05	2.28E+05	1.75E+05	2.54E+05	**1.19E+05**			

to improving the performance of the dimensional window method are to:

(1) Key point 1: Encouraging the population to explore a greater number of windows.
(2) Key point 2: Ensuring the population encounters valuable windows.

This resonates with the balance between exploration and exploitation in EAs. In the refined Sections 4.3 to 4.5, our strategy design is grounded in these two keys.

Additionally, the analogy between the dimensional window method and CC's learning pattern can also corroborate why the dimensional window method is effective. For the ith sub-problem of CC, his offspring $SO_{i,j}$ is learned from the sub-population SX_i (including the sub-problem dimensions), whereas for the dimensional window method, the individual O_j learns only some of the dimensions, which is similar to the sub-problem model of CC, however, it learns from the population X (including all the dimensions). Therefore, the learning pattern of the dimensional window method is similar to that of CC, only in the object of learning, which side by side shows the effectiveness of the dimensional window method.

4.3. Isometric segmentation individual-level window length

To bridge key point (1), we make a twofold effort: firstly, to transition the windows from the population level to the individual level, and secondly, to ensure that the distribution of window lengths encompasses the range $[wl_{min}, D]$ as comprehensively as feasible. As illustrated in the population depicted in Fig. 1(b), each individual is endowed with a unique window, and each of these windows is obtained from its specific corresponding wl and ds. The windows at the individual level can be depicted through sets:

$$ilw = \{w_1, w_2, \dots w_{NP}\} \tag{12}$$

Of course, there must also be NP pairs of wl and ds to control the generation of windows:

$$ilds = \{ds_1, ds_2, \dots ds_{NP}\} \tag{13}$$

$$ilwl = \{wl_1, wl_2, \dots wl_{NP}\} \tag{14}$$

The window for the ith individual at this point is:

$$w_i = ilw_i = \{ilds_{i,1}, ilds_{i,2}, \dots, ilds_{i,ilwl_i}\}, 1 \leqslant i \leqslant NP \tag{15}$$

In this manner, when compared to the dimensional window method where only a single window is observable per population generation, the individual-level approach allows for the observation of NP windows, a big step towards key point (1).

The distribution of $ilwl$ can further aid the population in exploring various windows. However, if a uniform distribution is employed to generate window lengths for all individuals, the resulting distribution shown in Fig. 5(a) may lead to some individuals having highly similar window lengths. Fig. 5(b) demonstrates the distribution of window lengths after sorting them in ascending order; notably, the window lengths of the 95th to 97th individuals are very close, thereby wasting the opportunity for the population to explore a wider range of window length possibilities. For this reason, we have devised an Isometric segmentation mechanism. Initially, a sequence of class isometries is generated, starting from wl_{min}, with a tolerance of $D/(NP+1)$, and terminating at D:

$$a_i = \begin{cases} wl_{min}, & if\ i = 1 \\ D, & if\ i = NP+1 \\ \left\lfloor i \cdot \dfrac{D}{NP+1} \right\rfloor, & otherwise \end{cases}, \ 1 \leqslant i \leqslant NP+1 \tag{16}$$

The length of this sequence is NP+1,. Next, the window length for each individual is generated using a:

$$wl_i = \lfloor r \cdot (a_{i+1} - a_i) + a_i \rfloor, \ 1 \leqslant i \leqslant NP \tag{17}$$

where $\lfloor \cdot \rceil$ represents the rounding operation, and Wl_i is a randomly selected integer within the range between a_i and a_{i+1}. Now, the distribution of $ilwl$ is depicted in Fig. 5(c), which is significantly sharper compared to the uniform distribution. Consequently, it becomes almost impossible for some individuals to possess windows of overly similar lengths. Lastly, to enhance the probability that different individuals observe distinct windows, the $ilwl$ is randomly ordered:

$$ilwl = \text{RandomOrdering}([wl_1, wl_2, \dots, wl_{NP}]) \tag{18}$$

The distribution of ilwl at this stage is illustrated in Fig. 5(d).

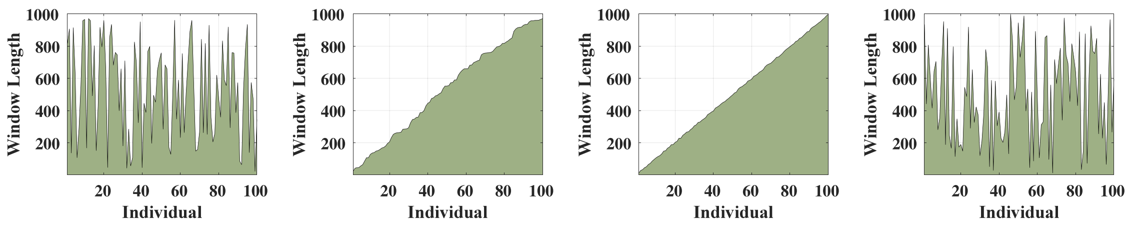
The above scheme boosts the number of windows seen by the population by a factor of NP in each iteration. Furthermore, it strives to ensure that each individual possesses a unique window length, to fitting the key point (1).

4.4. Neural network-guided window element

Individual-level windows and adjustments to the wl distribution compensate for key point (1), while to bridge key point (2), our primary emphasis lies on the design of the ds, aiming to ensure that it encapsulates a sequence of dimensions that aligns with the current evolutionary trend of the population, reflecting the importance of the different dimensions. Nevertheless, assessing this significance from a theoretical standpoint is challenging, making a data-driven approach potentially more intuitive. Within the realm of network pruning, a crucial preliminary understanding is that neurons vary in their contribution to the neural network model [35], with the removal of neurons that contribute minimally having negligible or no impact on model performance [36]. Drawing inspiration from this, this paper proposes the design of a three-layer neural network net to model the importance of dimensions, as illustrated in Fig. 6(a).

The first layer of *net*, known as the input layer, comprises D nodes and is responsible for receiving input data from the individuals in the EA. The second layer consists of NP nodes, while the third layer, which serves as the output layer, produces the fitness scores of the individuals. The number of nodes in this output layer is contingent upon the nature of the problem. For instance, in the case of a single-objective problem, the output layer would consist of one node. If the problem involves three objectives and three constraints, the output layer would have four nodes: the first three nodes output the values of the objective functions, while the fourth node outputs the CV.

Let $FC = [f1, f2, \dots, C]$ represent the matrix that encompasses the fitness values and CV of the population, $FC \in \mathbb{R}^{NP, M+(1 \text{ or } 0)}$. To train



(a) Uniform distribution (b) Sorted uniform distribution (c) Isometric segmentation without random ordering (d) Isometric segmentation

Fig. 5. Distribution of window lengths.

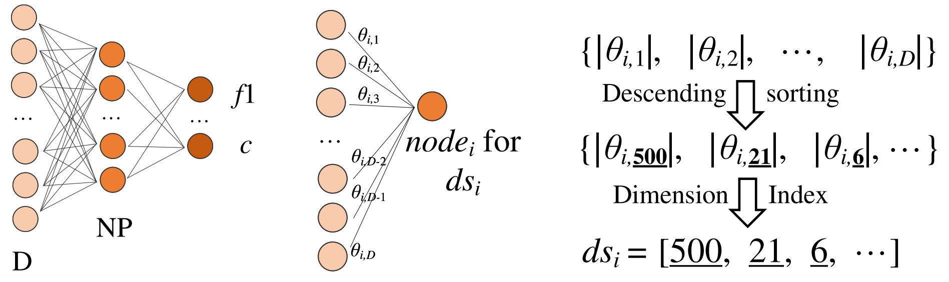

D
NP
f1
c
$\theta_{i,1}$
$\theta_{i,2}$
$\theta_{i,3}$
$\theta_{i,D-2}$
$\theta_{i,D-1}$
$\theta_{i,D}$
$node_i$ for ds_i
$\{|\theta_{i,1}|, |\theta_{i,2}|, \cdots, |\theta_{i,D}|\}$
Descending sorting
$\{|\theta_{i,\underline{500}}|, |\theta_{i,\underline{21}}|, |\theta_{i,\underline{6}}|, \cdots\}$
Dimension Index
$ds_i = [\underline{500}, \underline{21}, \underline{6}, \cdots]$
(a) Structure of neural network (b) Weight of the i-th node (c) Dimensional sequence of i-th individual

Fig. 6. Schematic representation of the neural network-guided window element.

the *net*, it is necessary to normalize both the population X and the fitness and constraint violation matrix FC:

$$\overline{X} = \frac{X - X_{\min}}{X_{\max} - X_{\min} + \varsigma} \tag{19}$$

$$\overline{FC} = \frac{FC - FC_{\min}}{FC_{\max} - FC_{\min} + \varsigma} \tag{20}$$

Where $X_{\max}$ and $X_{\min}$ represent the maximum and minimum element values within the population X, respectively, and $FC_{\max}$ and $FC_{\min}$ correspond to the maximum and minimum element values within the FC. A small constant ς, set to 0.0001, is introduced to prevent division by zero errors. Normalization of the X and the FC is crucial because, in large-scale problems, the fitness of the population can be excessively large, which could result in gradient explosion if used directly for training the *net*. Subsequently, the *net* is trained using the normalized $\overline{X}$ and $\overline{FC}$:

$$\ell = \frac{1}{NP} \cdot (net(\overline{X}) - \overline{FC})^2 \tag{21}$$

$$\theta n = SGD(net, \ell, lr, \gamma) \tag{22}$$

Where θn denotes the trainable parameter of the *net*, Eq. (21) represents the mean square error loss between the predicted FC of X by *net* and the actual FC. Stochastic Gradient Descent (SGD) is employed as the training method, with the learning rate lr and momentum γ serving as hyper-parameters for SGD. The iterations of Eq. (21)–(22) are repeated e times to finalize the training process. SGD is a widely adopted model training technique in deep learning; for further specifics, please refer to Ref. [37].

The trained *net* is actually an agent model that predicts the fitness value and CV of the current population. Here, the second layer of nodes plays a crucial role in determining prediction accuracy, as these nodes collectively influence the FC. Conceptually, each node in the second layer can be seen as a window encompassing all dimensions. Fig. 6(b) illustrates that each second-layer node possesses a weight array θ connecting to D input nodes. Based on prior knowledge discussed previously, let us consider the ith node $node_i$ in the second layer as an example. The D nodes preceding it vary in their importance to $node_i$, and this importance is reflected in the weights A of $node_i$ relative to the input layer's weights. Therefore, the influence of different dimensions on weight vector w_i can be modeled through θ_i. The dimensions represented by the more important weights in θ_i are also more critical to w_i and should be prioritized in the ds_i sequence. At this juncture, determining a reasonable ds_i for w_i transforms into a neuron importance ranking problem, a prevalent issue in the domain of network pruning. In this paper, we employ the classic L1-norm method [38] to rank the importance of input neurons to $node_i$:

$$\overline{\theta}_i = \{|\theta_{i,1}|, |\theta_{i,2}|, \ldots |\theta_{i,NP}|\} \tag{23}$$

$$ds_i = \text{DescendSorting}(\overline{\theta}_i) \tag{24}$$

As depicted in Fig. 6(c), the absolute values of θ_i are first computed. Subsequently, a descending sort operation is applied to obtain an indexed ordering of the input nodes based on their influence on $node_i$, from the largest to the smallest. This ordered sequence serves as ds_i. Lastly, a random permutation of ds_i is executed, analogous to the window length randomization described in Section 4.3:

$$ilds = \text{RandomOrdering}([ds_1, ds_2, \ldots, ds_{NP}]) \tag{25}$$

The neural network-guided window element strategy simulates the impact of windows on fitness and CV in a data-driven manner. It further leverages the concept of network pruning to ascertain the importance of dimensions to windows, thereby yielding more valuable $ilds$ that align with the current evolutionary trend of the population. Notably, due to the scalability of neural networks, this strategy can flexibly adjust the number of nodes in the output layer, enabling it to adapt to various types of problems.

4.5. Neural network-guided isometric segmentation individual-level dimensional window method

The proposed NgISILDW consists of three core components: isometric segmentation window length, neural network-guided window element, and the evolution guided by the idea of the window method. Using DE as an embedding example (Algorithms 1–3), we detail their implementation below.

The isometric segmentation window length component first generates an increasing sequence a bounded by $wl_{min} \leq a \leq D$ with interval

$D/(NP+1)$ (lines 1–4 of Algorithm 1). This sequence then determines individual window lengths (lines 5–8 of Algorithm 1), which are randomly permuted to form individual-level window lengths $ilwl$(line 9 of Algorithm 1). The neural network-guided window element component initializes a three-layer neural network *net* at generation one, featuring D input neurons, NP hidden-layer neurons, and output dimensionality matching objective functions (lines 2–5 of Algorithm 2). Retrained every t generation using population X with fitness or/and constraint violations (lines 6–12 of Algorithm 2), its input-hidden layer weights are sorted descending to construct individual-level window element $ilds$ (lines 14–19 of Algorithm 2).

Algorithm 1: Isometric segmentation individual-level window length

```
    Input: $wl_{\min}$, $D$, $NP$
    Output: $ilwl$
1   // Generate an isometric sequence that starts at $wl_{\min}$, is spaced at $D/(NP+1)$, and terminates at $D$.
2   for $i \leftarrow 1$ to $NP$ do
3       $a_i \leftarrow$ Eq. (16) $(wl_{\min}, D, NP)$ ;
4   end
5   // Generate window lengths for each individual using $a$.
6   for $i \leftarrow 1$ to $NP-1$ do
7       $wl_i \leftarrow$ Eq. (17) $(a_i, a_{i+1})$;
8   end
9   $ilwl \leftarrow$ Eq. (18) $([wl_1, wl_2, \dots, wl_{NP}])$;
```

Algorithm 2: Neural network-guided window element

```
    Input: $X$,$FC$, $D$,$NP$,$M$,$c$, $t$, $g$, $lr$, $\gamma$
    Output: $ilds$
1   if $g \% t = 0 \;||\; g = 1$ then
2       // Initialize the neural network.
3       if $g = 1$ then
4           $net \leftarrow$ CreateNeuralNetwork$(D, NP, M, c)$;
5       end
6       // Training the neural network.
7       $\overline{X} \leftarrow$ Eq. (19) $(X)$;
8       $\overline{FC} \leftarrow$ Eq. (20) $(FC)$;
9       for $i \leftarrow 1$ to $e$ do
10          $\ell \leftarrow$ Eq. (21) $(net, \overline{X}, \overline{FC})$;
11          $net \leftarrow$ Eq. (22) $(net, \ell, lr, \gamma)$;
12      end
13  end
14  // Generate window elements for each individual using $net$.
15  for $i \leftarrow 1$ to $NP$ do
16      $\overline{\theta}_i \leftarrow$ Eq. (23) $(net, \theta_i)$;
17      $ds_i \leftarrow$ Eq. (24) $(\overline{\theta}_i)$;
18  end
19  $ilds \leftarrow$ Eq. (25) $([ds_1, ds_2, \dots, ds_{NP}])$;
```

The third core is the evolution guided by the window method idea. Taking DE as an example, embedding NgISILDW only requires a few simple steps, as shown in Algorithm 3. During iteration, $ilds$ and $ilwl$ are first generated per individual (lines 5–7 of Algorithm 3). After standard mutation and crossover produces offspring u (lines 9–21 of Algorithm 3), each u_i's window w_i is formed by the leading $ilwl_i$ elements of $ilds_i$ (lines 22–23 of Algorithm 3). Dimensions outside w_i inherit parent x_i's values (lines 24–29 of Algorithm 3), confining evolution within designated windows before selection (lines 30–34 of Algorithm 3).

It is evident that the integration of NgISILDW into EAs is a straightforward process. The original initialization and evolution operators remain unmodified, requiring only pre-evolution window computation

Algorithm 3: NgISILDW-DE

```
    Input: Problem, $wl_{min}$, $NP$, $t$, $g$, $lr$, $\gamma$, scaling factor $F$,
           crossover rate $CR$
    Output: $X$
1   $D, M, c \leftarrow Problem$;
2   Initialize population $X = \{x_1, x_2, \dots, x_{NP}\}$;
3   $g \leftarrow 1$;
4   while termination condition not met do
5       // Generate window lengths and window elements for each individual.
6       $ilwl \leftarrow$ Algorithm 1 $(wl_t extmin, D, NP)$;
7       $ilds \leftarrow$ Algorithm 2 $(X, FC, D, NP, M, c, t, g, lr, \gamma)$;
8       for $i \leftarrow 1$ to $NP$ do
9           // Mutation operation
10          $\widetilde{O}_i \leftarrow$ Eq. (9) $(a, net, X, FC, t, g, lr, \gamma)$;
11          Randomly select 3 distinct integers
                $r_1, r_2, r_3 \in [1, NP] \setminus \{i\}$;
12          $v_i \leftarrow x_{r_1} + SF \cdot (x_{r_2} - x_{r_3})$;
13          // Crossover operation
14          Randomly select 1 integer $j_{rand} \in [1, D]$;
15          for $j \leftarrow 1$ to $D$ do
16              if $rand \le CR$ or $j = j_{rand}$ then
17                  $u_{i,j} \leftarrow v_{i,j}$;
18              else if condition 2 then
19                  $u_{i,j} \leftarrow x_{i,j}$;
20              end
21          end
22          // Generate the window for the current offspring using $ilwl$ and $ilds$.
23          $w_i \leftarrow$ Eq. (6) $(ilds_i, ilwl_i)$;
24          // Restrict the evolution of offspring only within the given window.
25          for $j \leftarrow 1$ to $D$ do
26              if $j \notin w_i$ then
27                  $u_{i,j} \leftarrow x_{i,j}$;
28              end
29          end
30          // Selection operation
31          if $u_i$ better than $x_i$ then
32              $x_i \leftarrow u_i$;
33          end
34      end
35      $g \leftarrow g + 1$;
36  end
```

and post-evolution dimensional constraints. Crucially, it does not even require any modification to the selection operation thus can be flexibly embedded in EAs applicable to various problem domains.

5. Experiment

In this section, we experimentally verify the efficacy of the proposed method. Initially, we determine the optimal hyper-parameter t using Pareto frontier (PF) optimization. Subsequently, we validate the proposed method's effectiveness and generalization capabilities. Ultimately, we assess the method's cross-problem domain proficiency by testing it on large-scale single-objective problems (LSSOP), large-scale multi-objective optimization problems (LSMOP), real-world problems (ratio error estimation of voltage transformers), large-scale constrained multi-objective optimization problems (LSCMOP), and discrete problems.

In relation to the hyper-parameter t, we consider both the algorithm's accuracy and its running time as objectives. Subsequently,

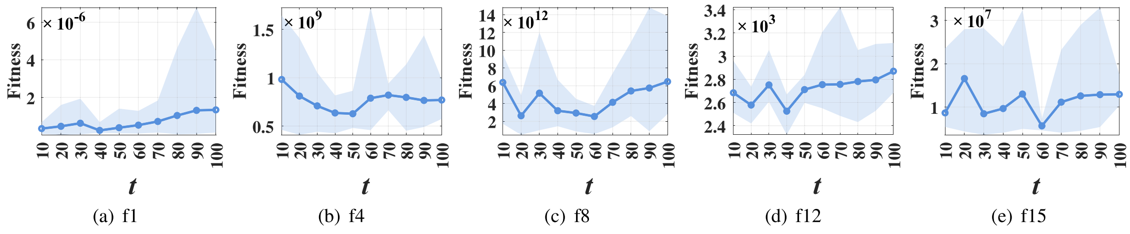


Fig. 7. Optimization results of NgISILDW-SHADE with 5 functions and different t values.

we calculate the Knee point of the Pareto front for these objectives, indicating the most suitable value of t. Further details regarding this process are provided in Section 5.1.

For NgISILDW, we set lr to 0.001, γ to 0.95, and e to 100. Given that the focus of our work does not center on neural network training—a domain with established methodologies, thus we adopted standard hyperparameter values for consistency and reproducibility, i.e. lr and γ are derived from SGD training method, while the e is empirically determined based on preliminary experimental observations. The parameter wl_{min} is set to 100 for optimizing LSMOP and LSCMOP, and to 10 for other problems. This decision is based on empirical evidence. Specifically, setting wl_{min} to 10 based on the fact that most EAs not tailored for large-scale attributes are validated with a minimum problem dimension of 10. We set wl_{min} to 100 for LSMOP and LSCMOP because their test sets require fewer function evaluations, and a smaller wl_{min} could hinder convergence. In addition, the t-test technique is used in the ranking algorithm. See our published code for more specific details.[2]

Note that for a deterministic experiment involving the hyper-parameter t (Section 5.1), ablation experiment (Section 5.2), and LSSOP experiment (Section 5.4), we incorporated NgISILDW into the SHADE algorithm [39], terming it NgISILDW-SHADE, to validate its effectiveness. SHADE assigns two parameters, F and CR, to each individual and dimension. During the evolution process, the evolution of these parameters is also constrained by the dimensional window method.

5.1. The appropriate hyper-parameter t

If the net is trained with an excessively small interval t, it will prevent the population from taking advantage of the "ability to see more windows", while also significantly hiking up the algorithm's operational costs. Conversely, an overly large interval t can lead to the selection of less valuable windows, thereby compromising the algorithm's performance. To pinpoint the most suitable t, we chose five functions with distinct characteristics from the LSSOP test suite CEC2013: f1, f4, f8, f12, and f15. We then observed the algorithm's performance and execution time at various t ranging from 10 to 100. Each function was independently run five times, the maximum number of function evaluations is set to 3E6, and the optimization results are depicted in Fig. 7.

A clear observation is the presence of a distinct inflection point, beyond which the algorithm's performance progressively deteriorates as t increases. Notably, the position of this inflection point differs across problems; for instance, the inflection point for f15 occurs later than that for f1, although most inflection points cluster between 40 and 60.

Regarding running time, as illustrated in Fig. 8(a), it is evident that a larger t results in fewer training times for the net, thereby reducing its execution time. However, there exists a conflict between running cost and algorithm performance, both of which are influenced by t. Constructing a PF that balances these two objectives is an effective strategy. As depicted in Fig. 8(b), we have plotted the normalized distributions of algorithm performance and running time for five functions, which exhibit a trend similar to the PF.

The special symbol markers in the plots denote the Knee points, which represent critical values of t; deviations from these points can lead to significant degradation in the algorithm's performance for a given objective. In this study, we employ a straightforward distance-based method to identify the Knee point. This approach calculates the distances between all points on the PF and boundary points and identifies the point with the largest distance as the Knee point. For the five functions considered, the Knee points correspond to t values of 40, 40, 60, 40, 60, and 60, respectively. To enhance the adaptability of NgISILDW to a broader range of problems, we have selected a value of t equal to 40.

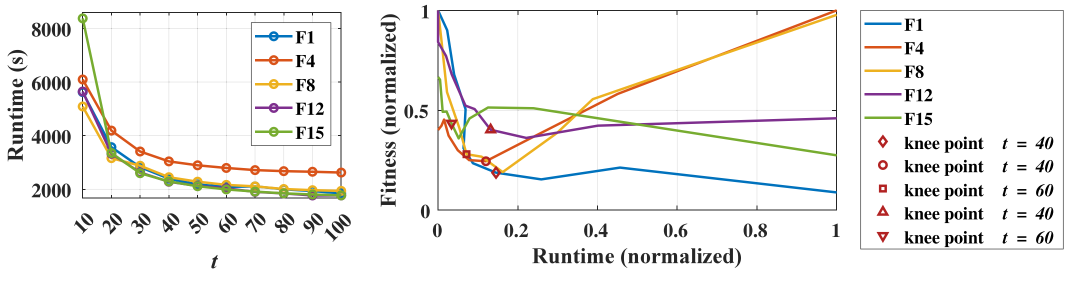
5.2. Ablation

In this paper, we initially introduce the dimensional window method. Given its proven effectiveness, we further propose the individual-level window, the isometric segmentation window length mechanism, and the neural network-guided window element strategy. To validate the efficacy of these approaches/strategies, we embedded them into SHADE and assessed their optimization accuracy on the CEC2013 benchmark [40]. Each algorithm independently optimized each problem 25 times, the maximum number of function evaluations is set to 3E6, and the mean and variance of these outcomes are detailed in Table 4. Specifically, Window and ILDW represent the proposed dimensional window method and individual-level dimensional window method, respectively. NgILDW is an enhanced version of ILDW that incorporates the neural network-guided window element strategy, while NgISILDW builds upon NgILDW by integrating the isometric segmentation window length mechanism, i.e., an improved version of the dimensional window method.

The advantage of DW-SHADE over plain SHADE is substantial, surpassing it by eleven orders of magnitude on f1, three on f12, two on f10, and one on f2, f3, and f7. This confirms the effectiveness of the dimensional window method introduced in this paper. Furthermore, the individual-level dimensional window method demonstrates a notable edge over the standard dimensional window method, trailing behind on only three problems but surpassing it by one order of magnitude on f1, f3, f8, and f14. NgILDW-SHADE, while slightly inferior to ILDW-SHADE overall ranked third, while ILDW-SHADE ranked second. We speculate this is due to the fact that the neural networks guide the algorithm towards "valuable windows", somewhat neglecting the ability to "explore more windows". The final improved version NgISILDW-SHADE emerges as the victor, demonstrating optimal performance across six benchmark functions (f4, f7, f9, f11, f12, and f14). Statistical validation via Friedman's test underscores its dominance, it combines the strengths of all the components and ranks first, clearly outperforming all the other versions.

Fig. 9 illustrates the population distribution across 500 iterations of the original SHADE algorithm and algorithms embedded with different versions of the DW method. These algorithms are employed to optimize the CEC2013-F2 and CEC2013-F4 benchmark functions. Given that the problem dimension is set at 1000, each individual within the population is represented as a 1000-dimensional vector. By connecting

[2] https://github.com/sunyafeng1996/WindowMethod

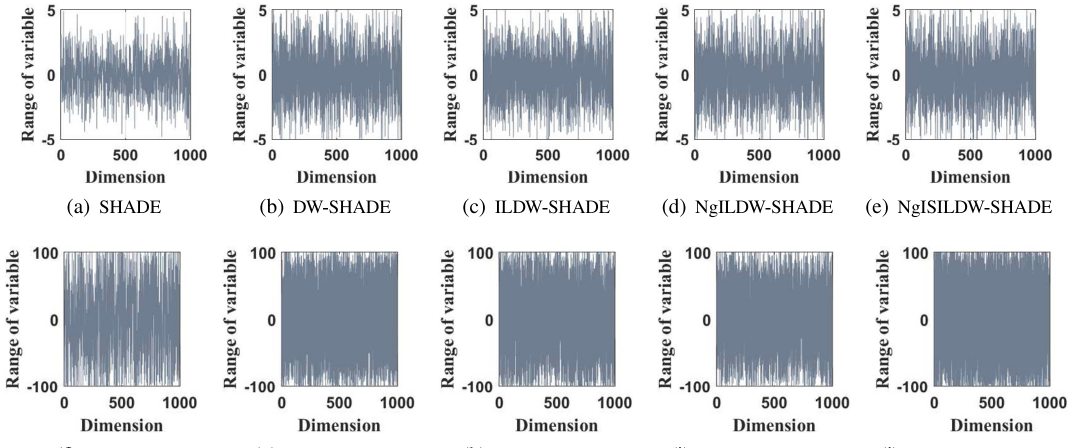


(a) Runtime (b) Knee point

Fig. 8. Running time of NgISILDW-SHADE and its Knee point with optimization performance.

Table 4
Mean(variance) results of dimensional window methods by version on CEC2013.

Function	SHADE	DW-SHADE	ILDW-SHADE	NgILDW-SHADE	NgISILDW-SHADE
f1	5.89E+06(1.04E+07)	4.62E−5(5.03E−5)	6.67E−7(2.16E−6)	**5.28E−7(6.60E−7)**	5.99E−7(1.68E−6)
f2	1.70E+04(1.03E+03)	4.08E+3(2.78E+2)	3.89E+3(2.97E+2)	**3.83E+3(2.71E+2)**	3.92E+3(2.40E+2)
f3	1.54E+01(3.00E−01)	8.97E+0(6.35E−1)	**7.97E+0(4.27E−1)**	8.30E+0(4.46E−1)	8.12E+0(6.12E−1)
f4	1.69E+09(6.75E+08)	1.03E+9(3.12E+8)	9.13E+8(3.25E+8)	8.37E+8(2.97E+8)	**8.23E+8(2.54E+8)**
f5	3.09E+06(6.83E+05)	3.55E+6(4.31E+5)	**2.65E+6(4.45E+5)**	3.086E+6(7.05E+5)	2.88E+6(4.03E+5)
f6	1.01E+05(1.63E+04)	5.49E+3(9.37E+1)	5.28E+3(1.10E+3)	**4.42E+3(2.24E+3)**	5.02E+3(1.51E+3)
f7	1.70E+07(3.19E+07)	2.27E+6(6.43E+5)	1.94E+6(5.74E+5)	2.26E+6(1.16E+6)	**1.88E+6(9.62E+5)**
f8	**6.78E+11(4.80E+11)**	2.40E+13(1.57E+13)	4.03E+12(3.27E+12)	4.04E+12(3.02E+12)	3.66E+12(2.36E+12)
f9	2.99E+08(3.29E+07)	3.07E+8(2.66E+7)	2.14E+8(3.51E+7)	2.25E+8(4.82E+7)	**2.07E+8(2.92E+7)**
f10	1.22E+06(1.50E+04)	**1.36E+4(1.98E+4)**	1.87E+4(2.19E+4)	2.57E+4(2.27E+4)	1.50E+4(2.03E+4)
f11	1.10E+08(6.69E+07)	1.27E+8(3.57E+7)	1.14E+8(3.89E+7)	1.14E+8(5.33E+7)	**1.01E+8(3.25E+7)**
f12	3.83E+06(7.96E+06)	2.83E+3(1.99E+2)	2.67E+3(1.84E+2)	2.71E+3(1.73E+2)	**2.65E+3(1.22E+2)**
f13	**9.48E+07(9.39E+07)**	1.29E+8(6.90E+7)	1.34E+8(7.08E+7)	1.15E+8(4.98E+7)	1.04E+8(4.72E+7)
f14	1.53E+08(2.85E+08)	1.14E+8(1.86E+8)	5.83E+7(3.78E+7)	6.69E+7(2.86E+7)	**5.47E+7(2.53E+7)**
f15	**2.18E+06(2.83E+05)**	4.52E+6(5.39E+5)	1.17E+7(1.74E+7)	8.18E+6(6.52E+6)	7.13E+6(4.10E+6)
+/−/=	9 + 2 − 4 =	9 + 1 − 5 =	5 + 2 − 8 =	8 + 1 − 6 =	
Rank	5	4	2	3	1



(a) SHADE (b) DW-SHADE (c) ILDW-SHADE (d) NgILDW-SHADE (e) NgISILDW-SHADE
(f) SHADE (g) DW-SHADE (h) ILDW-SHADE (i) NgILDW-SHADE (j) NgISILDW-SHADE

Fig. 9. Population distributions at generation 500 for each version of DW method on CEC2013-F2 (a–e) and CEC2013-F4 (f–j) problems.

the values of these dimensions, we generate visual representations of the individuals. Each subplot in the figure comprises 100 such lines, each symbolizing an individual, thereby providing a comprehensive view of the population distribution.

In these visualizations, the degree of clustering among the lines serves as an indicator of the population's compactness. A higher concentration of lines implies a more compact distribution, whereas a more dispersed arrangement suggests a sparser, and consequently, a more exploratory population. Notably, SHADE exhibits the most compact distribution for both f2 and f4 functions. In contrast, the DW-SHADE variants display a notably sparser distribution, with NgISILDW-SHADE being the most sparse among them. This observation suggests that the effectiveness of the DW method and its improved versions may stem from their ability to prolong the exploitation phase of the population. By delaying the convergence, these algorithms prioritize exploration early on, fostering a more diverse and exploratory population. Furthermore, this finding underscores the importance of avoiding premature convergence in large-scale optimization problems, as it can hinder the discovery of optimal solutions.

In summary, the dimensional window method presented in this paper significantly enhances the accuracy of EA on large-scale problems, and all strategies discussed are beneficial for such problems. This section validates the utility of the dimensional window method and its improved variant, NgISILDW, which significantly surpasses the original dimensional window method. Therefore, in subsequent experiments, only the impact of NgISILDW will be demonstrated.

Table 5
Mean results of 5 EAs and their embedded NgISILDW versions on CEC2013.

Function	DE	NgISILDW -DE	MVPA	NgISILDW -MVPA	PSO	NgISILDW -PSO	IMODE	NgISILDW -IMODE	OFA	NgISILDW -OFA
f1	4.84E+06	**8.61E+03**	6.75E+10	**3.84E+10**	1.97E+11	**1.60E+11**	2.19E+06	**8.57E+03**	2.58E+06	**1.87E+06**
f2	1.61E+04	**1.31E+04**	4.10E+04	**3.89E+04**	6.37E+04	**5.09E+04**	7.80E+03	**2.31E+03**	1.69E+04	**1.49E+04**
f3	1.99E+01	**3.62E+00**	**2.10E+01**	**2.10E+01**	**2.11E+01**	**2.11E+01**	9.82E+00	**6.82E+00**	1.89E+01	**1.67E+01**
f4	**4.63E+09**	9.58E+10	1.27E+12	**1.03E+12**	3.64E+12	**3.62E+12**	**7.88E+09**	7.90E+09	2.06E+11	**1.54E+11**
f5	9.07E+06	**6.78E+06**	1.73E+07	**1.51E+07**	2.94E+07	**2.57E+07**	**6.22E+06**	7.16E+06	**1.06E+07**	1.11E+07
f6	8.19E+03	**2.56E+01**	8.79E+05	**8.64E+05**	**9.78E+05**	9.93E+05	1.08E+05	**7.51E+04**	**2.27E+01**	2.34E+01
f7	**1.37E+08**	7.51E+08	8.88E+11	**1.97E+11**	6.57E+13	**2.46E+13**	1.13E+08	**5.80E+07**	6.36E+09	**2.50E+09**
f8	**2.48E+11**	3.67E+14	1.59E+16	**1.51E+16**	**1.17E+17**	1.57E+17	**3.35E+13**	1.63E+14	2.40E+15	**1.77E+15**
f9	7.15E+08	**5.44E+08**	1.35E+09	**1.13E+09**	2.21E+09	**2.05E+09**	**4.21E+08**	5.04E+08	8.73E+08	**8.37E+08**
f10	3.85E+04	**1.45E+03**	**1.35E+07**	1.65E+07	**7.82E+07**	8.16E+07	4.48E+04	**1.95E+04**	**2.60E+03**	6.85E+05
f11	**1.91E+09**	5.76E+10	2.82E+13	**8.41E+12**	2.87E+15	**1.18E+15**	2.83E+09	**2.15E+09**	1.68E+11	**8.09E+10**
f12	3.34E+08	**1.61E+06**	1.59E+12	**1.36E+12**	3.93E+12	**2.52E+12**	6.93E+07	**6.86E+04**	2.68E+10	**1.74E+09**
f13	**2.36E+09**	2.11E+10	1.71E+13	**2.79E+12**	2.54E+15	**1.08E+15**	1.39E+09	**1.23E+09**	7.36E+10	**2.86E+10**
f14	**6.61E+09**	8.00E+10	3.29E+13	**9.30E+12**	5.08E+15	**1.64E+15**	1.99E+10	**1.98E+10**	9.08E+11	**3.08E+11**
f15	4.16E+08	**6.59E+07**	1.70E+14	**4.18E+13**	5.63E+15	**2.06E+15**	**4.13E+06**	9.66E+06	7.87E+07	**4.00E+07**

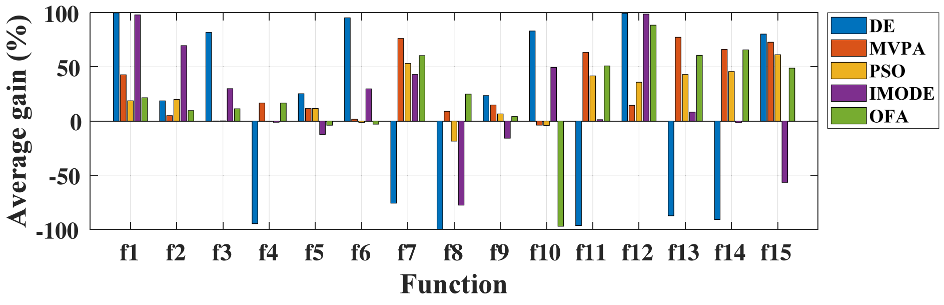


Fig. 10. Average gain of NgISILDW over 5 EAs.

5.3. Generalizability

To ascertain the versatility of NgISILDW in supporting a diverse array of EAs, this section integrates it into five prevalent EAs: DE [41], PSO [42], IMODE [43], MVPA [44], and OFA [45]. The objective is to validate its generalizability. The results of these EAs, both in their original and NgISILDW-embedded forms, on the CEC2013 benchmark are presented in Table 5. The experimental configurations adhere to those outlined in Section 5.2.

Upon incorporating NgISILDW, the baseline EAs exhibit a noticeable enhancement in their performance in solving LGSO problems. Across the five algorithms, the number of functions that benefit from this integration is 9, 13, 12, 11, and 12, respectively. The estimated efficiency of each EA, post-integration, stands at approximately 60%, 87%, 80%, 73%, and 80%. To delve deeper into the magnitude of performance improvement that NgISILDW offers when generalized to these five algorithms, we devised the average gain metric:

$$ag^j = \frac{1}{IR} \cdot \sum_{i=1}^{IR} \left(\frac{RE_i - RN_i}{REN_i}, REN_i = \begin{cases} RE_i, & \text{if } RE_i > RN_i \\ RN_i, & \text{otherwise} \end{cases}\right) \tag{26}$$

Here, ag_i represents the average gain on the jth problem, IR denotes the number of independent runs, RE signifies the results of the original EA, and RN corresponds to the results of the EA embedded with NgISILDW. Eq. (26) essentially encapsulates the proportion of gain that NgISILDW contributes to the EA. If the EA experiences an improvement, ag will be a positive number, with a higher value indicating a greater boost, up to a maximum of 1. Conversely, if there is a decline in performance, ag will be a negative number, with a lower value reflecting a steeper drop, down to a minimum of -1.

Fig. 10 illustrates the average gain of NgISILDW across the five EAs. The results depicted in the figure reveal that DE exhibits the most radical variations, with both the highest positive and negative average gains. All other methods maintain a consistently good average gain. In total, the graph comprises 75 bars, each representing the average gain of one of the 5 algorithms on one of the 15 functions from the CEC2013 benchmark when integrated with NgISILDW. 57 of these bars indicate a positive gain. Therefore, the generalization efficiency of NgISILDW can be roughly approximated as 57/75 = 76%. While NgISILDW may not be effective for all problems, it consistently guarantees a substantial performance enhancement for all algorithms on the test suite.

5.4. Comparison on LSSOP

In the LSSOP setting, we conduct a comparative analysis of NgISILDW-SHADE against eight advanced large-scale single-objective EAs on the CEC2013 benchmark. Among these, DECC-MDG [17], DECC-EADG [46], CCFR-IDG2 [47], and CCBC-DG2 [48] belong to the category of CC-based methods, whereas SLPSO-ARS [49], RLLPSO [50], HCLPSO [51], and SDLPSO [52] represent methodologies that have been specifically tailored to accommodate large-scale properties within a given EA. The experimental configurations employed in this comparison mirror those detailed in Section 5.2, and the comparative outcomes are presented in Table 6.

For functions f13 and f14, the dimensions are set to 905 for the CEC2013 benchmark and 1000 for other problems. In summary, despite NgISILDW's emphasis on generalizability, it enables NgISILDW-SHADE to attain optimal results and secure the top ranking. When considering the types of problems, NgISILDW demonstrates a more pronounced advantage in handling partially separable functions (f4–f11). However, on completely inseparable problems (f15), its performance lags behind the five compared algorithms, likely due to the insufficient precision in controlling the wl mechanism. Overall, NgISILDW-SHADE remains highly competitive and validates the efficacy of NgISILDW in solving LSSOP.

5.5. Comparison on LSMOP

In the context of the LSMOP setting, we utilize the LSMOP benchmark [53] to integrate NgISILDW with four established MOEAs:

Table 6
Mean(variance) results of 8 LSSO EAs and NgISILDW-SHADE on CEC2013.

Problem	DECC-MDG	DECC-EADG	SLPSO-ARS	RLLPSO	HCLPSO	SDLPSO	CCFR-IDG2	CCBC-DG2	NgISILDW-SHADE
f1	1.08E+00(1.5E+00)	1.02E+06(2.3E+06)	7.34E−19(5.0E−20)	**9.01E−22(3.50E−22)**	4.46E−21(7.3E−22)	4.32E−23(1.5E−23)	2.00E−05(5.0E−6)	8.65E+05(1.2E+06)	5.99E−07 (1.7E−06)
f2	1.21E+04(4.5E+02)	1.44E+04(1.6E+03)	2.79E+03(2.9E+02)	1.04E+03(7.4E+01)	7.22E+02(3.2E+01)	9.93E+02(5.6E+01)	**3.60E+02(2.0E+01)**	1.41E+04(1.5E+03)	3.92E+03 (2.4E+02)
f3	2.14E+01(1.9E−02)	2.06E+01(9.1E−03)	2.01E+01(2.0E−02)	2.16E+01(6.2E−03)	2.16E+01(7.6E−03)	2.16E+01(4.9E−03)	2.10E+01(1.0E−02)	2.06E+01(8.6E−03)	**8.16E+00 (6.1E−01)**
f4	3.68E+10(1.3E+10)	2.97E+08(1.5E+08)	1.02E+10(2.7E+09)	4.44E+09(5.0E+08)	8.48E+09(2.2E+09)	2.95E+09(5.4E+08)	9.60E+07(4.0E+07)	**3.39E+07(1.8E+07)**	8.23E+08 (2.5E+08)
f5	4.81E+06(6.9E+05)	2.27E+06(3.01E+05)	7.60E+05(1.4E+05)	7.25E+05(1.0E+05)	5.68E+05(8.4E+04)	6.69E+05(1.0E+05)	2.80E+06(3.0E+05)	**2.14E+06(4.2E+05)**	2.88E+06 (4.0E+05)
f6	1.06E+06(2.1E+03)	1.06E+06(2.2E+03)	1.03E+06(5.2E+03)	1.06E+06(9.9E+02)	1.06E+06(9.6E+02)	1.06E+06(1.1E+03)	1.10E+06(1.0E+03)	1.05E+06(3.4E+03)	**5.02E+03 (1.5E+03)**
f7	6.57E+07(2.8E+07)	6.10E+05(2.7E+06)	7.27E+06(3.4E+06)	9.84E+05(2.5E+05)	1.85E+06(6.7E+05)	**3.47E+05(3.1E+05)**	2.00E+07(3.0E+07)	2.95E+07(2.8E+07)	1.88E+06 (9.6E+05)
f8	3.68E+15(9.7E+14)	9.20E+13(5.4E+13)	1.59E+14(3.8E+13)	8.56E+13(1.1E+13)	1.41E+14(2.9E+13)	4.91E+13(1.6E+13)	7.00E+10(1.0E+11)	**6.74E+10(1.2E+11)**	3.66E+12 (2.4E+12)
f9	4.95E+08(5.1E+07)	2.67E+08(7.5E+07)	4.84E+07(9.2E+06)	**3.85E+07(8.1E+06)**	8.49E+07(2.5E+07)	4.07E+07(7.0E+06)	1.90E+08(3.0E+07)	1.70E+08(3.2E+07)	2.07E+08 (2.9E+07)
f10	9.41E+07(1.8E+05)	9.43E+07(2.7E+05)	9.20E+07(4.1E+05)	9.40E+07(2.2E+05)	9.24E+07(1.7E+06)	9.41E+07(1.8E+05)	9.50E+07(2.0E+05)	9.28E+07(7.2E+05)	**1.50E+04 (2.0E+04)**
f11	2.16E+09(7.0E+08)	1.58E+10(3.6E+10)	1.53E+09(5.2E+09)	9.25E+11(8.7E+09)	9.28E+11(9.9E+09)	1.74E+08(6.5E+07)	4.00E+08(3.0E+08)	7.70E+08(3.5E+08)	**1.01E+08 (3.2E+07)**
f12	3.66E+03(4.0E+02)	4.44E+07(1.8E+08)	**1.06E+03(8.5E+01)**	1.87E+03(1.7E+02)	1.20E+03(9.8E+01)	1.27E+03(1.2E+02)	1.60E+09(2.0E+09)	5.81E+07(7.8E+07)	2.65E+03 (1.2E+02)
f13	1.09E+09(3.3E+08)	5.29E+08(2.2E+08)	3.66E+08(2.6E+08)	2.17E+08(5.5E+07)	3.76E+08(1.6E+08)	2.54E+08(1.1E+08)	1.20E+09(6.0E+08)	6.03E+08(1.4E+08)	**1.04E+08 (4.7E+07)**
f14	4.23E+09(4.7E+09)	6.32E+08(7.7E+08)	1.83E+09(1.3E+09)	**5.06E+07(2.2E+07)**	1.00E+09(6.9E+08)	6.91E+07(4.8E+07)	3.40E+09(3.0E+09)	1.11E+09(1.1E+09)	5.47E+07 (2.5E+07)
f15	1.00E+07(1.6E+06)	5.27E+06(1.7E+06)	7.25E+06(6.4E+05)	**1.63E+06(5.9E+04)**	9.62E+06(7.4E+05)	1.19E+07 (8.0E+05)	9.80E+06(4.0E+06)	7.11E+06(1.4E+06)	7.13E+06 (4.1E+06)
+/−/=	15 + 0 − 0 =	10 + 4 − 1 =	8 + 4 − 3 =	7 + 6 − 2 =	9 + 4 − 2 =	8 + 5 − 2 =	10 + 4 − 1 =	10 + 4 − 1 =	
Rank	9	7	4	3	6	2	8	5	1

Table 7
Mean(variance) of IGD metric for 8 LSMO EAs on LSMOP benchmark.

M	Problem	FLEA	LMEA	LSMOF	MOCGDE	NgISILDW-GDE3	NgISILDW-NSGAIII	NgISILDW-MOEAD	NgISILDW-MOEADDE
2	LSMOP1	5.7E−01(9.8E−02)	8.2E+00(3.1E+00)	5.4E−01(4.2E−02)	**1.2E−01(5.5E−02)**	1.9E−01(6.6E−02)	6.9E−01(3.4E−02)	4.1E−01(1.9E−01)	1.8E−01(4.7E−02)
	LSMOP2	**4.4E−02(3.1E−03)**	1.5E−01(1.1E−02)	3.7E−02(2.2E−03)	1.5E−01(2.0E−03)	7.8E−02(9.3E−03)	8.4E−02(3.7E−03)	1.1E−01(3.7E−03)	8.5E−02(1.4E−03)
	LSMOP3	1.6E+00(2.2E−03)	1.1E+03(1.3E+03)	**1.5E+00(4.4E−02)**	1.2E+01(1.0E+01)	2.2E+00(4.8E−01)	1.0E+01(3.3E+00)	6.9E−01(7.2E−02)	7.1E−01(6.1E−04)
	LSMOP4	1.1E−01(6.8E−03)	2.6E−01(2.2E−02)	1.0E−01(6.2E−03)	2.4E−01(6.9E−03)	**8.6E−02(3.6E−03)**	1.5E−01(3.2E−03)	1.6E−01(5.9E−03)	7.5E−02(9.0E−03)
	LSMOP5	7.4E−01(2.3E−16)	9.2E+00(1.0E+01)	7.4E−01(2.3E−16)	2.1E−01(9.3E−02)	**1.1E−01(1.7E−02)**	6.0E−01(1.9E−01)	5.4E−01(1.6E−01)	1.8E−01(1.0E−01)
	LSMOP6	**2.6E−01(1.2E−01)**	1.3E+03(2.2E+03)	3.6E−01(2.2E−03)	7.2E−01(6.3E−02)	5.4E−01(9.8E−02)	8.5E−01(7.5E−02)	7.7E−01(3.3E−02)	7.5E−01(1.6E−03)
	LSMOP7	1.5E+00(3.0E−04)	1.6E+04(3.1E+04)	**1.5E+00(3.5E−03)**	2.2E+04(1.9E+04)	3.5E+00(4.7E−01)	2.6E+00(2.1E−01)	2.2E+00(5.5E−01)	1.8E+00(3.9E−01)
	LSMOP8	7.4E−01(2.3E−16)	4.1E+00(7.5E+00)	7.4E−01(2.3E−16)	2.0E−01(7.1E−02)	**1.3E−01(1.0E−02)**	4.5E−01(1.5E−01)	2.7E−01(2.2E−01)	6.5E−02(2.5E−03)
	LSMOP9	**7.8E−01(1.3E−01)**	4.1E+01(1.9E+01)	8.1E−01(0.0E+00)	9.9E−01(6.5E−01)	8.1E−01(3.9E−03)	8.1E−01(1.6E−05)	4.0E−01(1.7E−02)	3.9E−01(7.8E−03)
	Rank	8	5	3	7	1	4	6	2
		5.75				3.25			
3	LSMOP1	6.6E−01(1.7E−01)	9.4E+00(2.7E+00)	2.5E−01(1.6E−02)	**8.7E−02(3.8E−02)**	1.9E+00(2.2E+00)	4.7E−01(7.4E−02)	2.4E−01(3.2E−02)	3.0E−01(1.1E−02)
	LSMOP2	9.8E−02(7.0E−03)	1.0E−01(5.2E−03)	1.2E−01(4.8E−03)	**7.3E−02(3.8E−03)**	1.0E−01(1.2E−03)	7.7E−02(9.3E−04)	8.6E−02(9.0E−04)	8.6E−02(1.1E−03)
	LSMOP3	8.8E−01(1.0E−01)	4.5E+02(4.5E+02)	6.5E−01(2.2E−02)	1.6E+01(3.7E+01)	7.2E+00(6.0E−01)	2.0E+00(6.9E−01)	**6.5E−01(1.5E−01)**	8.3E−01(4.7E−02)
	LSMOP4	2.1E−01(1.1E−02)	3.1E−01(8.6E−03)	2.8E−01(9.1E−03)	**2.0E−01(3.3E−02)**	3.0E−01(4.9E−03)	1.9E−01(3.6E−03)	2.4E−01(6.5E−03)	1.9E−01(6.7E−03)
	LSMOP5	9.9E−01(2.8E−01)	1.4E+01(2.6E+00)	5.1E−01(3.0E−02)	**2.7E−01(1.2E−01)**	3.9E−01(4.3E−02)	1.9E+00(5.8E−01)	7.6E−01(1.4E−01)	4.1E−01(1.0E−01)
	LSMOP6	1.2E+00(2.7E−01)	1.7E+04(1.8E+04)	**6.9E−01(5.8E−02)**	9.6E+02(3.5E+03)	2.8E+00(1.3E+00)	1.7E+00(2.2E−01)	1.8E+00(6.9E−01)	1.6E+00(5.5E−01)
	LSMOP7	9.7E−01(1.3E−01)	5.2E+02(1.7E+03)	9.4E−01(1.3E−02)	1.2E+01(4.3E+01)	1.1E+00(3.8E−02)	1.6E+00(2.2E−02)	**7.2E−01(9.0E−02)**	9.5E−01(9.7E−05)
	LSMOP8	5.1E−01(9.3E−02)	4.6E−01(3.1E−01)	3.4E−01(4.4E−02)	**1.0E−01(2.6E−03)**	2.5E−01(1.4E−02)	3.3E−01(3.2E−02)	7.0E−01(7.7E−02)	3.2E−01(5.4E−02)
	LSMOP9	1.5E+00(2.3E−01)	8.4E+01(5.7E+01)	1.5E+00(4.6E−16)	2.7E+00(6.5E−01)	**1.8E+00(1.9E−01)**	2.2E+00(3.7E−01)	4.1E−01(7.4E−03)	7.4E−01(2.7E−01)
	Rank	7	8	4	1	6	5	3	2
		5				4			

GDE3 [54], NSGA-III [55], MOEAD [56], and MOEADDE [57]. This integration is compared against four advanced LSMOEAs: FLEA [58], LMEA [59], LSMOF [60], and MOCGDE [60]. Notably, the experimental configurations of these LSMOEAs exhibit considerable variation. For instance, FLEA is configured 8 times the dimensionality of function evaluations and tested on problems with dimensions ranging from 5000 to 10000. Conversely, LMEA is set up with a maximum number of function evaluations of 1E6 for 100-dimensional problems and 6.8E6 for 500-dimensional problems.

To ensure fairness, we configure the experimental parameters of all algorithms according to the guidelines provided in the benchmark's demonstration. In the LSMOP benchmark, the test problems can be configured with two or three objectives, corresponding to dimensions of 200 and 300, respectively. The termination conditions for the algorithms are set to 2000 and 3000 iterations for the two-objective and three-objective problems, respectively. However, using iterations as the termination condition may not be equitable; therefore, we calculate that the maximum number of function evaluations for the LSMOP benchmark is 200,000 for the two-objective problem and 273,000 for the three-objective problem. Each problem is independently run 20 times, and Table 7 presents the mean and variance of the IGD metrics.

In dual-objective LSMOPs, the four algorithms enhanced by NgISILDW secured 1st, 2nd, 4th, and 6th ranks respectively, demonstrating significant superiority over the other four LSMO EAs. For three-objective cases, NgISILDW-enhanced algorithms achieved an average rank of 4, outperforming the comparison LSMO EAs (average rank: 5), though with a less pronounced advantage than in dual-objective scenarios. These results collectively affirm the plugin's strong generalization capability on LSMOPs, yielding two observations: (1) NgISILDW's effectiveness on complex LSMOPs is attenuated compared to simpler cases, indicating the potential for plugin refinement. (2) NgISILDW helps multi-objective EAs rival or even surpass those specifically designed for LSMOP because valuable perturbations are still achieved in multi-objective cases.

5.6. Comparison on LSCMOP

To assess the effectiveness of NgISILDW on constrained problems, this section integrates it into IMTCMO and compares it with ICMA [61], IMTCMO [62], EMCMO [63], POCEA [64], DPSEA [65], CMOCSO [66], and IMTCMO-BS [67]. The test suite utilized is LSCM [67], and we adhere to its experimental setup by setting the maximum number of evaluations to 5000 times the dimensionality. Each algorithm is independently run 21 times, and Table 8 presents the mean and variance of the IGD metric. The symbol '-' in Table 8 indicates that the algorithm failed to find a feasible solution for the given problem.

Very consistently, NgISILDW-IMTCMO achieves the optimum on two problems in either dimension. While it does not surpass the state-of-the-art method IMTCMO-BS overall, NgISILDW-IMTCMO significantly outperforms the third-ranked IMTCMO, showcasing its competitiveness in the LSCMOP benchmark.

5.7. Comparison on real-world problem

Validating the proposed dimensional windowing method through real-world practical application problems is an indispensable step to substantiate its effectiveness. In this section, we employ the ratio error estimation (TREE) problem [68] for this purpose. The TREE problem, formulated by He et al. in 2020, is a well-crafted model that transforms the computationally demanding task of ratio error estimation into a relatively cost-effective optimization problem.

It falls under the umbrella of multi-objective large-scale constrained optimization problems, which are notoriously challenging due to their high dimensionality and the presence of multiple conflicting objectives and constraints. To assess the performance of our method, we integrate the NgISILDW approach into the IMTCMO-BS algorithm. We then put this enhanced algorithm against five advanced EAs: DPVAPS [69], DSSEA [70], DVCEA [71], IMCMOEAD [72], and LCMEA [73]. To

Table 8
Mean(variance)of IGD metric for 8 LSCMO EAs on LSCM benchmark.

500D								
Problem	ICMA	IMTCMO	EMCMO	POCEA	DPSEA	CMOCSO	IMTCMO-BS	NgISILDW-IMTCMO
LSCM1	8.38E−01(1.47E−1)	7.90E−01(4.67E−2)	1.04E+00(5.92E−2)	2.49E+00(5.34E+0)	1.04E+00(4.86E−2)	7.71E−01(4.55E−2)	**5.92E−01(5.86E−2)**	8.72E−01(6.05E−2)
LSCM2	1.73E+03(1.22E+3)	2.58E+02(3.89E+2)	5.79E+03(2.33E+3)	6.17E+03(5.50E+3)	6.22E+02(5.99E+2)	8.86E+03(6.78E+3)	**6.03E+01(1.95E+2)**	4.35E+02(4.60E+2)
LSCM3	5.94E+00(2.44E+0)	6.11E+00(1.56E+0)	7.52E+00(3.04E+0)	–	7.26E+00(2.08E+0)	2.36E+02(1.95E+2)	4.94E+00(2.33E+0)	**4.81E+00(1.54E+0)**
LSCM4	1.77E+03(1.61E+3)	5.99E+01(1.81E+2)	3.16E+03(1.19E+3)	–	8.95E+01(1.17E+2)	5.18E+03(4.45E+3)	5.19E+01(5.55E+1)	**6.36E+00(1.30E+0)**
LSCM5	3.74E+01(4.63E+0)	1.58E+01(3.87E+0)	1.26E+01(2.62E+0)	1.80E+02(3.79E+1)	**1.09E+01(1.80E+0)**	7.40E+01(3.48E+1)	1.36E+01(6.22E+0)	1.13E+01(2.13E+0)
LSCM6	7.95E−01(2.12E−1)	4.71E−01(1.19E−1)	7.36E−01(2.40E−1)	4.44E+00(7.74E+0)	6.90E−01(2.10E−1)	1.94E+00(1.68E+0)	**3.19E−01(4.94E−2)**	5.25E−01(1.28E−1)
LSCM7	5.29E+02(2.81E+2)	7.58E+01(1.55E+2)	6.30E+02(3.19E+2)	–	2.96E+02(2.53E+2)	3.61E+02(2.37E+2)	**2.65E+00(1.10E+0)**	6.68E+01(1.72E+2)
LSCM8	7.15E+00(1.25E+1)	3.15E+00(7.87E−1)	5.32E+00(0.00E+0)	–	–	2.63E+02(1.04E+2)	**2.70E+00(7.90E−1)**	3.79E+00(1.02E+0)
LSCM9	4.06E+02(3.53E+2)	4.10E+01(1.27E+2)	1.38E+03(5.41E+2)	1.27E+03(1.11E+3)	2.12E+02(1.93E+2)	8.27E+02(8.45E+2)	**8.56E−01(2.60E−1)**	8.98E−01(1.27E−1)
LSCM10	**1.68E+00(4.89E−1)**	3.30E+00(1.43E+0)	2.68E+00(1.47E+0)	–	2.75E+00(1.45E+0)	1.16E+01(2.52E+1)	5.07E+00(2.11E+0)	2.56E+00(2.55E−1)
LSCM11	1.79E+03(1.27E+3)	**6.39E+02(5.23E+2)**	6.63E+03(3.26E+3)	–	–	–	9.26E+02(8.24E+2)	1.61E+03(4.39E+2)
LSCM12	3.11E+03(3.04E+3)	1.58E+03(1.76E+3)	4.40E+03(9.15E+2)	2.74E+05(5.63E+5)	1.54E+03(0.00E+0)	1.62E+04(1.12E+4)	**9.02E+02(7.99E+2)**	1.88E+03(1.37E+3)
+/-/=	6+1-5=	4+3-5=	10+0-2=	11+0-1=	8+0-4=	10+1-1=	2+6-4=	
Rank	5	3	6	8	4	7	1	2
1000D								
LSCM1	1.49E+00(1.61E−1)	1.57E+00(9.48E−2)	2.00E+00(6.66E−2)	1.74E+00(3.96E−1)	2.00E+00(8.93E−2)	1.51E+00(1.27E−1)	**1.00E+00(1.12E−1)**	1.69E+00(1.13E−1)
LSCM2	8.26E+03(5.11E+3)	4.66E+02(5.03E+2)	1.40E+04(0.00E+0)	1.75E+04(8.35E+3)	–	–	**9.63E+01(2.24E+2)**	2.26E+03(3.16E+3)
LSCM3	1.41E+01(1.09E+1)	1.58E+01(3.68E+0)	1.15E+01(9.24E−1)	–	1.31E+01(3.18E+0)	2.95E+02(2.16E+2)	**1.16E+01(4.45E+0)**	1.71E+01(4.21E+0)
LSCM4	3.92E+03(2.63E+3)	5.94E+02(1.31E+3)	6.36E+03(2.11E+3)	–	1.56E+03(0.00E+0)	9.04E+03(0.00E+0)	**9.66E+01(7.27E+1)**	1.59E+03(2.69E+3)
LSCM5	8.22E+01(8.01E+0)	6.01E+01(4.15E+0)	**2.95E+01(4.25E+0)**	3.09E+02(6.92E+1)	3.07E+01(3.16E+0)	4.02E+02(2.20E+2)	5.84E+01(1.35E+1)	3.80E+01(5.35E+0)
LSCM6	1.36E+00(6.48E−1)	6.96E−01(1.23E−1)	8.94E−01(2.91E−1)	1.68E+01(1.95E+1)	7.83E−01(2.67E−1)	7.95E+00(7.97E+0)	**5.47E−01(1.43E−1)**	7.05E−01(1.78E−1)
LSCM7	1.38E+03(5.39E+2)	6.04E+01(1.30E+2)	2.36E+03(9.34E+2)	–	1.42E+03(4.38E+2)	3.28E+03(2.18E+3)	4.04E+02(1.39E+3)	**6.93E+01(1.57E+2)**
LSCM8	4.13E+01(9.95E+1)	9.56E+00(1.91E+0)	–	–	–	–	**7.73E+00(1.48E+0)**	9.54E+00(4.84E−1)
LSCM9	1.48E+03(1.31E+3)	6.17E+01(1.51E+2)	3.02E+03(1.15E+3)	–	3.02E+03(0.00E+0)	4.15E+03(4.48E+3)	**2.78E+00(3.97E+0)**	7.55E+01(1.71E+2)
LSCM10	**4.04E+00(1.48E+0)**	3.36E+01(1.91E+1)	6.37E+00(3.94E+0)	–	6.62E+00(2.76E+0)	3.72E+01(5.41E+1)	9.76E+00(3.21E+0)	6.02E+00(4.41E+0)
LSCM11	7.49E+03(2.23E+3)	5.50E+03(2.94E+3)	–	–	–	–	4.89E+03(2.37E+3)	**3.82E+03(0.00E+0)**
LSCM12	9.15E+03(4.17E+3)	**4.35E+03(4.90E+3)**	–	1.75E+06(2.87E+6)	–	9.32E+06(0.00E+0)	4.94E+03(2.95E+3)	6.50E+03(5.12E+3)
+/−/=	7 + 2 − 3 =	3 + 2 − 7 =	9 + 2 − 1 =	11 + 0 − 1 =	7 + 2 − 3 =	11 + 1 − 0 =	3 + 7 − 2 =	
Rank	4	3	6	8	5	7	1	2

Table 9
Mean(variance) of IGD metric for 6 EAs on TREE problems.

Problem	DPVAPS	DSSEA	DVCEA	IMCMOEAD	LCMEA	NgISILDW-IMTCMO-BS
TREE1	1.90E+1 (3.07E−2)	1.93E+1 (2.20E−2)	1.91E+1 (5.15E−2)	–	1.93E+1 (1.23E−3)	**4.08E+0 (8.48E+0)**
TREE2	8.72E+1 (6.89E−2)	8.73E+1 (1.47E−1)	8.71E+1 (1.70E−1)	8.36E+1 (2.50E−1)	8.69E+1 (1.18E−2)	**4.18E+1 (3.71E+1)**
TREE3	1.20E+1 (1.80E−4)	1.20E+1 (2.20E−4)	1.20E+1 (2.76E−4)	1.20E+1 (7.14E−3)	1.20E+1 (1.18E−6)	**4.05E−2 (0.00E+0)**
TREE4	2.40E+1 (1.27E−4)	2.40E+1 (3.75E−5)	2.40E+1 (7.22E−5)	2.40E+1 (9.33E−6)	2.40E+1 (1.05E−4)	**2.23E+1 (9.28E−1)**
TREE5	1.80E+1 (4.31E−2)	1.84E+1 (1.00E−2)	1.83E+1 (4.42E−2)	1.71E+1 (4.95E−2)	1.48E+1 (8.17E+0)	**7.54E+0 (9.82E+0)**
+/−/=	3/0/2	4/0/1	4/0/1	2/0/2	3/0/2	

ensure a fair and rigorous comparison, we maintained the maximum number of function evaluations consistent with the settings in the original text. Subsequently, we conducted a comprehensive statistical analysis of the results using the ANOVA test. The detailed outcomes of this analysis are presented in Table 9.

Table 9 employs five distinct configurations of the TREE problem as the objective functions for evaluation. Among these, TREE4 stands out with a dimension of 1200, while the remaining four problems each have a dimension of 600. Notably, when tackling the TREE1 problem, the IMCMOEAD algorithm failed to locate a feasible solution, so there are no available results for IMCMOEAD on this particular problem. In the context of these five real-world problems, the NgISILDW-IMTCMO-BS algorithm delivers remarkable outcomes. Numerically, it consistently outperforms the comparison algorithms across all five problems. From a statistical standpoint, the performance of NgISILDW-IMTCMO-BS is even more compelling. It surpasses the comparison algorithms on at least three of the problems and there is no instance where it ranks lower than the comparison algorithms on any problem. These results serve as compelling evidence of the efficacy of the proposed NgISILDW method. In particular, they highlight its exceptional capability to excel in real-world scenarios, demonstrating its practical value and robustness.

5.8. Comparison on discrete problem

NgISILDW can also enhance EAs in solving large-scale discrete problems. Table 10 presents the optimization results of GA and GA embedded with NgISILDW on three large-scale discrete problems, with a maximum number of function evaluations set to 2E5. These problems include the knapsack problem [74], which is a constrained binary-coded discrete problem; the MaxCut problem [75], which is an unconstrained binary-coded discrete problem; and the community detection problem [76], which is a label-coded discrete problem. The objective functions of these problems are adjusted, for instance, by multiplying by −1, to ensure that a smaller fitness value corresponds to a better solution.

As demonstrated in Table 10, NgISILDW-GA exhibits superior performance compared to GA for both the knapsack problem and the community detection problem, in terms of both mean and variance. For the MaxCut problem, NgISILDW-GA only slightly underperforms GA at D=5000. Those results underscore the effectiveness of NgISILDW in enhancing the performance of EAs on large-scale discrete problems.

6. Limitation

Section 5 underscores the great generality and cross-problem domain capability of the dimensional window method and its improved versions. However, it is imperative to acknowledge their notable shortcomings, as delineated below.

6.1. Incompatibility with CMAES

Based on our experimental observations, the dimensional window method and NgISILDW can substantially enhance the efficacy of most EAs in tackling large-scale problems, with the notable exception of CMAES [77] and CMAES-derived EAs.

Fig. 11 illustrates that upon incorporating NgISILDW, the convergence trajectory not only falls below that of CMAES but also exhibits abnormal patterns, failing to ensure that each successive generation's optimal individual is at least as good as its predecessor's. We speculate that CMAES leverages elite individuals within the population to fine-tune its distribution, whereas the dimensional window method disrupts this elite-guided distribution in CMAES. In contrast, the dimensional window method merely constrains the divergence between consecutive

Table 10
Mean(variance) of results for GA and NgISILDW-GA on discrete problems.

Problem	Knapsack problem			Max cut problem		
Type	Constrained binary			Binary		
D	1000	5000	10000	941	2344	5000
GA	1.93E+4 (5.60E+2)	1.15E+5 (2.01E+3)	2.44E+5 (1.61E+3)	−1.04E+3 (4.22E+1)	−2.50E+3 (2.73E+1)	**−3.67E+3** **(3.40E+1)**
NgISILDW-GA	**1.80E+4** **(2.93E+2)**	**1.12E+5** **(6.06E+2)**	**2.43E+5** **(1.28E+3)**	**−1.09E+3** **(9.79E+0)**	**−2.57E+3** **(2.31E+1)**	−3.61E+3 (1.07E+2)
Problem	Community detection problem				–	
Type	Label					
D	34	62	105	115		
GA	6.84E−1 (1.23E−1)	7.16E−1 (2.46E−2)	7.90E−1 (2.48E−2)	8.67E−1 (8.71E−3)		
NgISILDW-GA	**6.54E−1** **(1.61E−2)**	**7.08E−1** (2.29E−2)	**7.20E−1** **(0.00E+0)**	**8.42E−1** **(8.78E−3)**		

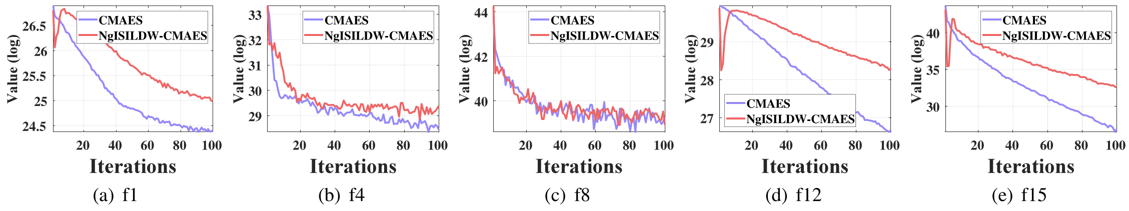


Fig. 11. Convergence trends of CEAES and NgISILDW-CMAES over the first hundred generations on five CEC2013 problems.

generations in other EAs. Consequently, we caution against integrating the dimensional window method and its advanced versions into CMAES or any EA that relies on CMAES as its foundational search mechanism.

6.2. Low cost-effectiveness of introducing neural network

The neural network in NgISILDW guides the population towards more promising dimensional windows, and while it introduces additional computational overhead, it effectively serves its intended purpose. To illustrate, we compare the time complexity of DE and NgISILDW-DE.

The time complexity of the DE algorithm is typically represented as $O(G \cdot NP \cdot D)$, where G denotes the number of iterations. The NgISILDW-DE operates differently. At each generation, it generates a dimensional window with randomly sized dimensions within equally spaced segments. The time complexity associated with this generation operation is approximately $O(NP)$. Additionally, every t generations, the neural network model undergoes training. This training process is computationally intensive, with a time complexity of $O(E \cdot NP \cdot C)$. Here, E refers to the number of training rounds per session, which is set at 100 in this paper, and C represents the dimension of the neural network's output data.

In the context of a single-objective, unconstrained problem, C is equal to $1 \cdot D$. The NP is 100, and the maximum number of function evaluations is 3E6. Consequently, the G is calculated as 3E6 / NP = 3E4. The t is set at 40, which means the neural network requires training a total of $G/t = 750$ times.

In summary, the time complexity of the DE is $O(3E6 \cdot D)$, while that of the NgISILDW-DE is $O(3E6 \cdot D) + O(7.5E6 \cdot D)$. From this analysis, it becomes evident that the time complexity of the NgISILDW-DE algorithm is roughly 3.5 times that of the DE algorithm. However, this does not necessarily imply that it is excessively time-consuming. In practical implementation, as demonstrated in Fig. 12, the actual time consumption of the NgISILDW-DE is not as pronounced, being approximately 1.3 times that of the DE, thanks to GPU acceleration technology.

Nevertheless, from a cost-performance perspective, the integration of the neural network may not have fully met our initial expectations. As depicted in Fig. 12, the incorporation of the neural network has resulted in a runtime increase for all algorithms, particularly for MVPA and IMODE, where the runtime has been prolonged by over three times on certain problems. However, in numerous instances, the performance enhancement has been modest, falling short of a 3-fold improvement. For instance, when MVPA tackles the f6 problem, it achieves a result of 8.79E+05, whereas NgISILDW-MVPA yields 8.64E+05, marking a mere 1.74% difference. Hence, we posit that the neural network's cost burden has not translated into the anticipated performance gains, and exploring alternative data-driven approaches for guiding the composition of window elements may offer superior cost-performance ratios.

6.3. Inapplicability to permutation optimization problems

Section 5.8 demonstrates that NgISILDW exhibits effectiveness across a range of discrete problems, except permutation optimization problems. As illustrated in Fig. 13 (a small-scale TSP problem is used as an example for the sake of demonstration), while the GA readily identifies a plausible solution for a TSP scenario with 30 city nodes, the solution searched by NgISILDW-GA can be seen as unreasonable to the naked eye.

Ultimately, the culprit behind this disparity lies in the neural network's capability. If its input data is encoded in the permutation format, the model becomes ineffective. Currently, the neural network can process 01 coding and label coding data, allowing it to simulate the relationship between solutions and their fitness. However, when faced with permutation coding, the neural network fails to discern the impact of the input sequence's order on fitness. Of course, this issue could be addressed by substituting the neural network with a model capable of managing permutation coding, such as a graph neural network, but this alternative would render NgISILDW ineffective for addressing other discrete and continuous optimization problems. In conclusion, we caution against utilizing NgISILDW for large-scale permutation optimization problems, as it will only cause negative effects.

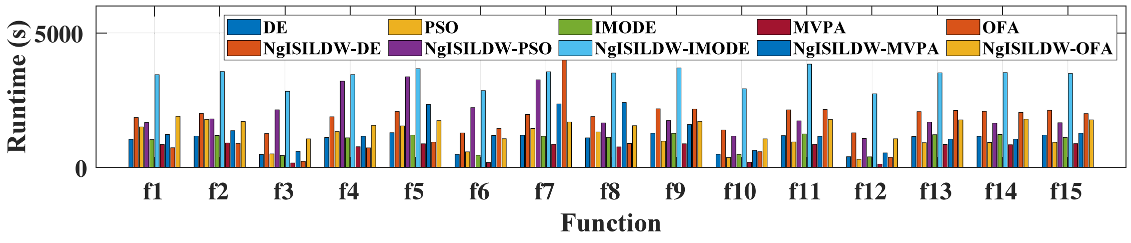


Fig. 12. Runtime of the five EAs and their NgISILDW embedded versions on CEC2013.

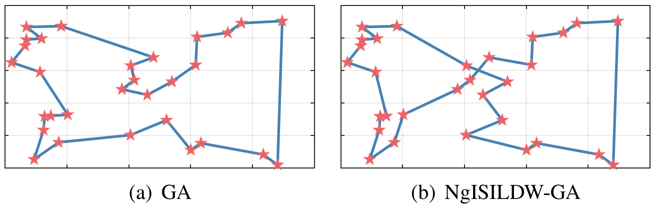


Fig. 13. Results of GA and NgISILDW-GA solving the 30-dimensional TSP problem.

7. Conclusion

Motivated by the practical demands of EA users and drawing inspiration from the CC framework, network pruning, and associated techniques, this paper introduces the dimensional window method and its refined version, NgISILDW. Distinct from existing LSEAs, the dimensional window method adopts a novel approach that restricts the evolutionary intensity of the population in the form of iterative evolutionary serialization. From a perturbation viewpoint, our analysis reveals that the effectiveness of the dimensional window method hinges on exposing the population to more diversity and valuable windows. Based on this insight, we devised the isometric segmentation individual-level window length mechanism and the neural network-guided window element strategy to form NgISILDW. Extensive experimentation has shown that both the dimensional window method and NgISILDW exhibit strong cross-problem domain adaptability and can be adapted to most EAs. The proposed method not only performs well on single-objective, multi-objective, and constrained problems with large-scale attributes but also achieves satisfactory results in ratio error estimation of voltage transformers, which real-world problems can be used to provide real-time estimations as early warnings of potential faulty voltage transformers. Notably, according to our experimental experience, the dimensional window method and NgISILDW can be seamlessly integrated into any EA on the PlatEMO platform with only a few lines of code, significantly enhancing the experience of EA users.

However, it is worth noting that no method is without its drawbacks. Firstly, (1) the proposed method is incompatible with CMAES, which leverages statistical distributions to direct evolution. Secondly, (2) the incorporation of neural networks significantly increases the computational overhead of EAs. Lastly, (3) it is unsuitable for permutation optimization problems. Potential solutions to drawbacks (2) and (3) may lie in alternative machine learning and deep learning techniques, such as meta-learning, curriculum learning, and continuous learning. In our future endeavors, we aim to further refine the utility of the dimensional window method by addressing these drawbacks and exploring the aforementioned techniques.

CRediT authorship contribution statement

Yafeng Sun: Writing – original draft, Software, Methodology, Conceptualization. **Xingwang Wang:** Writing – review & editing, Methodology, Conceptualization. **Junhong Huang:** Investigation, Conceptualization. **Bo Sun:** Investigation, Data curation. **Peng Liang:** Validation, Investigation.

Declaration of competing interest

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

Data availability

I have shared the link to my code at the manuscript.

References

[1] B. Cao, S. Fan, J. Zhao, S. Tian, Z. Zheng, Y. Yan, P. Yang, Large-scale many-objective deployment optimization of edge servers, IEEE Trans. Intell. Transp. Syst. 22 (2021) 3841–3849, http://dx.doi.org/10.1109/TITS.2021.3059455.
[2] M. Elshabrawy, A review on waste management techniques for sustainable energy production, Metaheuristic Optim. Rev. 3 (2025) 47–58.
[3] Z. Zhou, H. Yu, S. Mumtaz, S. Al-Rubaye, A. Tsourdos, R.Q. Hu, Power control optimization for large-scale multi-antenna systems, IEEE Trans. Wirel. Commun. 19 (2020) 7339–7352, http://dx.doi.org/10.1109/TWC.2020.3010701.
[4] J. Liu, R. Sarker, S. Elsayed, D. Essam, N. Siswanto, Large-scale evolutionary optimization: A review and comparative study, Swarm Evol. Comput. 85 (2024) 101466, http://dx.doi.org/10.1016/j.swevo.2023.101466.
[5] E.-S.M. El-kenawy, N. Khodadadi, S. Mirjalili, A.A. Abdelhamid, M.M. Eid, A. Ibrahim, Greylag goose optimization: Nature-inspired optimization algorithm, Expert Syst. Appl. 238 (2024) 122147, http://dx.doi.org/10.1016/j.eswa.2023.122147.
[6] H. Armanto, H.A. Rosyid, Muladi, Gunawan, Improved non-player character (NPC) behavior using evolutionary algorithm—A systematic review, Entertain. Comput. 52 (2025) 100875, http://dx.doi.org/10.1016/j.entcom.2024.100875.
[7] F. Yu, C. Lu, J. Zhou, L. Yin, K. Wang, A knowledge-guided bi-population evolutionary algorithm for energy-efficient scheduling of distributed flexible job shop problem, Eng. Appl. Artif. Intell. 128 (2024) 107458, http://dx.doi.org/10.1016/j.engappai.2023.107458.
[8] A.A. Alhussan, A.A. Abdelhamid, S.K. Towfek, A. Ibrahim, M.M. Eid, D.S. Khafaga, M.S. Saraya, Classification of diabetes using feature selection and hybrid Al-biruni Earth radius and dipper throated optimization, Diagnostics 13 (2023).
[9] Z. Xiong, X. Wang, Y. Li, W. Feng, Y. Liu, A problem transformation-based and decomposition-based evolutionary algorithm for large-scale multiobjective optimization, Appl. Soft Comput. 150 (2024) 111081, http://dx.doi.org/10.1016/j.asoc.2023.111081.
[10] X. Ma, X. Li, Q. Zhang, K. Tang, Z. Liang, W. Xie, Z. Zhu, A survey on cooperative co-evolutionary algorithms, IEEE Trans. Evol. Comput. 23 (2019) 421–441, http://dx.doi.org/10.1109/TEVC.2018.2868770.
[11] T.H.L. Fonseca, S.M. Nassar, A.C.M. de Oliveira, B. Agard, Low-dimensional space modeling-based differential evolution for large-scale global optimization problems, IEEE Trans. Evol. Comput. 27 (2023) 1529–1543, http://dx.doi.org/10.1109/TEVC.2022.3227440.
[12] F. Schoen, L. Tigli, Efficient large scale global optimization through clustering-based population methods, Comput. Oper. Res. 127 (2021) 105165, http://dx.doi.org/10.1016/j.cor.2020.105165.
[13] M. Chen, Y. Tan, SF-FWA: A self-adaptive fast fireworks algorithm for effective large-scale optimization, Swarm Evol. Comput. 80 (2023) 101314, http://dx.doi.org/10.1016/j.swevo.2023.101314.
[14] R. Montemanni, M. Dell'Amico, A. Corsini, Parallel drone scheduling vehicle routing problems with collective drones, Comput. Oper. Res. 163 (2024) 106514, http://dx.doi.org/10.1016/j.cor.2023.106514.
[15] C. Huang, X. Zhou, X. Ran, Y. Liu, W. Deng, W. Deng, Co-evolutionary competitive swarm optimizer with three-phase for large-scale complex optimization problem, Inform. Sci. 619 (2023) 2–18, http://dx.doi.org/10.1016/j.ins.2022.11.019.
[16] S. Mahdavi, S. Rahnamayan, M.E. Shiri, Multilevel framework for large-scale global optimization, Soft Comput. 21 (2017) 4111–4140, http://dx.doi.org/10.1007/s00500-016-2060-y.

[17] X. Ma, Z. Huang, X. Li, L. Wang, Y. Qi, Z. Zhu, Merged differential grouping for large-scale global optimization, IEEE Trans. Evol. Comput. 26 (2022) 1439–1451, http://dx.doi.org/10.1109/TEVC.2022.3144684.

[18] A. Kumar, S. Das, R. Mallipeddi, An efficient differential grouping algorithm for large-scale global optimization, IEEE Trans. Evol. Comput. 28 (2024) 32–46, http://dx.doi.org/10.1109/TEVC.2022.3230070.

[19] M. Chen, W. Du, W. Song, C. Liang, Y. Tang, An improved weighted optimization approach for large-scale global optimization, Complex Intell. Syst. 8 (2022) 1259–1280, http://dx.doi.org/10.1007/s40747-021-00596-3.

[20] W. Dong, Y. Wang, M. Zhou, A latent space-based estimation of distribution algorithm for large-scale global optimization, Soft Comput. 23 (2019) 4593–4615, http://dx.doi.org/10.1007/s00500-018-3390-8.

[21] X. Yao, Q. Zhao, D. Gong, S. Zhu, Solution of large-scale many-objective optimization problems based on dimension reduction and solving knowledge-guided evolutionary algorithm, IEEE Trans. Evol. Comput. 27 (2023) 416–429, http://dx.doi.org/10.1109/TEVC.2021.3110780.

[22] Y. Tian, Y. Feng, X. Zhang, C. Sun, A fast clustering based evolutionary algorithm for super-large-scale sparse multi-objective optimization, IEEE/ CAA J. Autom. Sin. 10 (2023) 1048–1063, http://dx.doi.org/10.1109/JAS.2022.105437.

[23] X. Gao, F. He, Y. Duan, C. Ye, J. Bai, C. Zhang, A space sampling based large-scale many-objective evolutionary algorithm, Inform. Sci. 679 (2024) 121077, http://dx.doi.org/10.1016/j.ins.2024.121077.

[24] J. Zhou, Y. Zhang, F. Yu, X. Yang, P.N. Suganthan, A staged fuzzy evolutionary algorithm for constrained large-scale multiobjective optimization, Appl. Soft Comput. (2024) 112297, http://dx.doi.org/10.1016/j.asoc.2024.112297.

[25] Q. Yang, W.-N. Chen, T. Gu, H. Jin, W. Mao, J. Zhang, An adaptive stochastic dominant learning swarm optimizer for high-dimensional optimization, IEEE Trans. Cybern. 52 (2022) 1960–1976, http://dx.doi.org/10.1109/TCYB.2020.3034427.

[26] Q.-V. Pham, T. Huynh-The, M. Alazab, J. Zhao, W.-J. Hwang, Sum-rate maximization for UAV-assisted visible light communications using NOMA: Swarm intelligence meets machine learning, IEEE Internet Things J. 7 (2020) 10375–10387, http://dx.doi.org/10.1109/JIOT.2020.2988930.

[27] Y. Li, W. Gong, Z. Hu, S. Li, A competitive and cooperative evolutionary framework for ensemble of constraint handling techniques, IEEE Trans. Syst. Man Cybern. Syst. (2023) http://dx.doi.org/10.1109/TSMC.2023.3343778.

[28] Z. Hu, W. Gong, W. Pedrycz, Y. Li, Deep reinforcement learning assisted co-evolutionary differential evolution for constrained optimization, Swarm Evol. Comput. 83 (2023) 101387, http://dx.doi.org/10.1016/j.swevo.2023.101387.

[29] B.-C. Wang, H.-X. Li, Q. Zhang, Y. Wang, Decomposition-based multiobjective optimization for constrained evolutionary optimization, IEEE Trans. Syst. Man Cybern. Syst. 51 (2021) 574–587, http://dx.doi.org/10.1109/TSMC.2018.2876335.

[30] G. Wu, X. Wen, L. Wang, W. Pedrycz, P.N. Suganthan, A voting-mechanism-based ensemble framework for constraint handling techniques, IEEE Trans. Evol. Comput. 26 (2022) 646–660, http://dx.doi.org/10.1109/TEVC.2021.3110130.

[31] Y. Tian, R. Cheng, X. Zhang, Y. Jin, PlatEMO: A MATLAB platform for evolutionary multi-objective optimization, IEEE Comput. Intell. Mag. 12 (2017) 73–87.

[32] Y. Li, W. Gong, F. Ming, T. Zhang, S. Li, Q. Gu, MToP: A MATLAB optimization platform for evolutionary multitasking, 2023, arXiv preprint arXiv:2312.08134.

[33] C. Peng, U. Mitra, Interference-constrained scheduling of a cognitive multihop underwater acoustic network, IEEE J. Ocean. Eng. 49 (2024) 507–521, http://dx.doi.org/10.1109/JOE.2023.3336462.

[34] L. Liu, A. Wang, G. Sun, J. Li, Maximizing data gathering and energy efficiency in UAV-assisted IoT: A multi-objective optimization approach, Comput. Netw. 235 (2023) 109986, http://dx.doi.org/10.1016/j.comnet.2023.109986.

[35] L. Saha, H.K. Tripathy, T. Gaber, H. El-Gohary, E.-S.M. El-kenawy, Deep churn prediction method for telecommunication industry, Sustainability 15 (2023) http://dx.doi.org/10.3390/su15054543.

[36] H. Cheng, M. Zhang, J.Q. Shi, A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations, IEEE Trans. Pattern Anal. Mach. Intell. (2024) 1–20, http://dx.doi.org/10.1109/TPAMI.2024.3447085.

[37] K.P. Murphy, Machine Learning: A Probabilistic Perspective, MIT Press, 2012.

[38] Z. Liu, M. Sun, T. Zhou, G. Huang, T. Darrell, Rethinking the value of network pruning, 2018, arXiv preprint arXiv:1810.05270.

[39] R. Tanabe, A. Fukunaga, Success-history based parameter adaptation for differential evolution, in: 2013 IEEE Congress on Evolutionary Computation, 2013, pp. 71–78, http://dx.doi.org/10.1109/CEC.2013.6557555.

[40] X. Li, K. Tang, M.N. Omidvar, Z. Yang, K. Qin, H. China, Benchmark functions for the CEC 2013 special session and competition on large-scale global optimization, Gene 7 (2013) 8.

[41] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359, http://dx.doi.org/10.1023/A:1008202821328.

[42] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995, pp. 39–43, http://dx.doi.org/10.1109/MHS.1995.494215.

[43] K.M. Sallam, S.M. Elsayed, R.K. Chakrabortty, M.J. Ryan, Improved multi-operator differential evolution algorithm for solving unconstrained problems, in: 2020 IEEE Congress on Evolutionary Computation, CEC, 2020, pp. 1–8, http://dx.doi.org/10.1109/CEC48606.2020.9185577.

[44] H. Bouchekara, Most valuable player algorithm: A novel optimization algorithm inspired from sport, Oper. Res. 20 (2020) 139–195, http://dx.doi.org/10.1007/s12351-017-0320-y.

[45] G.-Y. Zhu, W.-B. Zhang, Optimal foraging algorithm for global optimization, Appl. Soft Comput. 51 (2017) 294–313, http://dx.doi.org/10.1016/j.asoc.2016.11.047.

[46] A. Chen, Z. Ren, W. Guo, Y. Liang, Z. Feng, An efficient adaptive differential grouping algorithm for large-scale black-box optimization, IEEE Trans. Evol. Comput. 27 (2023) 475–489, http://dx.doi.org/10.1109/TEVC.2022.3170793.

[47] M. Yang, M.N. Omidvar, C. Li, X. Li, Z. Cai, B. Kazimipour, X. Yao, Efficient resource allocation in cooperative co-evolution for large-scale global optimization, IEEE Trans. Evol. Comput. 21 (2017) 493–505, http://dx.doi.org/10.1109/TEVC.2016.2627581.

[48] M.N. Omidvar, M. Yang, Y. Mei, X. Li, X. Yao, DG2: A faster and more accurate differential grouping for large-scale black-box optimization, IEEE Trans. Evol. Comput. 21 (2017) 929–942, http://dx.doi.org/10.1109/TEVC.2017.2694221.

[49] J.-R. Jian, Z.-G. Chen, Z.-H. Zhan, J. Zhang, Region encoding helps evolutionary computation evolve faster: A new solution encoding scheme in particle swarm for large-scale optimization, IEEE Trans. Evol. Comput. 25 (2021) 779–793, http://dx.doi.org/10.1109/TEVC.2021.3065659.

[50] F. Wang, X. Wang, S. Sun, A reinforcement learning level-based particle swarm optimization algorithm for large-scale optimization, Inform. Sci. 602 (2022) 298–312, http://dx.doi.org/10.1016/j.ins.2022.04.053.

[51] E. Zhang, Z. Nie, Q. Yang, Y. Wang, D. Liu, S.-W. Jeon, J. Zhang, Heterogeneous cognitive learning particle swarm optimization for large-scale optimization problems, Inform. Sci. 633 (2023) 321–342, http://dx.doi.org/10.1016/j.ins.2023.03.086.

[52] Z.-J. Wang, Q. Yang, Y.-H. Zhang, S.-H. Chen, Y.-G. Wang, Superiority combination learning distributed particle swarm optimization for large-scale optimization, Appl. Soft Comput. 136 (2023) 110101, http://dx.doi.org/10.1016/j.asoc.2023.110101.

[53] R. Cheng, Y. Jin, M. Olhofer, B. sendhoff, Test problems for large-scale multiobjective and many-objective optimization, IEEE Trans. Cybern. 47 (2017) 4108–4121, http://dx.doi.org/10.1109/TCYB.2016.2600577.

[54] S. Kukkonen, J. Lampinen, GDE3: The third evolution step of generalized differential evolution, in: 2005 IEEE Congress on Evolutionary Computation, vol. 1, 2005, pp. 443–450 Vol.1, http://dx.doi.org/10.1109/CEC.2005.1554717.

[55] K. Deb, H. Jain, An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints, IEEE Trans. Evol. Comput. 18 (2014) 577–601, http://dx.doi.org/10.1109/TEVC.2013.2281535.

[56] Q. Zhang, H. Li, MOEA/D: A multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 11 (2007) 712–731, http://dx.doi.org/10.1109/TEVC.2007.892759.

[57] H. Li, Q. Zhang, Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II, IEEE Trans. Evol. Comput. 13 (2009) 284–302, http://dx.doi.org/10.1109/TEVC.2008.925798.

[58] L. Li, C. He, R. Cheng, H. Li, L. Pan, Y. Jin, A fast sampling based evolutionary algorithm for million-dimensional multiobjective optimization, Swarm Evol. Comput. 75 (2022) 101181, http://dx.doi.org/10.1016/j.swevo.2022.101181.

[59] X. Zhang, Y. Tian, R. Cheng, Y. Jin, A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization, IEEE Trans. Evol. Comput. 22 (2018) 97–112, http://dx.doi.org/10.1109/TEVC.2016.2600642.

[60] C. He, L. Li, Y. Tian, X. Zhang, R. Cheng, Y. Jin, X. Yao, Accelerating large-scale multiobjective optimization via problem reformulation, IEEE Trans. Evol. Comput. 23 (2019) 949–961, http://dx.doi.org/10.1109/TEVC.2019.2896002.

[61] J. Yuan, H.-L. Liu, Y.-S. Ong, Z. He, Indicator-based evolutionary algorithm for solving constrained multiobjective optimization problems, IEEE Trans. Evol. Comput. 26 (2022) 379–391, http://dx.doi.org/10.1109/TEVC.2021.3089155.

[62] K. Qiao, J. Liang, K. Yu, C. Yue, H. Lin, D. Zhang, B. Qu, Evolutionary constrained multiobjective optimization: Scalable high-dimensional constraint benchmarks and algorithm, IEEE Trans. Evol. Comput. 28 (2024) 965–979, http://dx.doi.org/10.1109/TEVC.2023.3281666.

[63] K. Qiao, K. Yu, B. Qu, J. Liang, H. Song, C. Yue, An evolutionary multitasking optimization framework for constrained multiobjective optimization problems, IEEE Trans. Evol. Comput. 26 (2022) 263–277, http://dx.doi.org/10.1109/TEVC.2022.3145582.

[64] C. He, R. Cheng, Y. Tian, X. Zhang, K.C. Tan, Y. Jin, Paired offspring generation for constrained large-scale multiobjective optimization, IEEE Trans. Evol. Comput. 25 (2021) 448–462, http://dx.doi.org/10.1109/TEVC.2020.3047835.

[65] B.-C. Wang, Z.-Y. Shui, Y. Feng, Z. Ma, Evolutionary algorithm with dynamic population size for constrained multiobjective optimization, Swarm Evol. Comput. 73 (2022) 101104, http://dx.doi.org/10.1016/j.swevo.2022.101104.

[66] F. Ming, W. Gong, D. Li, L. Wang, L. Gao, A competitive and cooperative swarm optimizer for constrained multiobjective optimization problems, IEEE Trans. Evol. Comput. 27 (2023) 1313–1326, http://dx.doi.org/10.1109/TEVC.2022.3199775.

[67] K. Qiao, J. Liang, K. Yu, W. Guo, C. Yue, B. Qu, P. Suganthan, Benchmark problems for large-scale constrained multi-objective optimization with baseline results, Swarm Evol. Comput. 86 (2024) 101504, http://dx.doi.org/10.1016/j.swevo.2024.101504.

[68] C. He, R. Cheng, C. Zhang, Y. Tian, Q. Chen, X. Yao, Evolutionary large-scale multiobjective optimization for ratio error estimation of voltage transformers, IEEE Trans. Evol. Comput. 24 (2020) 868–881, http://dx.doi.org/10.1109/TEVC.2020.2967501.

[69] J. Liang, Z. Chen, Y. Wang, X. Ban, K. Qiao, K. Yu, A dual-population constrained multi-objective evolutionary algorithm with variable auxiliary population size, Complex Intell. Syst. 9 (2023) 5907–5922.

[70] X. Ban, J. Liang, K. Yu, K. Qiao, P.N. Suganthan, Y. Wang, A subspace search-based evolutionary algorithm for large-scale constrained multiobjective optimization and application, IEEE Trans. Cybern. 55 (2025) 2486–2499, http://dx.doi.org/10.1109/TCYB.2025.3548414.

[71] X. Ban, J. Liang, K. Qiao, K. Yu, Y. Wang, J. Peng, B. Qu, A decision variables classification-based evolutionary algorithm for constrained multi-objective optimization problems, IEEE/CAA J. Autom. Sin. 12 (2025) 1–19.

[72] L.R. Farias, A.F. Araújo, An inverse modeling constrained multi-objective evolutionary algorithm based on decomposition, in: 2024 IEEE International Conference on Systems, Man, and Cybernetics, SMC, 2024, pp. 3727–3732, http://dx.doi.org/10.1109/SMC54092.2024.10831275.

[73] L. Si, X. Zhang, Y. Zhang, S. Yang, Y. Tian, An efficient sampling approach to offspring generation for evolutionary large-scale constrained multi-objective optimization, IEEE Trans. Emerg. Top. Comput. Intell. (2025) 1–13, http://dx.doi.org/10.1109/TETCI.2025.3526268.

[74] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach, IEEE Trans. Evol. Comput. 3 (1999) 257–271, http://dx.doi.org/10.1109/4235.797969.

[75] Y. Tian, L. Wang, S. Yang, J. Ding, Y. Jin, X. Zhang, Neural network-based dimensionality reduction for large-scale binary optimization with millions of variables, IEEE Trans. Evol. Comput. (2024) 1, http://dx.doi.org/10.1109/TEVC.2024.3400398.

[76] Y. Tian, S. Yang, X. Zhang, An evolutionary multiobjective optimization based fuzzy method for overlapping community detection, IEEE Trans. Fuzzy Syst. 28 (2020) 2841–2855, http://dx.doi.org/10.1109/TFUZZ.2019.2945241.

[77] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evol. Comput. 9 (2001) 159–195, http://dx.doi.org/10.1162/106365601750190398.