A Universal Framework for Automatically Generating Single- and Multi-Objective Evolutionary Algorithms

Ye Tian, *Senior Member, IEEE*, Xuhong Qi, Shangshang Yang, Cheng He, *Senior Member, IEEE*, Kay Chen Tan, *Fellow, IEEE*, Yaochu Jin, *Fellow, IEEE*, and Xingyi Zhang, *Fellow, IEEE*

***Abstract*—The automated design of evolutionary algorithms (EAs) is receiving more and more attention, which considerably reduces the labor-intensive process of algorithm design demanding expertise. While existing automated methods for selecting, tuning, and creating EAs mainly rely on existing operators connected in sequence, they can hardly surpass the performance thresholds of existing EAs across various problems. To break through the shackles of existing EAs, this work proposes a universal framework to automatically create EAs from scratch: First, a series of building blocks are designed without using existing operators, and thus offer the potential to exceed the performance thresholds of existing EAs. Second, these blocks are connected like the layers of deep neural networks, which can form non-sequential architectures to pursue good performance. Third, these blocks are parameterized and can be trained like neural networks, able to exhibit better performance on given problems. Using the proposed framework, non-sequential EAs characterized by dozens of parameters are trained on a few problems, which outperform 36 existing algorithms on more than 200 single- and multi-objective problem instances. Particularly, the non-sequential EAs without surrogate demonstrate superior convergence over surrogate-assisted EAs on expensive problems, and outperform gradient methods on unimodal problems.**

***Index Terms*—Automated design, evolutionary algorithm, building block, graph representation, training.**

I. Introduction

EVOLUTIONARY computation has emerged as a pivotal technique for solving complex optimization problems, with a variety of algorithms inspired by biological evolution [1] and swarm behavior [2] developed over the past decades. These algorithms tackle specific optimization problems using various search strategies, traditionally crafted through expert-driven, labor-intensive trial-and-error efforts. As problems grow in multimodal landscapes [3], high-dimensional spaces [4], strict constraints [5], sparse optimal solutions [6], [7], and other difficulties, the manual crafting of evolutionary algorithms (EAs) proves inadequate [8].

To address this issue, the automated design of EAs has received increasing interest in recent years [9]. With the development of algorithm selection, tuning, and creation, EAs have shown significant improvements in adaptivity. Algorithm recommendation [10], [11] aims to identify the optimal EA from a set of candidates for specific problems before optimization, by means of learning underlying patterns between algorithm performance and problem characteristics. Algorithm tuning involves the adjustment of parameters, which is usually achieved by adaptive rules [12], [13] and reinforcement learning [14], [15]. Algorithm creation focuses on the development of new EAs, which includes the online switch between multiple existing components [16], [17] and the offline generation of new components [18], [19].

Given a minimization single-objective optimization problem $f(\mathbf{x})$, an automated design method finds the best configuration $\mathcal{C}$ (i.e., a complete EA for algorithm selection, parameters of a given EA for algorithm tuning, and a combination of multiple components for algorithm creation) for an algorithm prototype EA on $f(\mathbf{x})$:

$$\min_{\mathcal{C}} f(EA(P|\mathcal{C})) \ , \tag{1}$$

where algorithm instance $EA(P|\mathcal{C})$ evolves a randomly initialized population P using configuration $\mathcal{C}$, and outputs the best found solution after multiple generations. For this aim, Eq. (1) has been successfully solved using heuristic strategies [13], evolutionary computation [20], deep learning [21], and large language models [22].

However, a significant limitation is that many automated design methods define the prototype EA by drawing from the algorithmic components and architectures of existing EAs [23], which inherently caps

Manuscript received –. This work was supported in part by the National Natural Science Foundation of China (No. 62522301, No. 62276001, No. W2441019), the Anhui Provincial Natural Science Foundation (No. 2308085J03), and the State Key Laboratory of Advanced Electromagnetic Technology (No. AET 2025KF013). *(Corresponding author: Xingyi Zhang.)*

Y. Tian, X. Qi, S. Yang, and X. Zhang are with the Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, School of Computer Science and Technology, Anhui University, Hefei 230601, China (email: field910921@gmail.com; qi_xuhong@foxmail.com; yangshang0308@gmail.com; xyzhanghust@gmail.com).

C. He is with the State Key Laboratory of Advanced Electromagnetic Technology, School of Electrical and Electronic Engineering, Huazhong University of Science and Technology, Wuhan 430074, China (email: chenghehust@gmail.com).

K. C. Tan is with the Department of Data Science and Artificial Intelligence, The Hong Kong Polytechnic University, Hong Kong SAR (email: kctan@polyu.edu.hk).

Y. Jin is with the School of Engineering, Westlake University, Hangzhou 310030, China (email: jinyaochu@westlake.edu.cn).

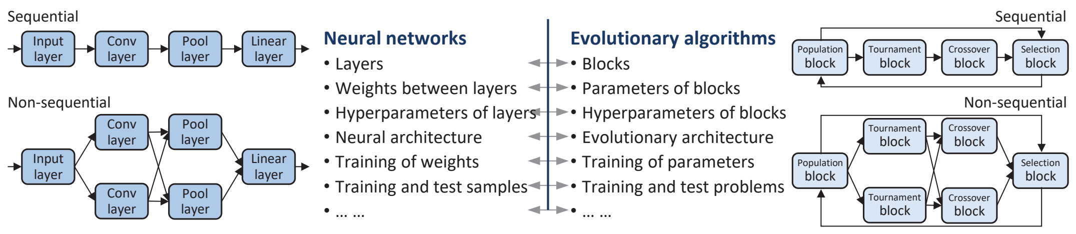


Fig. 1. Similarities between neural networks and NeuroEAs created by the proposed framework.

their performance to the thresholds achieved by the best candidate component and architecture. On the other hand, some methods employ models especially neural networks [19], [24] to steer clear of the performance thresholds of existing EAs, but they may cause over-customization issues. For example, if the algorithm prototype EA is defined as a simple linear model

$$EA_{linear}(P=\{\mathbf{x}\}|\mathcal{C}=\{\mathbf{w},\mathbf{b}\})=\mathbf{w}\mathbf{x}+\mathbf{b} \tag{2}$$

that iteratively receives a parent solution $\mathbf{x}$ and outputs an offspring solution, giving a configuration $\mathcal{C}^*=\{\mathbf{w}^*=(0,\ldots,0),\mathbf{b}^*=\text{argmin}_{\mathbf{x}}f(\mathbf{x})\}$, according to Eq. (2),

$$EA_{linear}(P|\mathcal{C}^*)\equiv\mathbf{b}^*=\text{argmin}_{\mathbf{x}}f(\mathbf{x}) \ . \tag{3}$$

It means that the algorithm instance can converge to the optimum of the training problem $f(\mathbf{x})$ at the first generation. Obviously, the algorithm instance cannot solve other problems and is completely useless. In fact, when solving Eq. (1), it is difficult to judge whether a "good" configuration $\mathcal{C}^*$ offers powerful search ability or just fixes the output of EA to the optimum of $f(\mathbf{x})$. Note that the adoption of multiple training problems $f_1, f_2, \ldots$ is still ineffective, as the output of EA can be fixed to a population $\{\mathbf{b}_1=\text{argmin}_{\mathbf{x}}f_1(\mathbf{x}),\mathbf{b}_2=\text{argmin}_{\mathbf{x}}f_2(\mathbf{x}),\ldots\}$.

Due to the adoption of existing algorithms and the over-customization issues, existing automated design methods were reported to have limited transferability and effectiveness [9], while some of them were even outperformed by manually designed EAs as compared in [25]. To address these limitations, inspired by the non-sequential architectures of deep neural networks, this paper suggests a universal framework to define the algorithm prototype as a graph G:

$$EA_G(P|\mathcal{C}=\{V,E\})=G(P|V,E) \ , \tag{4}$$

where V is a set of building blocks (i.e., nodes), E is a set of directed and weighted edges, and population P is processed through the workflow defined in the graph G. In particular, the blocks V are not drawn from existing algorithmic components, and the edges E define non-sequential architectures that are different from the sequential architectures of existing EAs. More importantly, the blocks are designed based on search space independent operators [18], which theoretically exhibit the same dynamics no matter how the optimal solution of a problem is translated, scaled, or rotated in the search space. That is, the blocks will not suffer from the over-customization issues in Eq. (3). To summarize, the evolutionary algorithms with neural architectures (NeuroEAs) have the following characteristics:

1) The components of NeuroEAs are defined as blocks, each functioning as a solution processing unit based on search space independent operators. These blocks are characterized by hyperparameters and parameters, which can be adjusted to modulate the search dynamics inherent to each block. The operators in genetic algorithms, differential evolution, and many other EAs can be considered as specific implementations of these blocks, and these blocks can also innovate novel operators.
2) The architectures of NeuroEAs are defined as graphs, in which each node denotes a block and each edge denotes the flow of solutions between blocks, enabling the representation of non-sequential architectures. The search space independent operators and non-sequential architectures offer the potential to exceed the performance thresholds of existing EAs, enhancing both the transferability and effectiveness of NeuroEAs.
3) Owing to the parameterized representation, NeuroEAs are created and trained like neural networks. In fact, NeuroEAs introduce many concepts similar to neural networks as illustrated in Fig. 1. By automatically tuning block parameters on training problems, NeuroEAs can achieve outstanding performance across various problems.

In the experiments, we create a series of NeuroEAs and train them on benchmark problems, where larger NeuroEAs exhibit better performance than smaller ones and outperform 36 existing algorithms on more than 200 single-objective, expensive, combinatorial, and multi-objective optimization problem instances.

The rest of this paper is organized as follows. Section II reviews existing automated design methods of EAs and gives the core idea of this work. Section III details the proposed framework for creating NeuroEAs. Section IV analyzes the experimental results. Lastly, Section V draws conclusions and discusses future directions.

II. Related Work and Motivation

A. Automated Design Methods for Evolutionary Algorithms

The automated design of EAs aims to adapt the algorithmic architectures, components, and parameters for solving specific types of optimization problems, so as to reduce the heavy requirements of expertise and effort. Existing automated design methods for EAs can be roughly divided into three categories, including algorithm selection [26], tuning [27], and creation [9].

The concept of algorithm selection was proposed as early as the 1970s [28] and developed since the 1990s [29]. Smith-Miles [30] proposed a meta-learning inspired framework based on neural networks, which develops insights into the relationships between the characteristics of quadratic assignment problems and algorithm performance. Nudelman *et al.* [31] suggested an empirical hardness model to select different solvers for satisfiability problems, by means of identifying features of satisfiability problem instances using regression models. Mayer *et al.* [32] investigated a simulation based supervised learning approach to determine a suitable algorithm for vehicle routing problems, on the basis of neural network based classification and regression models. Kerchke *et al.* [33] developed exploratory landscape analysis features for characterizing continuous optimization problems' landscapes, and constructed an algorithm selection model. Tian *et al.* [11] depicted continuous optimization problems with reverse Polish expressions consisting of multiple operand and operator symbols, and regarded algorithm selection as a natural language processing task handled by deep recurrent neural networks.

In contrast to algorithm selection involving multiple candidate EAs, parameter tuning focuses on the enhancement of a single EA. Eiben and Smit [34] suggested a conceptual framework that considers parameter tuning as an optimization problem, and established different taxonomies to categorize tuning methods. Tong *et al.* [35] proposed a framework for parameter-tuned algorithms, on the basis of sequential parameter optimization and population based algorithm portfolio. Lu *et al.* [15] defined an elite network model to learn the relationship between optimal particle velocity and position in particle swarm optimization, and the elite network model is trained according to well-performing particles. Sun *et al.* [14] adjusted the parameters of differential evolution using a deep neural network, which is trained by policy gradient. Zhao *et al.* [13] suggested a parameter fitness evaluation criterion, for adjusting the parameters for each particle in particle swarm optimization. It should be noted that most works considered online parameter tuning conducted during the optimization procedure of EAs, while the offline learning of parameter control policies has also been studied [36].

Algorithm creation aims to create new EAs based on the ensemble of existing components or the generation of new components, especially operators for generating new solutions. Bezerra *et al.* [23] designed a multi-objective EA template, which has a fixed architecture and replaceable components that can be optimized using irace. Tian *et al.* [17] employed deep Q-network for operator ensemble, where states are defined as the statistics of parent solutions and actions are defined as candidate operators. Stein and Bäck [22] used large language models to iteratively generate, mutate, and select new EAs, on the basis of in-context learning and delicate prompts. Tisdale and Tauritz [37] represented EAs using directed graphs, which can be optimized to find better architectures consisting of existing components. Yi *et al.* [38] proposed a general search framework to formulate metaheuristics, and used reinforcement learning to select and connect existing components. Li *et al.* [19] adopted the multi-head self-attention mechanism to generate new operators, Tian *et al.* [39] adopted the proximal policy optimization to generate new operators, and Zhao *et al.* [21] adopted the variational graph autoencoder to generate new evolutionary architectures.

In short, some of the above methods are based on existing algorithmic components and architectures, and some others are based on learnable models. As discussed in Section I, both these two types of methods have limitations, where the former is likely to have low versatility and limited performance improvement, and the latter may be over-customized for specific optimal solutions. In fact, many manually designed EAs were also over-customized for problems whose optimal solutions are the origin [40], having been a critical problem in the evolutionary computation community [8]. By contrast, this paper creates new EAs by following the search space independent operator theorem [18], where the generated operators are different from existing algorithmic components and not biased for specific optimal solutions.

B. Core Idea of This Work

Ideally, an EA effective for a training problem $f(\mathbf{x})$ should be independent of specific problem instances. For example, an EA effective for $f(\mathbf{x})$ should also be effective for the translated problem $f(\mathbf{x}+b)$ with any b, and thus it is not over-customized for any specific optimal solution as the optimal variables vary with the value of b. Formally, considering possible search space transformations $\mathcal{T}$ including translation $\mathcal{T}(\mathbf{x}) = \mathbf{x}+b$ (b denotes any scalar), scale $\mathcal{T}(\mathbf{x}) = a\mathbf{x}$ (a denotes any nonzero scalar), and rotation $\mathcal{T}(\mathbf{x}) = \mathbf{x}M$ (M denotes any orthogonal matrix), an EA effective for $f(\mathbf{x})$ should be also effective for $f(\mathcal{T}(\mathbf{x}))$; in other words, the EA is invariant to search space transformations [18].

An EA generally consists of operators for generating solutions and selection strategies for truncating solutions, which are conducted in search spaces and objective spaces, respectively. Thus, the operators are crucial to the search space transformation invariance properties of EAs. According to [41], an operator h invariant to transformation $\mathcal{T}$ should satisfy

$$h(\mathcal{T}(\mathbf{x}_1),\ldots,\mathcal{T}(\mathbf{x}_m)) = \mathcal{T}(h(\mathbf{x}_1,\ldots,\mathbf{x}_m)) \ , \qquad (5)$$

where operator $h(\mathbf{x}_1,\ldots,\mathbf{x}_m)$ receives m parent solutions and outputs one offspring solution. In [18], the sufficient and necessary condition of operator h satisfying Eq. (5) has been derived, which is known as the following theorem:

TABLE I
SEVEN CATEGORIES OF BLOCKS FOR CREATING NEUROEAS USING THE PROPOSED FRAMEWORK

Block	Function	Hyperparameters	Parameters	Input n solutions can generate how many new solutions?	Function evaluations needed?	Time complexity
Population	Storing solutions without any processing	–	–	n	Yes	$O(1)$
Crossover	Mating multiple solutions to generate a new solution	No. of parents m, no. of parameter sets s	$3(m-1)s$ operator parameters	$\lceil n/m \rceil$	No	$O(nsD)$
Mutation	Mutating one solution	No. of parameter sets s	$2s$ operator parameters	n	No	$O(nsD)$
Exchange	Exchanging multiple solutions to generate a new solution	No. of parents m	m probabilities of selecting each parent	$\lceil n/m \rceil$	No	$O(nD)$
Kopt	Flipping part of a solution for sequence optimization	Max no. k for k-opt	k probabilities of applying 1-, ... ,k-opt	n	No	$O(nkD)$
Tournament	Tournament selection for mating selection	No. of output solutions n', max no. k for k-tournament	One parameter $i \leq k$ for applying i-tournament	n'	Yes	$O(n'k)$
Selection	Retaining part of solutions for environmental selection	No. of retained solutions n'	–	$\min\{n, n'\}$	Yes	$O(n^2M)$

*Theorem 1 (**Search space independent operator**):* A continuously differentiable operator h is translation, scale, and rotation invariant if and only if the operator is

$$h(x_{1d}, \dots, x_{md}) = w_1 x_{1d} + \cdots + w_m x_{md} \ , \qquad (6)$$

where x_{1d} denotes the d-th variable of solution $\mathbf{x}_1$ and $w_1, \dots, w_m$ can be any constants with $w_1 + \cdots + w_m = 1$.

Obviously, the operators of many existing EAs like genetic algorithms and differential evolution can be regarded as specific implementations of Eq. (6). This formula can not only emulate existing operators but also innovate new ones by tuning the parameters $w_1, \dots, w_m$, thereby reinforcing the universality of created EAs. More importantly, search space independent operators are theoretically proved to avoid over-customization for specific optimal solutions. For example, Eq. (6) does not have a constant term like Eq. (2) that can be set to the optimum of a training problem.

Obviously, both the limitation of adopting existing operators and the issues of over-customization can be addressed by search space independent operators. By implementing Eq. (6) based on different restrictions, this work designs several categories of blocks for creating novel EAs, whose details are given in the next section.

III. THE PROPOSED FRAMEWORK FOR CREATING EVOLUTIONARY ALGORITHMS

A. Design of Blocks

The proposed framework introduces seven categories of blocks, where each block is a function whose input and output are multiple solutions. A block is similar to a layer in neural networks, which includes a number of hyperparameters and parameters. Similar to the trainable weights of neural networks, the parameters of blocks can be tuned on training problems, thus achieving outstanding performance. Table I summarizes the definitions of seven categories of blocks for creating NeuroEAs, including:

- **Population:** It is the essential block in NeuroEAs for storing solutions, analogous to the role of populations in general EAs. At each generation, solutions are passed from population blocks to other blocks, then passed back to population blocks. A population block does not contain any operation, hyperparameter, or parameter, and its output equals its input.
- **Crossover:** It is an implementation of search space independent operators described in Eq. (6), which uses every m input solutions to generate one offspring solution. To provide diverse search dynamics with randomness, a crossover block contains the following parameters rather than $w_1, \dots, w_m$:
$$\begin{pmatrix} \mu_{12}, \sigma_{12}, p_{12}, \mu_{13}, \sigma_{13}, p_{13}, \dots, \mu_{1m}, \sigma_{1m}, p_{1m} \\ \mu_{22}, \sigma_{22}, p_{22}, \mu_{23}, \sigma_{23}, p_{23}, \dots, \mu_{2m}, \sigma_{2m}, p_{2m} \\ \dots \quad \dots \\ \mu_{s2}, \sigma_{s2}, p_{s2}, \mu_{s3}, \sigma_{s3}, p_{s3}, \dots, \mu_{sm}, \sigma_{sm}, p_{sm} \end{pmatrix}, \qquad (7)$$
where m is the number of parents, s is the number of parameter sets, each $\mu \in [-1, 1]$ and $\sigma \in [0, 1]$ are the mean and standard deviation of a normal distribution for sampling w, respectively, and the subsequent $p \in [0, 1]$ is the probability of selecting the distribution. When performing Eq. (6), each w_j for $j = 2, \dots, m$ is sampled from the normal distribution $\mathcal{N}(\mu_{ij}, \sigma_{ij}^2)$, where $i \in [1, s]$ is determined by roulette-wheel selection based on $p_{1j}, \dots, p_{sj}$. Then, w_1 can be calculated by $1 - w_2 - \cdots - w_m$ so that $w_1 + \cdots + w_m = 1$ always holds. Fig. 2 depicts an example of this procedure when there are $m = 4$ parents, $s = 3$ parameter sets, and $D = 3$ variables.
- **Mutation:** It is an implementation of search space independent operators with three fixed parent solutions, including a solution $\mathbf{x}$ to mutate, the lower

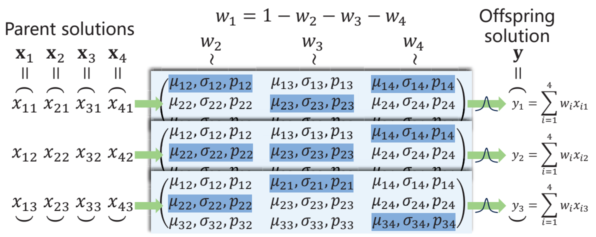

$w_1 = 1 - w_2 - w_3 - w_4$
Parent solutions
Offspring solution
$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$ $\mathbf{x}_4$
$\mathbf{y}$
w_2 w_3 w_4
x_{11} x_{21} x_{31} x_{41}
x_{12} x_{22} x_{32} x_{42}
x_{13} x_{23} x_{33} x_{43}
$\mu_{12}, \sigma_{12}, p_{12}$ $\mu_{13}, \sigma_{13}, p_{13}$ $\mu_{14}, \sigma_{14}, p_{14}$
$\mu_{22}, \sigma_{22}, p_{22}$ $\mu_{23}, \sigma_{23}, p_{23}$ $\mu_{24}, \sigma_{24}, p_{24}$
$\mu_{21}, \sigma_{21}, p_{21}$
$\mu_{32}, \sigma_{32}, p_{32}$ $\mu_{33}, \sigma_{33}, p_{33}$ $\mu_{34}, \sigma_{34}, p_{34}$
$y_1 = \sum_{i=1}^{4} w_i x_{i1}$
$y_2 = \sum_{i=1}^{4} w_i x_{i2}$
$y_3 = \sum_{i=1}^{4} w_i x_{i3}$

Fig. 2. An example of the calculation of crossover blocks.

bound $\mathbf{l}$, and the upper bound $\mathbf{u}$ of the search space. That is, the solution is mutated by

$$y_d = x_d + \mathcal{N}(0, \sigma_i^2) \cdot (u_d - l_d) \ , \tag{8}$$

and a mutation block has the following parameters:

$$(\sigma_1, p_1, \sigma_2, p_2, \dots, \sigma_s, p_s) \ , \tag{9}$$

where s is the number of parameter sets, each $\sigma \in [0,1]$ is the standard deviation of a normal distribution, and the subsequent $p \in [0,1]$ is the probability of selecting the distribution. When performing Eq. (8), i is determined by roulette-wheel selection based on $p_1, \dots, p_s$.

- **Exchange:** It is an implementation of search space independent operators with w being either 0 or 1. That is, each variable of an offspring solution is directly taken from one of m parent solutions. An exchange block contains the following parameters:

$$(p_1, p_2, \dots, p_m) \ , \tag{10}$$

where m is the number of parents and each $p \in [0,1]$ is the probability of selecting a parent solution. When setting the offspring variable y_d to a parent variable x_{id}, i is determined by roulette-wheel selection based on $p_1, \dots, p_m$.

- **Kopt:** It mimics the operation of k-opt [42] for sequence optimization like travelling salesman problems. To provide a universal and black-box search paradigm, it randomly selects one exchange from all the $2^{k-1}(k-1)!$ exchanges of k-opt, and accepts the exchange without consuming any function evaluation. A kopt block has the following parameters:

$$(p_1, p_2, \dots, p_k) \ , \tag{11}$$

where k is the maximum number for k-opt and $p_i \in [0,1]$ is the probability of selecting i-opt. When performing i-opt (1-opt means that the solution is not changed) on an offspring solution, i is determined by roulette-wheel selection based on $p_1, \dots, p_k$.

- **Tournament:** It selects solutions with replacement from the input solutions, which is usually used as mating selection. It employs tournament selection based on solutions' fitness values, which are defined as objective values for single-objective optimization and non-dominated front numbers [43] for multi- and many-objective optimization; besides, feasible solutions are given higher priority than infeasible solutions. A tournament block has one parameter i for using i-tournament selection.

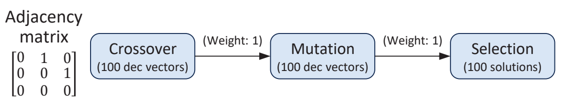
- **Selection:** It selects solutions without replacement from the input solutions, which is usually used as environmental selection. It truncates solutions using their objective values for single-objective optimization, using the strategy of SPEA2 [44] for multi-objective optimization, and using the strategy of SPEA2 with shift based density estimation [45] for many-objective optimization. A selection block does not have parameters to tune.

With these seven categories of blocks, the created NeuroEAs are able to solve single-, multi-, and many-objective optimization problems with real, binary, and sequence encoding. While search space independent operators search in continuous spaces, the variables are discretized before the function evaluations of binary and sequence optimization problems. On the other hand, more categories of blocks could be designed to further enhance the performance on specific problems, such as gradient-guided operators for continuous optimization [46], delicate selection strategies for many-objective optimization [47], and customized search strategies for applications with special encoding.

B. Design of Edges

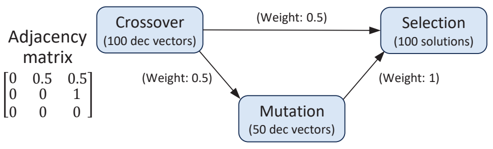
The proposed framework connects blocks using directed and weighted edges, thus constituting neural architectures for NeuroEAs. The directions of edges indicate the flows of solutions between blocks, and the weights of edges indicate the ratio of passed solutions. Such a simple consideration encompasses complex scenarios, where the adjustment of edges' weights can lead to distinct architectures as illustrated in Fig. 3.

In Fig. 3(a), the crossover and mutation blocks are connected with a weight of 1 and the mutation and selection blocks are connected with a weight of 1. It implies that 100 solutions are passed from the crossover block through the mutation block to the selection block. In Fig. 3(b), the crossover and mutation blocks are connected with a weight of 0.5, the crossover and selection blocks are connected with a weight of 0.5, and the mutation and selection blocks are connected with a weight of 1. It implies that the crossover block outputs 100 solutions, where 50 of them are passed to the mutation block and the other 50 are passed to the selection block; then, the solutions are passed from the mutation block to the selection block. In Fig. 3(c), the crossover and mutation blocks are connected with a weight of 1, the crossover and selection blocks are connected with a weight of 1, and the mutation and selection blocks are connected with a weight of 1. It implies that the crossover block outputs 100 solutions, all of which are passed to both the mutation and selection blocks; then, the solutions are passed from the mutation block to the selection block. In contrast to Fig. 3(b), the selection block in Fig. 3(c) receives 200 rather than 100 solutions.


Adjacency matrix
Crossover (100 dec vectors)
(Weight: 1)
Mutation (100 dec vectors)
(Weight: 1)
Selection (100 solutions)

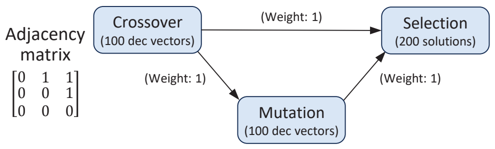
$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

(a) Solutions are passed from Crossover through Mutation to Selection.


Adjacency matrix
Crossover (100 dec vectors)
(Weight: 0.5)
Selection (100 solutions)
(Weight: 0.5)
(Weight: 1)
Mutation (50 dec vectors)

$$\begin{bmatrix} 0 & 0.5 & 0.5 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

(b) Solutions are equally divided and passed from Crossover to Mutation and Selection, then the solutions are passed from Mutation to Selection.


Adjacency matrix
Crossover (100 dec vectors)
(Weight: 1)
Selection (200 solutions)
(Weight: 1)
(Weight: 1)
Mutation (100 dec vectors)

$$\begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

(c) Solutions are duplicated and passed from Crossover to both Mutation and Selection, then the solutions are passed from Mutation to Selection.

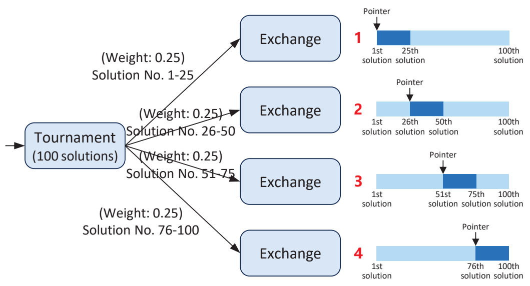
Fig. 3. Examples of three blocks with different connections.

As a consequence, Fig. 3(a) mimics genetic algorithms mutating all solutions, Fig. 3(b) is more conservative that mutates only half the solutions, and Fig. 3(c) doubles the number of function evaluations to save the better one between each solution before and after mutation. These distinct search behaviors can be achieved by merely adjusting the edges between blocks, showing high versatility in creating diverse EAs.

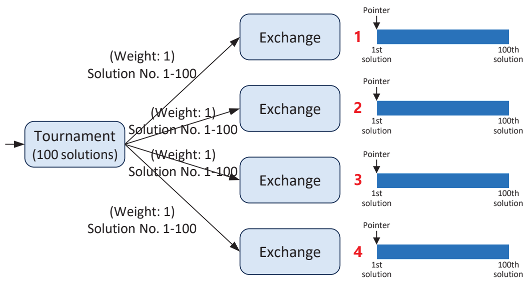
C. Delicate Considerations

In addition to the designs of blocks and edges, the proposed framework incorporates the following three delicate considerations to facilitate the flow of solutions:

- *Solution pointer.* According to Figs. 3(b)-(c), when a block passes solutions to multiple successors, the solutions may be divided or duplicated depending on the weights of edges. To unify these two scenarios, a solution pointer is implicitly involved in each block, enabling it to pass solutions from the pointer and increase the pointer afterwards. As illustrated in Fig. 4(a), after the tournament block outputs 100 solutions, it saves these solutions and initializes a pointer to 1. This way, the 1st-25th solutions are passed to the first successor, the 26th-50th solutions are passed to the second successor, and so on. Note that the pointer will be reset to 1 when it exceeds the maximum number of solutions, which enables the passing of all 100 solutions to the four successors in Fig. 4(b) and even more complex scenarios.
- *Solution interleaving.* While a block may have multiple successors, it may also have multiple predecessors in complex architectures. If the solutions received from multiple predecessors are directly concatenated as shown in Fig. 5(a), the crossover block will use the solutions from the same predecessor to generate offspring solutions, which highly reduces the exploration ability. Therefore, the solutions received from multiple predecessors are actually interleaved as shown in Fig. 5(b).
- *Delayed evaluation.* In Table I, the population, tournament, and selection blocks save or select solutions based on their objective values requiring function evaluations. By contrast, the crossover, mutation, exchange, and kopt operators generate new solutions based on parent solutions without function evaluation. It means that the solutions output by a crossover block need not be evaluated when passed to a mutation block and need to be evaluated when passed to a selection block. To unify these two scenarios, the successors of a block ask it to evaluate solutions before passing them if necessary. As depicted in Fig. 6, the solutions need not be evaluated when passed to the crossover and mutation blocks, so only decision variables are actually received. When the

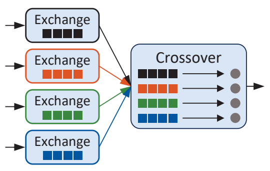

Tournament (100 solutions)
(Weight: 0.25) Solution No. 1-25
(Weight: 0.25) Solution No. 26-50
(Weight: 0.25) Solution No. 51-75
(Weight: 0.25) Solution No. 76-100
Exchange
Exchange
Exchange
Exchange
Pointer
1
2
3
4
1st solution
25th solution
26th solution
50th solution
51st solution
75th solution
76th solution
100th solution

(a) Four edges with weights 0.25, 0.25, 0.25, and 0.25, where 100 solutions are divided and passed to four exchange blocks.

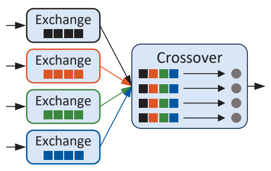

Tournament (100 solutions)
(Weight: 1) Solution No. 1-100
(Weight: 1) Solution No. 1-100
(Weight: 1) Solution No. 1-100
(Weight: 1) Solution No. 1-100
Exchange
Exchange
Exchange
Exchange
Pointer
1
2
3
4
1st solution
100th solution

(b) Four edges with weights 1, 1, 1, and 1, where 100 solutions are duplicated and passed to four exchange blocks.

Fig. 4. Examples of passing different ratios of solutions to four blocks.



(a) Solutions from four predecessors are concatenated (unexpected).



(b) Solutions from four predecessors are interleaved (expected).

Fig. 5. Examples of the concatenation and interleaving of solutions from multiple predecessors.

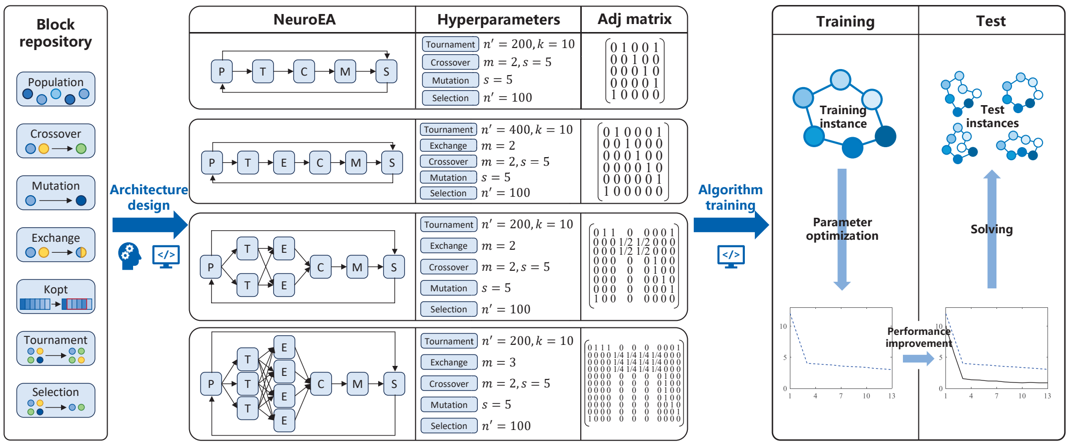


Fig. 8. Procedure of using NeuroEAs to solve specific optimization problems.

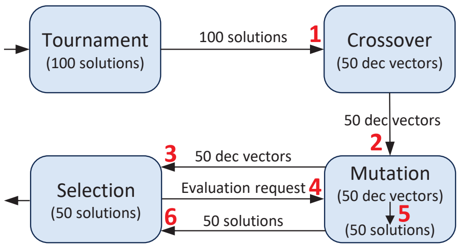


Fig. 6. An example of passing solutions with delayed evaluation.

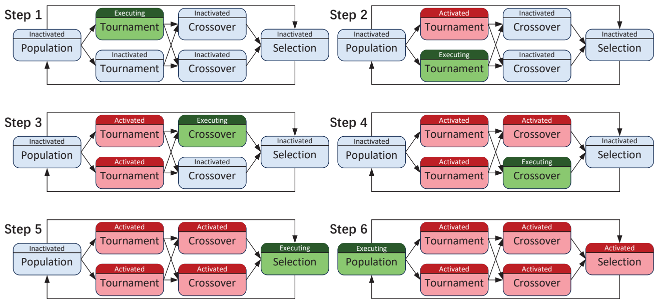


Fig. 7. An example of the execution procedure of NeuroEAs. The inactivated blocks, activated blocks, and the block being executed are shown in different colors.

mutation block attempts to pass solutions to the selection block, the latter asks the former to evaluate solutions, so evaluated solutions are received at last.

The designs of blocks and edges enable the thorough parameterization of EAs, and the above delicate considerations enable them to automatically execute like iterative optimizers. The execution procedure of NeuroEAs is elaborated in the next subsection.

D. Execution Procedure of NeuroEAs

As illustrated in Fig. 7, all the blocks of a NeuroEA are flagged as *inactivated* at the beginning of each generation. Then, the first inactivated block whose predecessors are all population blocks or activated blocks receives solutions from its predecessors. Afterwards, the block generates output solutions and changes its flag to *activated*. This procedure repeats until all blocks are flagged as *activated*, which means that all blocks have been executed at the current generation. Algorithms S1 and S2 in the Supplementary Materials present the pseudocode of the execution procedure of NeuroEAs. In short, NeuroEAs can be regarded as workflows represented by graphs.

E. Creating and Training NeuroEAs

The procedure of using NeuroEAs in practice consists of two core steps as depicted in Fig. 8, including architecture design and algorithm training. The step of architecture design refers to creating a NeuroEA represented by a graph $G(P|V,E)$, where engineers should determine the set of blocks V, the adjacency matrix E, and the hyperparameters of each block. This procedure is similar to creating neural networks, which can also be automated through hyperparameter optimization and architecture search techniques [48], [49].

Since the hyperparameters of blocks have explicit semantic roles but the parameters do not, the latter have to undergo tuning via a training procedure. Similar to Eq. (1), here the algorithm training is conceptualized as the following optimization task:

$$\min_{\theta} \; metric(f(G(P|V_\theta,E))) \;, \tag{12}$$

where the graph $G(P|V,E)$ receives a randomly initialized population P and outputs a new population after multiple generations, each node in V denotes a block, each edge in E denotes the flow of solutions between two nodes, θ includes the parameters of all the blocks in V, and $metric$ denotes the performance metric (e.g., minimum objective value for single-objective optimization and negative of hypervolume for multi-objective optimization) of the output population on the training problem f. Considering that the metric value exhibits variability particularly on multimodal problems, each algorithm is executed for three independent runs and the mean metric value is recorded.

In our experiments, we manually create several NeuroEAs with different architectures and employ a sim-

TABLE II
COMPARED ALGORITHMS AND TEST PROBLEMS INVOLVED IN SIX EXPERIMENTS

No.	Experiment	Test problems	Compared algorithms (excluding NeuroEA)	#NeuroEA wins/ #total instances
1	Performance of NeuroEAs with different architectures and training problems	f_1–f_9 [50]	Multiple NeuroEAs	–
2	Single-objective optimization	f_1–f_{13} [50], f_{14}–f_{30} (BBOB) [55]	Metaheuristics: GA [1], PSO [2], ABC [51], CMA-ES [52], DE [53]	37/60
			Metaheuristics with automated design methods: GA+ARSBX [54], IMODE [56], LDE [14], AutoV [18], ReEvo [57]	
			Gradient methods: Adam [58], BFGS [59], FRCG [60], RMSProp [61], SQP [62]	45/60
3	Expensive single-objectiveoptimization	f_1–f_{13} [50], f_{14}–f_{30} (BBOB) [55]	Surrogate-assisted metaheuristics: SACOSO [63], SACC-EAM-II [64], SADE-Sammon [65], SAMSO [66], L2SMEA [67]	45/60
4	Combinatorial optimization	Knapsack problems (KPs) [68], travelling salesman problems (TSPs) [71]	Metaheuristics: GA [1], PSO [2]	7/8
			Metaheuristics with automated design methods: IMODE [56]	
			Combinatorial metaheuristics: BSPGA [69], ACO [70]	
5	Multi-objective optimization	WFG1–WFG9 [75], IMF1–IMF10 [80]	MOEAs: NSGA-II [72], MOEA/D-DE [73], LMOCSO [74], S^3-CMA-ES [76], MOBCA [77], FDV [78], SSCEA [79]	27/38
			MOEAs with automated design methods: MOEA/D-FRRMAB [16], MOEA/D-DYTS [81], AutoV [18]	
6	Expensive multi-objective optimization	WFG1–WFG9 [75], IMF1–IMF10 [80]	Surrogate-assisted MOEAs: CSEA [82], ESB-CEO [83], LDS-AF [84], SSDE [85], MO-L2SMEA [67]	32/38

ple genetic algorithm to tune their parameters. These NeuroEAs are trained on a few problems and tested on all the problems. As a pioneering work for creating complex EAs, this paper does not focus on seeking optimal architectures. Nevertheless, the NeuroEAs with casually determined architectures are enough to show marked superiority compared to the state-of-the-art.

IV. EXPERIMENTAL STUDIES

A. Experimental Settings

Several experiments are conducted to compare the performance of NeuroEAs and existing algorithms. The first experiment compares the performance of the four NeuroEAs shown in Fig. 8, which verifies that the largest NeuroEA exhibits the best overall performance. Then, the other experiments compare the performance of the largest NeuroEA in Fig. 8 with representative metaheuristics and gradient methods for single-objective optimization, expensive single-objective optimization, combinatorial optimization, multi-objective optimization, and expensive multi-objective optimization. Table II details the algorithms and problems involved in the experiments.

Algorithms: The population size of all the algorithms is set to 100. For algorithm-specific parameters, all the algorithms follow their suggested parameter settings in the literature as much as possible, where the details are given in Section SII in the Supplementary Materials. The architectures and hyperparameters of NeuroEAs can be found in Fig. 8, and each NeuroEA is trained on one or two problems and tested on all the problems in each experiment using a genetic algorithm with a population size of 50 and 5 000 function evaluations. For example, the training procedure consumes a total of 10 000×5 000×3 function evaluations and about two hours on a single-objective training problem when using 14 CPU cores simultaneously.

Problems: The benchmark problems f_1–f_{13} [50] and the first 17 BBOB problems [55] (named f_{14}–f_{30} here) are adopted in single-objective optimization, where the number of variables $D = 10, 100$ for general optimization and $D = 10, 30$ for expensive optimization. The number of function evaluations is set to $1\,000\times D$ for general optimization and set to 500 for expensive optimization. Besides, the search spaces are translated by $x_i = x_i - i(u_i - l_i)/2/D$ to avoid optimal solutions locating at the origin, where $i = 1, \dots, D$, u_i is the upper bound, and l_i is the lower bound. The knapsack problems (KPs) [68] with $D = 100, 300, 500, 1\,000$ items and travelling salesman problems (TSPs) [71] with $D = 10, 30, 50, 100$ cities are adopted in combinatorial optimization, where the number of function evaluations is set to $50\times D$ for KPs and $1\,000\times D$ for TSPs. The benchmark problems WFG1–WFG9 [75] and IMF1–IMF10 [80] are adopted in multi-objective optimization, where the number of objectives $M = 2, 3$ and the number of variables $D = 10, 30$. The number of function evaluations is set to $1\,000\times D$ for general optimization and set to 500 for expensive optimization.

Metrics: The minimum objective value and inverted generational distance (IGD) [86] are adopted for single- and multi-objective optimization, respectively. Each algorithm is tested on each problem for 30 independent runs, where the mean and standard deviation of the 30 metric values are recorded. Also, the Wilcoxon rank sum test with a significance level of 0.05 is employed, where the symbols '$+$', '$-$', and '$\approx$' indicate that NeuroEAs show significantly worse, significantly better, and no statistically different performance, respectively.

TABLE IV
MINIMUM OBJECTIVE VALUES OBTAINED BY NINE NEUROEAS TRAINED ON DIFFERENT SINGLE-OBJECTIVE OPTIMIZATION PROBLEMS

Problem	NeuroEA trained on								
	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9
f_1	2.4217e-22 (3.98e-22)	4.8899e-5 (1.59e-4)	1.4458e-11 (2.10e-11)	4.2224e-14 (6.57e-14)	2.0077e-11 (2.01e-11)	7.3486e-2 (7.62e-2)	4.1301e-2 (4.56e-2)	5.4661e-19 (1.07e-18)	2.2646e-10 (8.47e-10)
f_2	5.7907e-15 (5.22e-15)	2.0634e-16 (4.19e-16)	6.1270e-8 (3.54e-8)	1.2510e-9 (7.12e-10)	3.3887e-8 (2.21e-8)	2.8528e-2 (2.23e-2)	1.4425e-2 (1.10e-2)	4.5175e-13 (3.70e-13)	6.5106e-11 (7.07e-11)
f_3	3.7949e+0 (2.32e+0)	5.0475e+1 (5.08e+1)	1.7748e-2 (1.91e-2)	2.5798e+0 (2.62e+0)	1.0968e+0 (1.15e+0)	4.2920e+1 (2.84e+1)	4.0932e+1 (2.04e+1)	4.6995e+0 (5.43e+0)	1.2213e+1 (1.32e+1)
f_4	1.5698e+0 (2.59e+0)	1.5487e+0 (1.05e+0)	5.7583e-2 (2.08e-1)	1.1317e-3 (1.29e-3)	2.7463e-1 (3.14e-1)	5.2802e-1 (2.78e-1)	8.4271e-1 (3.58e-1)	7.5540e-1 (9.20e-1)	5.0622e-1 (2.45e-1)
f_5	6.1912e+0 (3.46e+0)	4.9034e+1 (8.07e+1)	6.6302e+0 (4.24e+0)	6.7903e+0 (2.11e+0)	5.5566e+0 (3.55e+0)	1.6801e+1 (2.12e+1)	2.1683e+1 (2.64e+1)	1.7854e+1 (2.95e+1)	1.5880e+1 (2.54e+1)
f_6	0.0000e+0 (0.00e+0)	0.0000e+0 (0.00e+0)	0.0000e+0 (0.00e+0)	0.0000e+0 (0.00e+0)	0.0000e+0 (0.00e+0)	0.0000e+0 (0.00e+0)	0.0000e+0 (0.00e+0)	0.0000e+0 (0.00e+0)	0.0000e+0 (0.00e+0)
f_7	2.4551e-3 (9.83e-4)	3.1364e-3 (1.96e-3)	3.8771e-3 (1.93e-3)	2.3576e-3 (9.75e-4)	6.3259e-3 (4.11e-3)	2.8237e-3 (1.46e-3)	1.6532e-3 (1.24e-3)	3.8589e-3 (2.05e-3)	3.1825e-3 (1.78e-3)
f_8	-3.8402e+3 (1.63e+2)	-3.8556e+3 (1.41e+2)	-3.4764e+3 (2.47e+2)	-3.9530e+3 (8.05e+1)	-3.6738e+3 (2.00e+2)	-3.8136e+3 (1.73e+2)	-3.9017e+3 (1.20e+2)	-4.1391e+3 (7.65e+1)	-4.0629e+3 (1.18e+2)
f_9	4.1220e+0 (1.55e+0)	2.7265e+0 (1.27e+0)	5.6144e+0 (3.01e+0)	3.6245e+0 (1.54e+0)	2.1353e-1 (4.24e-1)	3.8560e+0 (2.18e+0)	1.0233e-1 (2.98e-1)	1.4936e+0 (1.28e+0)	2.1518e-10 (3.04e-10)

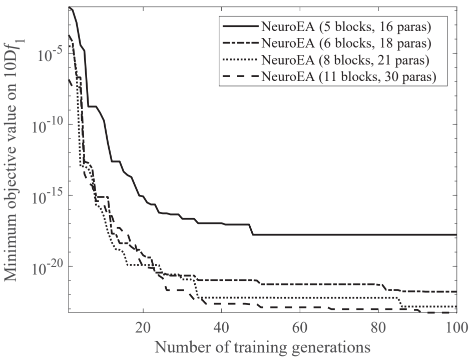


Fig. 9. Convergence profiles of the training procedures of four NeuroEAs with different architectures.

TABLE III
MINIMUM OBJECTIVE VALUES AND AVERAGE RUNTIMES OBTAINED BY FOUR NEUROEAS WITH DIFFERENT ARCHITECTURES ON SINGLE-OBJECTIVE OPTIMIZATION PROBLEMS

Problem	NeuroEA (5 blocks, 16 paras)	NeuroEA (6 blocks, 18 paras)	NeuroEA (8 blocks, 21 paras)	NeuroEA (11 blocks, 30 paras)
f_1	4.6175e-16 (9.46e-16)	7.3200e-20 (1.41e-19)	2.0441e-21 (3.41e-21)	2.4217e-22 (3.98e-22)
f_2	1.5897e-10 (1.68e-10)	1.2461e-13 (6.21e-14)	1.9495e-14 (1.69e-14)	5.7907e-15 (5.22e-15)
f_3	2.1090e+1 (2.43e+1)	2.6444e+0 (5.51e+0)	8.1900e+0 (1.06e+1)	3.7949e+0 (2.32e+0)
f_4	4.3747e+0 (3.44e+0)	1.2376e+0 (1.22e+0)	1.7513e+0 (1.46e+0)	1.5698e+0 (2.59e+0)
f_5	1.9801e+1 (4.60e+1)	4.7733e+1 (8.46e+1)	1.3826e+1 (1.81e+1)	6.1912e+0 (3.46e+0)
f_6	0.0000e+0 (0.00e+0)	0.0000e+0 (0.00e+0)	0.0000e+0 (0.00e+0)	0.0000e+0 (0.00e+0)
f_7	5.3363e-3 (2.23e-3)	3.3237e-3 (1.87e-3)	3.0430e-3 (1.96e-3)	2.4551e-3 (9.83e-4)
f_8	-3.0676e+3 (2.52e+2)	-3.6064e+3 (2.19e+2)	-3.6231e+3 (2.23e+2)	-3.8402e+3 (1.63e+2)
f_9	1.3434e+1 (5.49e+0)	5.1884e+0 (2.56e+0)	5.1169e+0 (2.49e+0)	4.1220e+0 (1.55e+0)
f_{10}	3.0162e-7 (1.08e-6)	2.5322e-11 (2.52e-11)	4.4505e-11 (9.25e-11)	1.0984e-11 (1.74e-11)
f_{11}	1.7727e-1 (1.74e-1)	5.2078e-2 (2.83e-2)	2.8298e-2 (1.96e-2)	2.6897e-2 (1.39e-2)
f_{12}	3.1262e-1 (9.75e-1)	1.4810e-2 (3.76e-2)	3.3757e-14 (1.26e-13)	1.7707e-21 (4.60e-21)
f_{13}	3.1392e-3 (5.15e-3)	2.3544e-3 (4.68e-3)	7.8481e-4 (2.94e-3)	2.6587e-21 (3.57e-21)
Average runtime	1.3559s	3.2724s	3.3662s	3.4967s

B. Comparisons Between Different NeuroEAs

The four NeuroEAs shown in Fig. 8 are trained on the single-objective optimization problem f_1 with 10 variables, where the convergence profiles of the training procedures are depicted in Fig. 9. In spite of their different architectures, all of them evolve a population with 100 solutions by generating 100 offspring solutions at each generation. It can be observed that the NeuroEAs with more blocks and parameters converge faster. Moreover, Table III presents the minimum objective values and average runtimes obtained by the four trained NeuroEAs on f_1-f_{13} with 10 variables, where the NeuroEAs are trained on only f_1. It can be found that the largest NeuroEA with 11 blocks and 30 parameters exhibits the best overall performance, which linearly increases the computational complexity but improves the convergence performance without using more function evaluations. While most existing EAs adopt sequential architectures containing a few components, this experiment reveals the bright prospect of creating non-sequential EAs with many blocks. Therefore, the largest architecture in the four NeuroEAs is employed in the following.

To study the transferability of NeuroEAs, Table IV lists the minimum objective values obtained by nine NeuroEAs (11 blocks, 30 parameters) on f_1-f_9 with 10 variables, where each NeuroEA is trained on one of f_1-f_9. According to the experimental results, it can be found that the NeuroEAs always exhibit the best performance on their training problems, while they obtain competitive performance across the other problems. This observation verifies the transferability of NeuroEAs and is also consistent with the no free lunch theorem. Nevertheless, how to measure the similarities between problems and determine whether a NeuroEA trained on a problem is effective for another problem remains a challenging issue. For simplicity, the NeuroEAs in the following experiments are trained on only one or two problems and tested on all the problems, which is enough to outperform the state-of-the-art.

C. Experiment of Single-Objective Optimization

Table V lists the minimum objective values obtained by five metaheuristics, five metaheuristics with automated design methods, and a NeuroEA on f_1-f_{30} with 10 and 100 variables, where the NeuroEA is first trained on 10-variable f_1 and then trained on 10-variable f_9 to consider both unimodal and multimodal landscapes. It can be observed that the NeuroEA exhibits the best overall performance, which gains 37 best results. Besides, the metaheuristics with automated design methods hold better overall performance than those without automated design methods, and the proposed framework for creating NeuroEAs is more effective than existing automated design methods. In particular, the NeuroEA obtains much lower objective values than the others on unimodal problems like f_1 and f_2, which verifies its significantly faster convergence speed.

Furthermore, Table VI lists the minimum objective values obtained by five gradient methods and a NeuroEA on the same problems, where the NeuroEA also outperforms the others on 45 out of 60 problem instances. In particular, the NeuroEA outperforms the others on

TABLE V
MINIMUM OBJECTIVE VALUES OBTAINED BY A NEUROEA, FIVE METAHEURISTICS, AND FIVE AUTOMATED METAHEURISTICS ON 30 SINGLE-OBJECTIVE OPTIMIZATION PROBLEMS WITH 10 & 100 VARIABLES AND 10 000 & 100 000 FUNCTION EVALUATIONS

Problem	D	GA	PSO	ABC	CMA-ES	DE	GA+ARSBX	IMODE	LDE	AutoV	ReEvo	NeuroEA
f_1	10	5.8061e-2 (3.21e-2) -	6.3815e+1 (9.87e+1) -	5.2171e-1 (2.50e-1) -	3.4014e+4 (6.42e+3) -	8.0101e+1 (2.99e+1) -	4.5568e-2 (4.17e-2) -	1.0000e+3 (0.00e+0) -	2.4693e-6 (1.29e-6) -	4.7182e-9 (1.53e-8) -	2.6112e-2 (3.23e-2) -	4.8105e-17 (9.23e-17)
	100	5.9842e-1 (1.17e-1) -	2.3002e+5 (1.99e+4) -	6.4034e+4 (8.71e+3) -	7.1761e+3 (2.90e+3) -	6.1206e+3 (1.60e+3) -	5.9700e-1 (1.29e-1) -	1.0000e+4 (0.00e+0) -	4.9813e+2 (1.02e+2) -	9.9854e-6 (1.47e-5) -	6.4084e-1 (1.73e-1) -	2.7319e-12 (9.48e-12)
f_2	10	2.6804e-2 (9.18e-3) -	1.8256e+0 (2.96e+0) -	1.3915e-1 (4.32e-2) -	5.0000e+1 (0.00e+0) -	3.5272e+0 (6.39e-1) -	2.4332e-2 (9.66e-3) -	1.1000e+1 (0.00e+0) -	6.2144e-4 (1.90e-4) -	4.1303e-6 (7.01e-7) -	1.3967e-2 (7.34e-3) -	5.1986e-12 (6.60e-12)
	100	3.6480e-1 (4.69e-2) -	4.5686e+2 (6.04e+1) -	1.9252e+2 (9.35e+0) -	2.7308e+1 (1.08e+1) -	7.3188e+1 (8.82e+0) -	3.6467e-1 (3.59e-2) -	1.0100e+2 (0.00e+0) -	1.7821e+1 (1.71e+0) -	2.0853e-4 (8.35e-6) -	6.4084e-1 (1.73e-1) -	2.9404e-11 (1.27e-10)
f_3	10	3.9189e+1 (3.05e+1) -	1.0713e+3 (1.20e+3) -	2.0682e+2 (8.39e+1) -	5.4294e+4 (4.44e+3) -	3.3300e+2 (8.37e+1) -	4.2277e+1 (3.81e+1) -	3.8500e+4 (0.00e+0) -	6.2528e-1 (3.27e-1) +	3.1339e+1 (3.00e+1) -	4.8062e+1 (3.18e+1) -	5.2965e+0 (7.27e+0)
	100	2.6520e+4 (4.28e+3) -	2.9423e+7 (1.10e+7) -	5.6263e+5 (9.53e+4) -	2.4235e+5 (5.01e+4) -	1.4430e+5 (2.08e+4) -	2.6827e+4 (3.44e+3) -	3.3835e+7 (0.00e+0) -	5.2622e+4 (6.47e+3) -	1.3057e+4 (2.72e+3) +	7.4528e+4 (7.87e+3) -	1.4565e+4 (2.75e+3)
f_4	10	8.3878e-1 (2.98e-1) -	1.4833e+1 (8.79e+0) -	3.0167e+0 (8.05e-1) -	7.7973e+1 (4.12e+1) -	1.1082e+1 (2.12e+0) -	8.0054e-1 (2.39e-1) -	1.0000e+1 (0.00e+0) -	2.6016e-2 (2.05e-2) +	1.2258e+0 (1.24e+0) -	1.1029e+0 (3.68e-1) -	1.0910e-1 (1.22e-1)
	100	4.4087e+1 (5.86e+0) -	1.0400e+2 (6.00e+0) -	1.1020e+2 (6.00e+0) -	5.5758e+1 (1.43e+1) -	6.6306e+1 (4.23e+0) -	4.1187e+1 (7.52e+0) -	1.0000e+1 (0.00e+0) +	1.4284e+1 (1.00e+0) +	1.0979e+1 (1.52e+0) +	5.5096e+1 (7.93e+0) -	2.9625e+1 (3.30e+0)
f_5	10	1.2016e+1 (1.55e+1) -	2.8429e+4 (6.41e+4) -	2.5937e+1 (8.01e+0) -	1.4433e+8 (6.32e+7) -	2.8022e+3 (1.80e+3) -	8.0558e+0 (4.16e+0) -	3.2436e+4 (0.00e+0) -	7.3237e+0 (5.75e-1) -	7.3904e+0 (1.34e+0) -	3.5663e+1 (8.15e+1) -	5.8534e+0 (1.88e+0)
	100	3.1360e+2 (5.56e+1) -	1.1431e+9 (1.31e+8) -	1.9017e+8 (4.14e+7) -	8.0215e+6 (5.40e+6) -	4.9301e+6 (2.10e+6) -	3.4397e+2 (5.30e+1) -	3.5680e+5 (0.00e+0) -	2.8719e+4 (9.03e+3) -	1.8557e+2 (6.26e+1) =	8.1339e+2 (8.86e+2) -	1.8466e+2 (4.83e+1)
f_6	10	0.0000e+0 (0.00e+0) =	6.1400e+1 (1.03e+2) -	3.6667e-1 (5.56e-1) -	3.5201e+4 (3.14e+3) -	8.1700e+1 (2.41e+1) -	0.0000e+0 (0.00e+0) =	1.0000e+3 (0.00e+0) -	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0)
	100	0.0000e+0 (0.00e+0) =	2.3175e+5 (2.01e+4) -	6.4337e+4 (9.87e+3) -	6.0766e+3 (2.17e+3) -	6.2215e+3 (1.12e+3) -	0.0000e+0 (0.00e+0) =	1.0000e+4 (0.00e+0) -	5.3100e+2 (1.03e+2) -	1.1333e+0 (1.11e+0) -	6.6667e-2 (2.49e-1) -	0.0000e+0 (0.00e+0)
f_7	10	1.0076e-2 (4.77e-3) -	6.1080e-2 (4.57e-2) -	2.9980e-2 (1.50e-2) -	1.6246e+0 (3.84e+0) -	4.0842e-2 (1.74e-2) -	9.7745e-3 (3.59e-3) -	1.4843e-2 (7.92e-5) -	5.1477e-3 (1.98e-3) -	5.4419e-3 (4.29e-3) =	4.1698e-2 (1.72e-2) -	3.8250e-3 (1.76e-3)
	100	9.1907e-2 (1.54e-2) =	2.1220e+3 (4.77e+2) -	1.7000e+2 (2.77e+1) -	1.9435e+1 (7.28e+0) -	3.6905e+0 (1.59e+0) -	1.0086e-1 (1.61e-2) -	1.3556e+0 (1.66e-5) -	2.4543e-1 (4.72e-2) -	7.3410e-2 (1.34e-2) =	1.4115e-1 (2.58e-2) -	7.0719e-2 (2.14e-2)
f_8	10	-6.3448e+3 (1.14e+2) -	-4.3535e+3 (5.40e+2) -	-5.7425e+3 (2.76e+2) -	-3.1752e+3 (9.26e+2) -	-4.3979e+3 (2.68e+2) -	-6.3645e+3 (1.14e+2) -	-3.5443e+2 (1.16e-13) -	-4.5960e+3 (1.23e+2) -	-6.2860e+3 (1.83e+2) -	-5.5931e+3 (2.50e+2) -	-6.4303e+3 (2.16e+1)
	100	-5.8710e+4 (9.94e+2) -	-2.5247e+4 (2.26e+3) -	-3.7035e+4 (1.94e+3) -	-3.8098e+4 (2.57e+3) -	-2.2550e+4 (1.33e+3) -	-5.8397e+4 (1.05e+3) -	-3.5443e+3 (9.25e-13) -	-2.4240e+4 (1.24e+3) -	-6.1019e+4 (7.87e+2) -	-4.6882e+4 (2.54e+3) -	-6.3774e+4 (2.36e+2)
f_9	10	5.7752e-2 (1.85e-1) +	2.7236e+1 (8.98e+0) -	2.9918e+1 (6.29e+0) -	4.9299e+1 (3.01e+1) -	4.6896e+1 (8.28e+0) -	5.7577e-2 (1.95e-1) +	2.0234e+2 (5.78e-14) -	1.2730e+1 (2.43e+0) -	2.7475e+0 (1.30e+0) -	1.1169e-2 (8.65e-3) +	7.1574e-2 (2.53e-1)
	100	8.9046e-1 (6.43e-1) -	1.2954e+3 (7.11e+1) -	1.1660e+3 (4.47e+1) -	2.9309e+2 (1.64e+2) -	1.1242e+3 (4.09e+1) -	9.9390e-1 (7.39e-1) -	2.0234e+3 (2.31e-13) -	7.6414e+2 (2.47e+1) -	3.7523e+1 (5.71e+0) -	1.6176e+2 (2.71e+0) -	5.6728e-1 (7.75e-1)
f_{10}	10	9.7158e-2 (3.74e-2) -	4.2872e+0 (2.34e+0) -	9.3873e-1 (3.55e-1) -	1.9092e+1 (5.07e+0) -	6.8466e+0 (8.87e-1) -	8.7896e-2 (4.37e-2) -	1.0810e+1 (0.00e+0) -	9.6811e-4 (1.75e-4) -	1.5085e-5 (6.43e-6) -	2.2364e-1 (9.43e-2) -	1.7899e-9 (1.67e-9)
	100	1.1240e-1 (1.18e-2) -	2.0158e+1 (9.10e-2) -	1.8524e+1 (3.56e-1) -	1.1163e+1 (3.88e+0) -	1.5080e+1 (1.57e+0) -	1.0924e-1 (1.45e-2) -	1.0810e+1 (1.81e-15) -	4.4537e+0 (1.75e-1) -	1.3961e-3 (1.87e-3) -	6.3401e+0 (2.71e+0) -	1.1902e-14 (3.53e-15)
f_{11}	10	8.5535e-2 (3.64e-2) -	8.5988e-1 (1.09e+0) -	7.8587e-1 (1.15e-1) -	2.6598e+2 (1.18e+2) -	1.7269e+0 (2.89e-1) -	1.0871e-1 (4.52e-2) -	1.0000e+1 (0.00e+0) -	1.6456e-1 (6.95e-2) -	2.7803e-2 (2.09e-2) =	9.0243e-2 (3.57e-2) -	2.3133e-2 (1.72e-2)
	100	3.0410e-1 (4.68e-2) -	2.0699e+3 (1.81e+2) -	5.8834e+2 (7.43e+1) -	6.6656e+1 (2.39e+1) -	5.9818e+1 (1.23e+1) -	2.9241e-1 (4.86e-2) -	9.1000e+1 (0.00e+0) -	5.7050e+0 (7.85e-1) -	5.0767e-3 (1.36e-2) +	1.7840e-1 (3.97e-2) -	1.6805e-2 (3.67e-2)
f_{12}	10	1.0940e-1 (7.57e-3) -	2.2916e+1 (4.20e+1) -	1.1610e+0 (5.47e-1) -	4.8611e+8 (3.28e+8) -	2.5128e+1 (9.96e+0) -	1.1167e-1 (1.59e-2) -	1.6755e+1 (7.23e-15) -	1.4061e-1 (5.82e-2) -	1.0472e-1 (8.68e-7) +	7.9707e-3 (2.12e-2) +	1.0818e-1 (1.89e-2)
	100	1.6951e-1 (1.30e-2) -	2.7429e+9 (6.10e+8) -	4.0768e+8 (1.18e+8) -	1.3188e+7 (1.56e+7) -	1.3109e+6 (1.39e+6) -	1.7766e-1 (3.09e-2) -	1.6755e+2 (2.89e-14) -	2.7821e+2 (6.03e+1) -	8.7632e-1 (1.02e+0) -	5.2095e-2 (4.29e-2) +	1.6495e-1 (4.97e-2)
f_{13}	10	1.0582e-1 (6.22e-3) -	1.2178e+4 (5.90e+4) -	3.3171e-1 (1.19e-1) -	8.2504e+8 (2.90e+8) -	6.5312e+0 (2.64e+0) -	1.0575e-1 (6.60e-3) -	1.6000e+1 (0.00e+0) -	8.9045e-3 (3.68e-3) +	1.0180e-1 (4.94e-3) -	5.2094e-3 (5.42e-3) +	1.0073e-1 (2.79e-3)
	100	1.5296e-1 (8.70e-3) -	4.7449e+9 (8.98e+8) -	8.3234e+8 (2.05e+8) -	2.3332e+7 (1.47e+7) -	6.5485e+6 (3.67e+6) -	1.5067e-1 (5.93e-3) -	1.6000e+2 (0.00e+0) -	6.4822e+1 (1.60e+1) -	1.3047e-1 (9.40e-3) -	3.3697e-2 (5.79e-3) +	1.2666e-1 (5.20e-3)
f_{14}	10	2.0975e+1 (4.34e-5) -	2.1706e+1 (1.08e+0) -	2.0976e+1 (4.88e-4) -	2.7963e+1 (7.23e-15) -	2.1104e+1 (7.52e-2) -	1.4558e-4 (1.23e-4) -	2.0975e+1 (5.10e-12) -	2.6291e-8 (1.17e-8) -	2.2147e-11 (6.66e-11) -	1.1399e-5 (9.84e-6) -	2.7935e-19 (1.06e-18)
	100	1.8273e+2 (9.85e-5) -	2.1257e+3 (1.39e+2) -	1.8902e+2 (1.32e+0) -	2.1128e+2 (3.67e+0) -	2.1722e+2 (7.08e+0) -	1.4228e-3 (2.17e-4) -	1.8273e+2 (5.77e-6) -	1.1100e+0 (1.81e-1) -	4.5400e-8 (9.13e-8) -	3.9881e-4 (9.91e-5) -	3.2920e-14 (1.72e-13)
f_{15}	10	9.9793e+6 (1.56e-1) -	1.0004e+7 (2.52e+4) -	9.9793e+6 (1.28e-1) -	9.9793e+6 (2.91e+0) -	9.9793e+6 (8.35e-1) -	4.3338e-1 (3.76e-1) -	9.9793e+6 (3.76e-9) -	2.2060e-1 (8.85e-2) -	7.8014e-8 (1.69e-7) -	4.4621e-2 (3.74e-2) -	4.0719e-16 (1.56e-15)
	100	5.2603e+7 (2.68e-2) -	1.1822e+8 (1.64e+7) -	5.2604e+7 (2.42e+1) -	5.2604e+7 (4.77e+1) -	5.2715e+7 (3.01e+4) -	1.5567e+1 (6.57e+0) -	5.2603e+7 (2.19e-3) -	1.5756e-3 (1.55e-3) -	2.7512e-4 (4.05e-4) -	4.5234e+0 (2.58e+0) -	1.1015e-12 (1.24e-12)
f_{16}	10	1.8797e+2 (2.19e-2) -	2.6326e+2 (4.15e+1) -	1.9728e+2 (3.05e+0) -	2.8233e+2 (5.47e+0) -	2.2519e+2 (7.79e+0) -	8.0188e-2 (1.90e-1) +	1.8800e+2 (2.52e-1) -	1.7273e+1 (7.94e+0) -	2.7709e+0 (1.62e+0) -	5.6321e-2 (1.82e-1) +	2.3217e-1 (5.01e-1)
	100	1.4960e+3 (6.05e-1) -	1.0239e+4 (7.47e+2) -	2.1787e+3 (4.39e+1) -	2.2629e+3 (4.03e+1) -	2.7754e+3 (1.02e+2) -	1.2476e+0 (6.12e-1) +	1.5121e+3 (5.38e+0) -	9.4016e+2 (5.95e+0) -	4.1682e+1 (4.84e+0) -	1.5169e+2 (2.90e+1) -	4.2466e+0 (2.15e+0)
f_{17}	10	1.8814e+2 (3.40e-1) -	2.9355e+2 (3.95e+1) -	2.0306e+2 (4.53e+0) -	4.7425e+2 (3.77e+1) -	2.3866e+2 (9.05e+0) -	4.2376e+0 (4.49e-1) -	1.8813e+2 (4.05e-1) -	5.0216e+1 (8.93e+0) -	6.6088e+0 (2.25e+0) -	2.8544e-1 (4.40e-1) +	9.8076e-1 (7.72e-1)
	100	1.4971e+3 (1.18e+0) -	1.0955e+4 (7.46e+2) -	2.6015e+3 (7.25e+1) -	2.2144e+3 (5.75e+1) -	2.9200e+3 (1.32e+2) -	2.4599e+1 (1.66e+0) -	1.5310e+3 (8.60e+0) -	1.0971e+3 (1.65e+1) -	8.3336e+1 (7.97e+0) -	8.4866e+0 (2.72e+0) +	1.3853e+1 (3.28e+0)
f_{18}	10	1.2391e+2 (4.47e-11) -	1.2749e+2 (4.50e+0) -	1.2391e+2 (2.55e-7) -	1.2391e+2 (4.34e-14) -	1.2393e+2 (8.71e-2) -	1.2391e+2 (7.94e-11) -	1.2391e+2 (4.34e-14) -	5.4313e+1 (5.79e+0) -	1.2391e+2 (4.34e-14) -	5.4192e+1 (1.97e-3) -	0.0000e+0 (0.00e+0)
	100	5.7574e+2 (1.79e-13) -	1.2392e+3 (4.28e+1) -	5.7606e+2 (2.23e-1) -	5.7574e+2 (2.31e-13) -	6.2103e+2 (1.26e+1) -	5.7574e+2 (1.27e-13) -	5.7574e+2 (1.48e-12) -	1.4940e+3 (2.68e+1) -	5.7574e+2 (3.15e-6) -	6.3034e+2 (2.03e+0) -	0.0000e+0 (0.00e+0)
f_{19}	10	1.8172e+4 (1.85e+3) -	3.5381e+4 (1.33e+4) -	1.7319e+4 (4.23e+3) -	2.4680e+4 (2.36e+4) -	1.5645e+4 (3.76e+2) -	1.2160e-2 (6.45e-3) -	1.5029e+4 (4.71e-3) -	6.1102e-3 (2.48e-3) -	6.8455e-2 (4.64e-2) -	4.3354e-3 (2.20e-3) -	1.0912e-4 (1.36e-4)
	100	4.5882e+3 (1.24e+3) -	2.4514e+6 (2.55e+5) -	8.9708e+4 (2.35e+4) -	5.3207e+4 (2.20e+4) -	8.2070e+4 (2.49e+4) -	1.7095e-1 (5.37e-2) -	2.8467e+3 (5.55e+2) -	2.7385e-1 (8.50e-2) -	1.5695e+3 (1.25e+2) -	1.5823e-1 (2.07e-2) -	6.6964e-5 (9.51e-5)
f_{20}	10	1.9484e+2 (8.34e-3) -	2.2172e+2 (1.95e+1) -	1.9487e+2 (2.48e-2) -	2.9507e+2 (6.99e-1) -	1.9540e+2 (2.74e-1) -	5.6374e-1 (5.71e-1) -	1.9483e+2 (6.07e-9) -	9.4943e-1 (3.95e-1) -	4.8416e-1 (3.95e-1) -	5.6549e+0 (2.53e+0) -	1.2915e-1 (1.59e-1)
	100	1.8092e+3 (3.38e-2) -	1.6961e+4 (1.47e+3) -	1.9445e+3 (2.55e+1) -	2.0055e+3 (6.22e+1) -	2.3462e+3 (1.19e+2) -	8.1979e+0 (1.59e+0) -	1.8092e+3 (8.37e-3) -	3.1437e+1 (1.09e+1) -	8.8272e+0 (1.96e+0) -	1.2147e+2 (3.44e+1) -	6.0379e-1 (2.35e-1)
f_{21}	10	2.2952e+3 (9.97e-1) -	2.8230e+3 (1.34e+3) -	2.2973e+3 (8.77e-1) -	8.8842e+3 (5.55e-12) -	2.3125e+3 (5.74e+0) -	5.6137e+0 (2.54e+0) =	2.2936e+3 (2.54e-2) -	6.4370e+0 (5.05e-1) -	7.2369e+0 (7.48e-1) -	6.4129e+0 (2.15e+0) -	5.2538e+0 (2.18e+0)
	100	1.0330e+5 (2.99e+1) -	1.4402e+7 (1.57e+6) -	1.2378e+5 (4.54e+3) -	1.4797e+5 (7.80e+3) -	1.2614e+5 (1.16e+4) -	2.0609e+2 (4.96e+1) -	1.0327e+5 (1.35e+0) -	8.9037e+2 (8.42e+1) -	1.2034e+2 (2.49e+1) +	2.2644e+2 (5.85e+1) -	1.7332e+2 (3.79e+1)
f_{22}	10	1.5305e+4 (4.27e-2) -	1.8087e+4 (3.55e+3) -	1.5306e+4 (8.43e-1) -	2.0724e+4 (1.11e-11) -	1.6184e+4 (9.41e+2) -	2.1720e+1 (1.74e+1) -	1.5305e+4 (1.04e-11) -	3.1584e+1 (6.72e+0) -	2.2862e+1 (1.78e+1) -	3.2465e+1 (3.44e+1) -	1.8662e+1 (2.02e+1)
	100	5.7706e+5 (9.97e-2) -	4.6312e+8 (8.62e+7) -	5.7764e+5 (1.92e+2) -	5.8263e+5 (3.81e+3) -	9.8161e+6 (2.96e+6) -	2.1362e+2 (8.95e+1) -	5.7706e+5 (2.43e-7) -	1.6089e+3 (1.36e+2) -	1.8128e+2 (6.21e+1) -	3.7185e+2 (1.91e+2) -	1.2933e+2 (3.11e+1)
f_{23}	10	4.1511e+3 (2.63e+3) -	8.8397e+3 (9.11e+3) -	2.4509e+4 (1.46e+4) -	3.8886e+5 (2.02e+5) -	3.7426e+3 (1.22e+3) -	5.6345e+3 (4.73e+3) -	4.8630e+3 (1.28e+3) -	1.4418e+4 (6.44e+3) -	3.4078e+3 (2.07e+3) -	6.2959e+3 (2.55e+3) -	2.1727e+3 (8.53e+2)
	100	1.9089e+5 (4.09e+4) -	4.2009e+6 (1.17e+6) -	8.6529e+6 (2.22e+6) -	2.0067e+5 (1.27e+5) -	2.5478e+5 (4.05e+4) -	1.8770e+5 (4.39e+4) -	1.1078e+5 (8.64e+3) -	5.1271e+6 (7.10e+5) -	1.1067e+5 (3.15e+4) -	1.9982e+5 (5.07e+4) -	6.9918e+4 (2.11e+4)
f_{24}	10	2.6053e+6 (1.11e-1) -	4.6892e+6 (2.60e+6) -	2.6054e+6 (5.13e+1) -	2.6053e+6 (0.00e+0) -	3.4850e+6 (9.69e+5) -	2.7379e+1 (1.29e+1) =	2.6053e+6 (0.00e+0) -	2.1487e+1 (7.69e+0) +	2.6193e+1 (1.04e+1) =	4.6575e+1 (2.08e+1) -	2.9609e+1 (9.39e+0)
	100	2.3359e+7 (4.81e-3) -	1.1876e+9 (3.74e+7) -	2.3369e+7 (3.03e+3) -	2.3359e+7 (1.89e-8) -	2.5711e+8 (2.24e+7) -	3.7388e+2 (3.71e+1) -	2.3359e+7 (1.84e-8) -	4.0718e+2 (3.69e+1) -	2.9728e+2 (4.39e+1) -	3.0930e+2 (3.51e+1) -	2.3306e+2 (2.74e+1)
f_{25}	10	1.8364e+7 (5.96e+2) -	2.5776e+7 (1.23e+7) -	1.8374e+7 (5.09e+3) -	3.8790e+7 (1.52e-8) -	1.8611e+7 (1.49e+5) -	8.6852e+2 (2.96e+3) =	1.8363e+7 (3.23e-3) -	2.4757e+1 (1.05e+1) +	6.9465e+2 (2.80e+3) =	3.7177e+2 (1.68e+3) +	6.1946e+2 (1.09e+3)
	100	1.9346e+8 (2.12e+2) -	2.6833e+9 (1.98e+8) -	2.1151e+8 (3.69e+6) -	2.2696e+8 (6.73e+6) -	2.3481e+8 (9.89e+6) -	4.0083e+5 (1.77e+5) -	1.9346e+8 (1.86e+1) -	2.8194e+6 (5.87e+5) -	1.7486e+5 (1.46e+5) =	1.9168e+5 (8.35e+4) -	1.3715e+5 (2.86e+4)
f_{26}	10	5.6834e+2 (3.66e-1) -	6.0036e+2 (3.67e+1) -	5.6867e+2 (3.79e-1) -	1.0382e+3 (4.63e-13) -	5.7376e+2 (2.31e+0) -	2.1412e+1 (1.97e+1) =	5.6808e+2 (4.19e-6) -	6.1865e+0 (1.36e+0) +	2.9063e+1 (1.64e+1) =	1.8917e+1 (1.52e+1) +	2.1733e+1 (1.99e+1)
	100	1.8114e+3 (5.84e-3) -	5.8563e+3 (1.51e+2) -	2.1992e+3 (4.99e+1) -	2.3058e+3 (1.02e+2) -	2.1256e+3 (7.59e+1) -	4.0586e+1 (7.80e+0) -	1.8114e+3 (1.94e-2) -	3.8016e+2 (3.05e+1) -	6.6076e+1 (1.09e+1) -	2.5063e+2 (9.23e+1) -	5.2431e+0 (5.68e+0)
f_{27}	10	4.7964e+0 (1.53e-4) -	7.2661e+0 (1.40e+0) -	4.8083e+0 (7.28e-3) -	1.0672e+1 (0.00e+0) -	4.8438e+0 (2.30e-2) -	7.3505e-3 (2.64e-3) -	4.7963e+0 (3.87e-9) -	6.4582e-3 (1.61e-3) -	2.7205e-3 (1.48e-3) -	7.0543e-3 (2.46e-3) -	1.4268e-3 (5.24e-4)
	100	1.4500e+1 (9.46e-4) -	7.1075e+1 (5.12e+0) -	6.8037e+1 (1.23e+1) -	1.6254e+1 (1.93e+0) -	2.1196e+1 (2.74e+0) -	3.4682e-2 (2.86e-3) -	1.4499e+1 (8.61e-4) -	2.1758e+0 (4.62e-1) -	1.6693e-2 (1.88e-3) -	4.1111e-2 (8.15e-3) -	5.8386e-3 (4.54e-4)
f_{28}	10	1.5780e+2 (2.89e+1) -	2.6668e+2 (1.25e+2) -	1.5424e+2 (8.80e+0) -	3.2753e+2 (2.49e+1) -	1.7641e+2 (9.61e+0) -	2.1353e+1 (8.88e+0) -	1.3070e+2 (6.57e+0) -	3.6373e+1 (4.81e+0) -	1.3806e+1 (7.40e+0) =	1.8969e+1 (8.23e+0) -	1.0857e+1 (5.58e+0)
	100	3.2353e+3 (8.64e+2) -	8.3368e+3 (6.74e+2) -	2.1469e+3 (1.66e+2) -	1.5865e+3 (4.91e+1) -	4.3768e+3 (2.06e+2) -	7.7994e+2 (1.26e+2) +	3.8513e+3 (1.05e+2) -	9.2824e+2 (2.98e+1) =	4.7610e+2 (3.84e+1) +	8.8673e+2 (2.94e+1) +	9.3869e+2 (1.82e+2)
f_{29}	10	3.9547e+0 (2.12e+0) -	5.0189e+0 (3.26e+0) -	1.8556e+1 (3.73e+0) -	1.0926e+1 (1.50e+1) =	1.4371e+1 (2.88e+0) -	3.4242e+0 (2.07e+0) -	5.1398e+0 (1.63e+0) -	1.0781e+1 (2.10e+0) -	5.6542e+0 (2.72e+0) -	3.5248e+0 (2.48e+0) -	2.3990e+0 (2.27e+0)
	100	3.1654e+1 (4.72e+0) +	3.2569e+1 (5.26e+0) +	6.1513e+1 (2.58e+0) -	2.4565e+0 (1.17e+0) +	5.3359e+1 (1.58e+0) -	3.1145e+1 (5.81e+0) +	2.6643e+1 (2.74e+0) +	4.7467e+1 (1.95e+0) =	2.8184e+1 (6.85e+0) +	4.4127e+1 (1.32e+0) +	4.8213e+1 (2.11e+0)
f_{30}	10	2.5466e+0 (8.83e-1) -	4.7591e+0 (1.12e+0) -	2.0191e+0 (2.90e-1) -	9.9250e+0 (1.14e+0) -	3.0593e+0 (4.21e-1) -	6.4811e-1 (4.77e-1) -	1.8203e+0 (5.59e-1) -	2.5760e-1 (7.25e-2) +	4.3714e-1 (2.36e-1) =	7.6534e-1 (5.72e-1) -	3.5359e-1 (3.70e-1)
	100	7.3022e+0 (5.11e-1) -	1.6715e+1 (2.87e+0) -	1.0509e+1 (1.68e+0) -	3.8306e+0 (4.10e-1) -	1.0381e+1 (5.69e-1) -	4.5969e+0 (5.77e-1) -	8.3731e+0 (3.61e-1) -	3.2510e+0 (3.08e-1) -	2.3408e+0 (3.61e-1) =	2.9579e+0 (4.83e-1) -	2.1886e+0 (4.05e-1)
+/-/=		2/55/3	1/59/0	0/60/0	1/58/1	0/60/0	5/49/6	2/58/0	8/49/3	7/41/12	12/47/1	

some unimodal problems like f_1 and f_2, which verifies that black-box EAs are able to be better than gradient methods on unimodal problems.

While the parameters of NeuroEAs can be tuned via a training procedure, the parameters of the manually designed algorithms compared in the experiments can be tuned as well. Table VII lists the minimum objective values obtained by GA, PSO, DE, Adam, RMSProp, and a NeuroEA on f_1-f_{13} with 10 variables, where the parameters of all the algorithms are tuned on 10-variable f_1 and f_9. It can be found that the fine-tuning of manually designed algorithms brings limited performance increment, where the fine-tuned algorithms are still outperformed by the NeuroEA. This is mainly due to the inherent rigidity of manually designed algorithms, whose search dynamics remains largely static with slight adjustments permitted through a few parameters. By contrast, NeuroEAs exhibit universal search dynamics, which is exclusively governed by block parameters that allow for comprehensive tuning.

D. Experiment of Expensive Single-Objective Optimization

Table SI in the Supplementary Materials lists the minimum objective values obtained by five surrogate-assisted metaheuristics and a NeuroEA on f_1-f_{30} with 10 and 30 variables, where the NeuroEA is first trained on 10-variable f_1 and then trained on 10-variable f_9. Obviously, the NeuroEA obtains the best results on 45 out of 60 problem instances. Moreover, Fig. 10 depicts the convergence profiles on 10-variable f_1, where the NeuroEA without surrogate converges faster than the others using surrogates. It verifies that NeuroEAs also hold fast convergence speed when using a limited number of function evaluations.

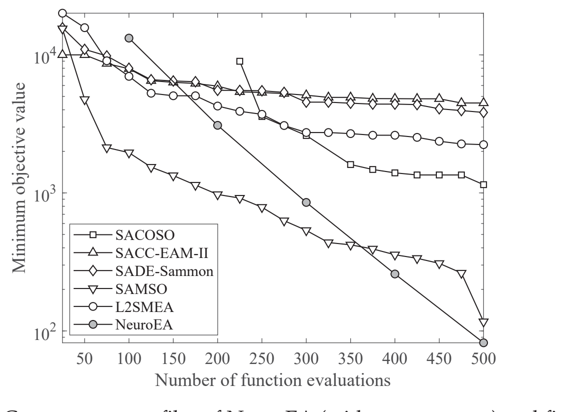


Fig. 10. Convergence profiles of NeuroEA (without surrogate) and five surrogate-assisted EAs on f_1 with 10 variables.

TABLE VI
MINIMUM OBJECTIVE VALUES OBTAINED BY A NEUROEA AND FIVE GRADIENT METHODS ON 30 SINGLE-OBJECTIVE OPTIMIZATION PROBLEMS WITH 10 & 100 VARIABLES AND 10 000 & 100 000 FUNCTION EVALUATIONS

Problem	D	Adam	BFGS	FRCG	RMSProp	SQP	NeuroEA
f_1	10	2.5425e+1 (2.92e+0) -	8.9695e-9 (3.10e-11) -	5.0141e-9 (1.62e-9) -	3.1630e+1 (8.89e+0) -	2.4747e+5 (4.67e+4) -	4.8105e-17 (9.23e-17)
	100	2.5675e+2 (7.44e+0) -	8.9685e-8 (4.14e-10) -	9.0000e-8 (6.86e-15) -	3.0038e+2 (2.03e+1) -	1.7067e+5 (6.49e+5) -	2.7319e-12 (9.48e-12)
f_2	10	1.0260e+2 (3.01e+1) -	6.8074e+8 (3.13e+9) -	9.5291e+11 (3.80e+11) -	9.8194e+1 (1.84e+2) -	4.7646e+11 (5.54e+11) -	5.1986e-12 (6.60e-12)
	100	5.9482e+19 (6.70e+19) -	1.0622e+75 (5.28e+75) -	2.5822e+120 (0.00e+0) -	6.3038e+10 (2.76e+11) -	2.5822e+120 (0.00e+0) -	2.9404e-11 (1.27e-10)
f_3	10	6.7483e+3 (5.40e+3) -	8.8716e-9 (2.86e-10) +	5.1200e+3 (4.17e+3) -	7.4331e+3 (2.51e+3) -	5.9136e+6 (4.91e+6) =	5.2965e+0 (7.27e+0)
	100	1.5294e+6 (7.14e+5) -	2.3962e+7 (1.31e+8) -	9.0873e-8 (1.70e-8) +	1.6579e+6 (6.57e+5) -	5.7745e+8 (2.20e+9) -	1.4565e+4 (2.75e+3)
f_4	10	1.5786e+0 (1.15e-1) -	5.2000e+1 (3.66e+1) -	1.6000e+2 (0.00e+0) -	1.5978e+0 (1.51e-1) -	1.0667e+2 (7.67e+1) -	1.0910e-1 (1.22e-1)
	100	1.5867e+0 (2.06e-2) +	1.5600e+2 (2.19e+1) -	1.6000e+2 (0.00e+0) -	1.5836e+0 (2.72e-2) +	1.6000e+2 (0.00e+0) -	2.9625e+1 (3.30e+0)
f_5	10	8.3358e+4 (2.62e+4) -	6.4746e+8 (1.13e+9) -	2.9049e+9 (2.24e+9) -	1.2242e+5 (6.51e+4) -	4.5806e+8 (1.40e+9) -	5.8534e+0 (1.88e+0)
	100	5.1715e+5 (4.32e+4) -	2.3127e+10 (5.78e+9) -	2.8552e+10 (2.54e+10) =	1.2656e+6 (1.98e+5) -	5.0387e+9 (1.54e+10) -	1.8466e+2 (4.83e+1)
f_6	10	6.0583e+4 (2.29e+4) -	7.6119e+4 (2.38e+4) -	2.5600e+5 (0.00e+0) -	7.6016e+4 (2.43e+4) -	2.5600e+5 (0.00e+0) -	0.0000e+0 (0.00e+0)
	100	7.0317e+5 (7.84e+4) -	6.7458e+5 (6.13e+4) -	2.5600e+6 (0.00e+0) -	6.7683e+5 (5.84e+4) -	2.5600e+6 (0.00e+0) -	0.0000e+0 (0.00e+0)
f_7	10	4.9091e+2 (1.47e+2) -	4.2650e+2 (1.63e+2) -	5.2679e+2 (1.82e+2) -	5.0136e+2 (1.66e+2) -	9.6806e+2 (3.07e-1) -	3.8250e-3 (1.76e-3)
	100	4.3577e+4 (4.43e+3) -	4.4570e+4 (6.04e+3) -	4.2633e+4 (5.91e+3) -	4.4442e+4 (3.88e+3) -	8.8841e+4 (3.29e-1) -	7.0719e-2 (2.14e-2)
f_8	10	-3.2517e+3 (7.62e+2) -	-3.3255e+3 (9.13e+2) -	-1.5403e+3 (1.85e+3) -	-3.3636e+3 (8.44e+2) -	-7.0800e+2 (1.48e+3) -	-6.4303e+3 (2.16e+1)
	100	-3.4609e+4 (2.54e+3) -	-3.4063e+4 (1.89e+3) -	-7.0021e+3 (1.63e+4) -	-3.4305e+4 (2.91e+3) -	-7.9498e+2 (0.00e+0) -	-6.3774e+4 (2.36e+2)
f_9	10	3.9784e+2 (1.18e+2) -	1.6895e+2 (7.36e+1) -	6.9859e+2 (1.40e+2) -	2.6737e+2 (6.86e+1) -	6.4810e+2 (2.27e+2) -	7.1574e-2 (2.53e-1)
	100	3.8314e+3 (5.16e+2) -	1.7717e+3 (1.83e+2) -	2.8170e+3 (2.15e+3) -	1.7142e+3 (6.97e+2) -	7.3545e+3 (2.78e-12) -	5.6728e-1 (7.75e-1)
f_{10}	10	2.1269e+1 (2.84e-1) -	1.9835e+1 (1.98e-1) -	1.9900e+1 (4.03e-1) -	2.0294e+1 (2.18e-1) -	2.0154e+1 (6.51e-1) -	1.7899e-9 (1.67e-9)
	100	2.0852e+1 (6.97e-1) -	1.9958e+1 (2.67e-1) -	2.0967e+1 (6.55e-1) -	2.0366e+1 (5.63e-2) -	2.1258e+1 (3.69e-1) -	1.1902e-14 (3.53e-15)
f_{11}	10	1.3853e+0 (1.00e+0) -	6.8970e-1 (1.97e+0) =	1.4737e+1 (1.02e+1) -	9.9700e-1 (1.23e-1) -	3.0754e+2 (7.97e+2) -	2.3133e-2 (1.72e-2)
	100	6.6604e-1 (6.27e-1) -	7.5423e-8 (1.52e-8) =	8.4989e-8 (2.28e-9) =	1.0330e+0 (1.17e-1) -	6.3940e-8 (2.02e-8) =	1.6805e-2 (3.67e-2)
f_{12}	10	1.1857e+4 (1.21e+4) -	4.4049e+9 (5.50e+9) -	6.4062e+9 (1.08e+10) -	5.3355e+3 (9.31e+3) -	8.0033e+9 (1.15e+10) -	1.0818e-1 (1.89e-2)
	100	2.6860e+5 (6.28e+4) -	1.1418e+11 (4.42e+10) -	1.1205e+11 (1.22e+11) -	7.2984e+4 (3.04e+4) -	1.1205e+11 (1.22e+11) -	1.6495e-1 (4.97e-2)
f_{13}	10	4.1802e+3 (4.49e+3) -	7.0867e+9 (7.83e+9) -	1.6887e+10 (1.60e+10) -	4.9928e+3 (6.21e+3) -	1.2656e+10 (1.58e+10) -	1.0073e-1 (2.79e-3)
	100	4.3996e+4 (1.81e+4) -	1.3666e+11 (4.15e+10) -	1.8995e+11 (1.58e+11) -	2.8151e+4 (1.44e+4) -	1.6875e+11 (1.61e+11) -	1.2666e-1 (5.20e-3)
f_{14}	10	2.4672e+1 (1.68e+0) -	6.6667e+1 (1.12e+2) -	2.5000e+2 (0.00e+0) -	4.0649e+1 (7.47e+0) -	8.3333e+0 (4.56e+1) -	2.7935e-19 (1.06e-18)
	100	2.5429e+2 (4.31e+0) -	2.3333e+3 (6.34e+2) -	2.5000e+3 (0.00e+0) -	3.5005e+2 (2.21e+1) -	8.3333e+2 (1.20e+3) -	3.2920e-14 (1.72e-13)
f_{15}	10	3.3656e+6 (5.56e+5) -	4.9199e+6 (1.39e+7) -	3.4934e+7 (1.14e+7) -	4.8076e+6 (3.09e+6) -	5.6882e-7 (5.01e-8) -	4.0719e-16 (1.56e-15)
	100	1.9602e+7 (9.58e+5) -	1.7998e+8 (2.52e+7) -	1.6022e+8 (4.37e+7) -	3.0400e+7 (6.45e+6) -	2.4874e+7 (4.61e+7) -	1.1015e-12 (1.24e-12)
f_{16}	10	2.2153e+3 (4.88e+2) -	4.8891e+2 (5.94e+2) -	4.4957e+2 (6.24e+2) -	1.1722e+3 (4.98e+2) -	2.4476e+3 (1.12e+3) -	2.3217e-1 (5.01e-1)
	100	4.7048e+3 (6.02e+2) -	3.7267e+3 (2.95e+3) -	4.4937e+3 (1.53e+3) -	7.5250e+3 (1.26e+3) -	2.7781e+4 (7.40e-12) -	4.2466e+0 (2.15e+0)
f_{17}	10	1.1041e+3 (4.52e+0) -	1.5740e+3 (6.05e+3) -	7.6807e+3 (1.67e+4) -	3.7591e+3 (3.35e+3) -	3.1084e+4 (2.05e+4) -	9.8076e-1 (7.72e-1)
	100	4.5442e+4 (1.49e+4) -	2.6756e+4 (6.84e+4) -	2.1457e+4 (8.67e+4) -	5.1470e+4 (1.16e+4) -	4.8006e+5 (2.96e-10) -	1.3853e+1 (3.28e+0)
f_{18}	10	0.0000e+0 (0.00e+0) =	6.1661e+0 (1.83e+1) -	1.0199e+2 (2.89e-14) -	0.0000e+0 (0.00e+0) =	1.0199e+2 (2.89e-14) -	0.0000e+0 (0.00e+0)
	100	0.0000e+0 (0.00e+0) =	0.0000e+0 (0.00e+0) =	2.1563e+3 (4.63e-13) -	0.0000e+0 (0.00e+0) =	2.1563e+3 (4.63e-13) -	0.0000e+0 (0.00e+0)
f_{19}	10	1.0194e+2 (1.03e+1) -	5.6240e-4 (8.74e-4) -	2.5298e-2 (6.82e-2) -	1.5777e+2 (4.43e+1) -	9.0412e+3 (2.56e+4) -	1.0912e-4 (1.36e-4)
	100	8.8909e+2 (3.11e+1) -	4.1284e+1 (3.40e+1) -	5.2568e+1 (2.56e+1) -	1.3098e+3 (1.62e+2) -	1.4236e+5 (4.03e+5) -	6.6964e-5 (9.51e-5)
f_{20}	10	8.7168e+2 (8.52e+1) -	2.8112e-10 (5.20e-11) +	5.4427e-10 (2.69e-10) +	1.2755e+3 (4.67e+2) -	9.6543e+1 (2.73e+2) =	1.2915e-1 (1.59e-1)
	100	7.0172e+3 (2.83e+2) -	1.5875e-9 (4.42e-11) +	1.6486e-9 (1.26e-10) +	9.3110e+3 (1.15e+3) -	1.6046e-9 (2.48e-11) +	6.0379e-1 (2.35e-1)
f_{21}	10	3.4603e+4 (1.28e+4) -	9.3020e-1 (1.71e+0) +	1.8905e+5 (3.49e+5) -	1.0310e+5 (4.53e+4) -	2.1606e+5 (3.64e+5) -	5.2538e+0 (2.18e+0)
	100	8.4353e+5 (1.06e+5) -	3.9866e-1 (1.22e+0) +	4.0662e+6 (8.27e+6) -	2.1312e+6 (2.48e+5) -	1.7620e+7 (7.03e+6) -	1.7332e+2 (3.79e+1)
f_{22}	10	4.5917e+4 (3.45e+4) -	3.9866e-1 (1.22e+0) +	3.9866e-1 (1.22e+0) +	6.9936e+4 (5.07e+4) -	9.4541e+5 (1.92e+6) -	1.8662e+1 (2.02e+1)
	100	7.0750e+6 (2.38e+6) -	8.8837e+6 (1.84e+7) -	7.0288e+7 (1.35e+8) =	1.2424e+7 (9.29e+6) -	3.4226e+8 (6.31e+8) -	1.2933e+2 (3.11e+1)
f_{23}	10	2.0971e+7 (7.60e+6) -	2.0002e+8 (7.81e+7) -	1.3956e+7 (1.07e+7) -	1.8145e+7 (9.12e+6) -	1.0658e+7 (3.61e+6) -	2.1727e+3 (8.53e+2)
	100	2.0895e+8 (3.06e+7) -	3.8901e+8 (3.70e+8) -	6.7504e+8 (4.98e+8) -	2.0146e+8 (4.92e+7) -	6.9204e+5 (1.88e+5) -	6.9918e+4 (2.11e+4)
f_{24}	10	2.3515e+7 (8.65e+6) -	2.2891e+8 (8.16e+7) -	2.5480e+8 (5.43e+7) -	1.5502e+7 (1.75e+7) -	1.7012e+8 (1.32e+8) -	2.9609e+1 (9.39e+0)
	100	2.5673e+8 (3.66e+7) -	1.9298e+9 (1.06e+9) -	2.3942e+9 (4.62e+8) -	9.2382e+7 (9.23e+7) -	4.9996e+8 (1.02e+9) -	2.3306e+2 (2.74e+1)
f_{25}	10	3.0221e+7 (1.92e+7) -	5.1986e+9 (1.53e+10) -	6.1689e+10 (3.03e+11) -	4.3024e+7 (2.96e+7) -	1.7845e+8 (3.37e+7) -	6.1946e+2 (1.09e+3)
	100	7.5043e+8 (4.08e+8) -	1.3426e+15 (7.30e+15) -	7.5292e+16 (2.39e+17) -	6.8028e+8 (3.26e+8) -	2.4717e+9 (1.45e-6) -	1.3715e+5 (2.86e+4)
f_{26}	10	1.2594e+3 (7.02e+1) -	3.7121e+2 (1.05e+3) -	5.6877e+0 (6.40e+0) +	1.6138e+3 (2.63e+2) -	1.0389e+3 (4.20e+2) -	2.1733e+1 (1.99e+1)
	100	4.3254e+3 (7.60e+1) -	6.2495e-2 (1.69e-2) +	1.8203e+0 (2.52e-1) =	5.4700e+3 (2.83e+2) -	4.5933e+2 (1.30e+3) -	5.2431e+0 (5.68e+0)
f_{27}	10	2.7454e+1 (3.19e+1) -	1.3520e-5 (5.97e-6) +	2.1714e-5 (4.51e-6) +	8.8942e+1 (5.85e+1) -	1.0631e-5 (3.05e-6) +	1.4268e-3 (5.24e-4)
	100	8.2822e+2 (5.26e+2) -	1.2867e-7 (9.36e-8) +	1.8129e-5 (1.31e-6) +	1.8146e+3 (7.92e+2) -	5.4497e-8 (2.52e-8) +	5.8386e-3 (4.54e-4)
f_{28}	10	2.6783e+3 (1.17e+3) -	1.0202e+3 (2.57e+2) -	9.2961e+2 (2.97e+2) -	2.4719e+3 (7.65e+2) -	1.9211e+3 (9.12e+2) -	1.0857e+1 (5.58e+0)
	100	2.6783e+3 (1.17e+3) -	1.0202e+3 (2.57e+2) -	9.2961e+2 (2.97e+2) -	2.4719e+3 (7.65e+2) -	1.9211e+3 (9.12e+2) -	9.3869e+2 (1.82e+2)
f_{29}	10	8.6864e+1 (3.29e+1) -	6.9553e+1 (2.64e+1) -	7.3753e+1 (2.44e+1) -	9.6807e+1 (2.78e+1) -	5.5000e+1 (3.61e-14) -	2.3990e+0 (2.27e+0)
	100	8.2861e+1 (1.03e+1) -	8.3395e+1 (1.33e+1) -	7.8350e+1 (9.78e+0) -	7.8112e+1 (9.16e+0) -	1.0034e+2 (1.24e-1) -	4.8213e+1 (2.11e+0)
f_{30}	10	3.4157e+3 (1.90e+3) -	1.4857e+1 (9.23e+0) -	8.7646e+0 (2.83e+0) -	2.0914e+3 (1.86e+3) -	1.1687e+1 (5.19e+0) -	3.5359e-1 (3.70e-1)
	100	1.3790e+6 (1.30e+6) -	2.0105e+1 (7.62e+0) -	1.7359e+1 (4.60e+0) -	1.3091e+6 (1.09e+6) -	1.4483e+1 (2.26e+0) -	2.1886e+0 (4.05e-1)
+/-/=		1/57/2	9/48/3	7/49/4	1/57/2	3/54/3	

TABLE VII
MINIMUM OBJECTIVE VALUES OBTAINED BY A NEUROEA, THREE FINE-TUNED METAHEURISTICS, AND TWO FINE-TUNED GRADIENT METHODS ON 13 SINGLE-OBJECTIVE OPTIMIZATION PROBLEMS WITH 10 VARIABLES AND 10 000 FUNCTION EVALUATIONS

Pro	Fine-tuned GA	Fine-tuned PSO	Fine-tuned DE	Fine-tuned Adam	Fine-tuned RMSProp	NeuroEA
f_1	1.1953e-2 ↓	7.2379e-13 ↓	4.3621e+1 ↓	1.5153e+0 ↓	2.5892e+0 ↓	4.8105e-17
f_2	9.7938e-3 ↓	3.7285e-6 ↓	6.1125e+0 ↑	3.2584e+0 ↓	4.3937e+0 ↓	5.1986e-12
f_3	1.8121e+2 ↑	3.3064e-1 ↓	4.3230e+1 ↓	2.1892e+3 ↓	3.7390e+3 ↓	5.2965e+0
f_4	3.4216e+0 ↑	5.2427e-4 ↓	9.7033e+0 ↓	1.7808e+0 ↑	5.4674e-1 ↓	1.0910e-1
f_5	4.2049e+2 ↑	6.2242e+1 ↓	1.3342e+3 ↓	2.6474e+3 ↓	5.1293e+3 ↓	5.8534e+0
f_6	0.0000e+0 -	0.0000e+0 ↓	5.0750e+1 ↓	3.8253e+4 ↓	2.6206e+4 ↓	0.0000e+0
f_7	2.7238e-2 ↑	1.7233e-2 ↓	4.6323e-2 ↑	6.2108e+1 ↓	6.4505e+1 ↓	3.8250e-3
f_8	-3.8639e+3 ↑	-2.9755e+3 ↑	-2.2005e+3 ↑	-2.6144e+3 ↑	-2.2359e+3 ↑	-6.4303e+3
f_9	8.2143e+0 ↑	1.6294e+1 ↓	5.1365e+1 ↑	2.0914e+2 ↓	2.5516e+2 ↓	7.1574e-2
f_{10}	3.8020e-2 ↓	2.0578e-1 ↓	6.3066e+0 ↓	2.1293e+1 ↑	2.1682e+1 ↑	1.7899e-9
f_{11}	8.8616e-2 ↑	1.1046e-1 ↓	1.4460e+0 ↓	7.0243e+1 ↑	5.8025e+1 ↑	2.3133e-2
f_{12}	4.2819e-1 ↑	6.0295e+0 ↓	2.0145e+1 ↓	6.0019e+2 ↓	3.0110e+2 ↓	1.0818e-1
f_{13}	1.1183e-1 ↑	1.1014e+0 ↓	5.7908e+0 ↓	6.6648e+1 ↓	6.0841e+1 ↓	1.0073e-1

"↓" indicates that the fine-tuned algorithm is better than the original algorithm, "↑" indicates that the fine-tuned algorithm is worse than the original algorithm.

E. Experiment of Combinatorial Optimization

Table SII in the Supplementary Materials lists the minimum objective values obtained by five metaheuristics and a NeuroEA on four KPs and four TSPs, where the NeuroEA is trained on 500-variable KP and 50-variable TSP. Note that the mutation block is tailed by a kopt block in the NeuroEA for solving TSPs. Besides, BSPGA and ACO can only be applied to solving KPs and TSPs, respectively. According to the statistical results, the NeuroEA obtains the best results on all the problems except for 100-variable TSP, which means that the NeuroEA optimizing real variables is more effective than the other EAs including two tailored for combinatorial optimization. This experiment verifies that black-box continuous EAs are able to outperform problem-dependent algorithms on combinatorial optimization problems.

F. Experiment of Multi-Objective Optimization

Table SIII in the Supplementary Materials lists the IGD values obtained by seven multi-objective evolutionary algorithms (MOEAs), three MOEAs with automated design methods, and a NeuroEA on 3-objective WFG1–WFG9 and IMF1–IMF10 with 10 and 30 variables, where the NeuroEA is first trained on 10-variable WFG1 and then trained on 10-variable IMF1. It can be found that the NeuroEA also exhibits the best overall performance, having achieved 27 best IGD values on 38 problem instances. This experiment verifies that NeuroEAs are effective for multi-objective optimization.

G. Experiment of Expensive Multi-Objective Optimization

Table SIV in the Supplementary Materials lists the IGD values obtained by five surrogate-assisted MOEAs and a NeuroEA on 3-objective WFG1–WFG9 and IMF1–IMF10 with 10 and 30 variables, where the NeuroEA is first trained on 10-variable WFG1 and then trained on 10-variable IMF1. Similarly, the NeuroEA without surrogate could outperform the state-of-the-art surrogate-assisted MOEAs on 32 out of 38 problem instances, which verifies the effectiveness of NeuroEAs for expensive multi-objective optimization.

V. Conclusions and Future Work

To create EAs from scratch, this paper has developed a universal framework parameterizing NeuroEAs using a set of blocks and their connections. Each block serves as a solution processing unit including population, crossover, mutation, exchange, kopt, tournament, and selection, and each connection denotes the flow of solutions between blocks. With this framework, engineers can effortlessly create innovative NeuroEAs and train them on specific problems to achieve outstanding performance. The experimental results have verified that the trained NeuroEAs are effective for single-objective optimization, expensive optimization, combinatorial optimization, and multi-objective optimization.

It should be noted that the NeuroEAs tested in the experiments are casually created, where the blocks, adjacency matrix, and hyperparameters are manually determined without exhaustive trials. Nevertheless, it can be found from the results in Table III that more tournament blocks and exchange blocks exhibit better performance. That is, many blocks arranged in parallel may offer diverse search dynamics and thus accelerate the convergence. Besides, the tested NeuroEAs may have over-customization issues on the training problems even if the search spaces are arbitrarily transformed. As a result, advanced ideas taken from deep learning could be adopted to seek more effective architectures for NeuroEAs, including but not limited to:

- *Hyperparameter optimization:* A variety of methods have been developed for optimizing the hyperparameters of neural networks [48], and they can also be adopted to optimize the hyperparameters of blocks in NeuroEAs.
- *Architecture search:* Neural architecture search has emerged as a popular topic in deep learning [49], whose principles can be leveraged to create NeuroEAs, i.e., using reinforcement learning, evolutionary computation, Bayesian optimization [87], and pipeline design methods [88] to optimize the set of blocks and their connections.
- *Block transfer:* Neural architecture search is often limited by the time-consuming training of candidate neural networks, and some methods have been suggested to reuse well-trained weights [89]. Analogously, blocks can be shared in a supergraph or inherited from evaluated NeuroEAs to accelerate architecture search.
- *Performance estimation:* Due to the inefficiency of neural architecture search, surrogate-assisted optimizers have been developed to estimate the performance of candidate neural networks without training [90]. Hence, they hold the potential for accelerating architecture search.
- *Architecture pruning:* Given the high computational complexities of deep neural networks, many pruning methods have been proposed to eliminate unnecessary layers [91]. These methods can also be adopted to prune well-trained NeuroEAs, so as to balance between effectiveness and efficiency.
- *Adversarial generation:* Generative adversarial networks have proved effective in generating realistic images [92], and the idea of training two neural networks in opposition can be adopted for training NeuroEAs. One agent trains a NeuroEA on a given problem, and the other agent varies the problem to challenge the NeuroEA [93], thereby achieving a NeuroEA effective on challenging problems.
- *Adversarial attack:* The adversarial attack of neural networks has been a key issue in artificial intelligence security, where some methods generate adversarial examples misclassified by neural networks [94]. Recent research has found that EAs are also vulnerable to such attacks [95], and adversarial attack methods can be customized to promote the development of trustworthy evolutionary algorithms.
- *Hardware acceleration:* Deep neural networks are of high computational complexity, but they are very amendable to hardware acceleration. While evolutionary algorithms can also be refactored and significantly accelerated using GPU [96], it can be considered to improve the efficiency of NeuroEAs especially those with complex architectures.

References

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems*. MIT Press, 1992.

[2] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, 1995, pp. 39–43.

[3] Y. Tian, R. Liu, X. Zhang, H. Ma, K. C. Tan, and Y. Jin, "A multipopulation evolutionary algorithm for solving large-scale multimodal multiobjective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 3, pp. 405–418, 2021.

[4] Y. Tian, L. Si, X. Zhang, R. Cheng, C. He, K. C. Tan, and Y. Jin, "Evolutionary large-scale multi-objective optimization: A survey," *ACM Computing Surveys*, vol. 54, no. 8, p. 174, 2022.

[5] Y. Tian, R. Wang, Y. Zhang, and X. Zhang, "Adaptive population sizing for multi-population based constrained multi-objective optimization," *IEEE Transactions on Cybernetics*, vol. 621, p. 129296, 2025.

[6] S. Shao, Y. Tian, Y. Zhang, and X. Zhang, "Knowledge learning-based dimensionality reduction for solving large-scale sparse multiobjective optimization problems," *IEEE Transactions on Cybernetics*, vol. 55, no. 7, pp. 3471–3484, 2025.

[7] S. Shao, Y. Tian, Y. Zhang, S. Yang, P. Zhang, C. He, X. Zhang, and Y. Jin, "Evolutionary computation for sparse multi-objective optimization: A survey," *ACM Computing Surveys*, 2025.

[8] J. Kudela, "A critical problem in benchmarking and analysis of evolutionary computation methods," *Nature Machine Intelligence*, vol. 4, pp. 1238–1245, 2022.

[9] Z. Ma, H. Guo, Y. Gong, J. Zhang, and K. C. Tan, "Toward automated algorithm design: A survey and practical guide to meta-black-box-optimization," *IEEE Transactions on Evolutionary Computation*, 2025.

[10] Y. Tian, S. Peng, T. Rodemann, X. Zhang, and Y. Jin, "Automated selection of evolutionary multi-objective optimization algorithms," in *Proceedings of the 2019 IEEE Symposium Series on Computational Intelligence*, 2019.

[11] Y. Tian, S. Peng, X. Zhang, T. Rodemann, K. C. Tan, and Y. Jin, "A recommender system for metaheuristic algorithms for continuous optimization based on deep recurrent neural networks," *IEEE Transactions on Artificial Intelligence*, vol. 1, no. 1, pp. 5–18, 2020.

[12] B. Case and P. K. Lehre, "Self-adaptation in nonelitist evolutionary algorithms on discrete problems with unknown structure," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 4, pp. 650–663, 2020.

[13] F. Zhao, F. Ji, T. Xu, N. Zhu, and Jonrinaldi, "Hierarchical parallel search with automatic parameter configuration for particle swarm optimization," *Applied Soft Computing*, vol. 151, p. 111126, 2024.

[14] J. Sun, X. Liu, T. Bäck, and Z. Xu, "Learning adaptive differential evolution algorithm from optimization experiences by policy gradient," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 4, pp. 666–680, 2021.

[15] L. Lu, H. Zheng, J. Jie, M. Zhang, and R. Dai, "Reinforcement learning-based particle swarm optimization for sewage treatment control," *Complex & Intelligent Systems*, vol. 7, no. 5, pp. 2199–2210, 2021.

[16] K. Li, A. Fialho, S. Kwong, and Q. Zhang, "Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 1, pp. 114–130, 2014.

[17] Y. Tian, X. Li, H. Ma, X. Zhang, K. C. Tan, and Y. Jin, "Deep reinforcement learning based adaptive operator selection for evolutionary multi-objective optimization," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 7, no. 4, pp. 1051–1064, 2023.

[18] Y. Tian, X. Zhang, C. He, K. C. Tan, and Y. Jin, "Principled design of translation, scale, and rotation invariant variation operators for metaheuristics," *Chinese Journal of Electronics*, vol. 32, no. 1, pp. 111–129, 2023.

[19] X. Li, K. Wu, Y. B. Li, X. Zhang, H. Wang, and J. Liu, "Pretrained optimization model for zero-shot black box optimization," in *Proceedings of the 38th Conference on Neural Information Processing Systems*, 2024.

[20] R. Rivers and D. R. Tauritz, "Evolving black-box search algorithms employing genetic programming," in *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation*, 2013, pp. 1497–1504.

[21] Q. Zhao, B. Yan, T. Hu, X. Chen, J. Yang, and S. Cheng, "AutoOpt: A general framework for automatically designing metaheuristic optimization algorithms with diverse structures," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2025.

[22] N. Stein and T. Bäck, "LLaMEA: A large language model evolutionary algorithm for automatically generating metaheuristics," *IEEE Transactions on Evolutionary Computation*, vol. 29, no. 2, pp. 331–345, 2025.

[23] L. C. T. Bezerra, M. L. lbá nez, and T. Stützle, "Automatic component-wise design of multiobjective evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 3, pp. 403–417, 2016.

[24] C. He, S. Huang, R. Cheng, K. C. Tan, and Y. Jin, "Evolutionary multiobjective optimization driven by generative adversarial networks (GANs)," *IEEE Transactions on Cybernetics*, vol. 51, no. 6, pp. 3129–3142, 2021.

[25] Z. Ma, H. Guo, J. Chen, Z. Li, G. Peng, Y. Gong, Y. Ma, and Z. Cao, "Metabox: A benchmark platform for meta-black-box optimization with reinforcement learning," in *Advances in Neural Information Processing Systems*, vol. 36, 2023, pp. 10 775–10 795.

[26] P. Kerschke, H. H. Hoos, F. Neumann, and H. Trautmann, "Automated algorithm selection: Survey and perspectives," *Evolutionary Computation*, vol. 27, no. 1, pp. 3–45, 2019.

[27] C. Huang, Y. Li, and X. Yao, "A survey of automatic parameter tuning methods for metaheuristics," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 201–216, 2020.

[28] J. R. Rice, "The algorithm selection problem," *Advances in Computers*, vol. 15, pp. 65–118, 1976.

[29] D. J. Cook and R. C. Varnell, "Maximizing the benefits of parallel search using machine learning," in *Proceedings of the 14th National Conference on Artificial Intelligence*, 1997, pp. 559–564.

[30] K. A. Smith-Miles, "Towards insightful algorithm selection for optimization using metalearning concepts," in *Proceedings of the 2008 IEEE International Joint Conference on Neural Networks*, 2008, pp. 4118–4124.

[31] E. Nudelman, K. Leyton-Brown, H. H. Hoos, A. Devkar, and Y. Shoham, "Understanding random SAT: Beyond the clauses-to-variable ratio," in *Proceedings of the 10th International Conference on Principles and Practice of Constraint Programming*, 2004, pp. 438–452.

[32] T. Mayer, T. Uhlig, and O. Rose, "Simulation-based autonomous algorithm selection for dynamic vehicle routing problems with the help of supervised learning methods," in *Proceedings of the 2018 Winter Simulation Conference*, 2018, pp. 3001–3012.

[33] P. Kerchke and H. Trautmann, "Automated algorithm selection on continuous black-box problems by combing exploratory landscape analysis and machine learning," *Evolutionary Computation*, vol. 27, no. 1, pp. 99–127, 2019.

[34] A. E. Eiben and S. K. Smit, "Parameter tuning for configuring and analyzing evolutionary algorithms," *Swarm and Evolutionary Computation*, vol. 1, pp. 19–31, 2011.

[35] H. Tong, S. Zhang, C. Huang, and X. Yao, "Algorithm portfolio for parameter tuned evolutionary algorithms," in *Proceedings of the 2019 IEEE Symposium Series on Computational Intelligence*, 2019, pp. 1849–1856.

[36] A. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 124–141, 1999.

[37] B. N. Tisdale and D. R. Tauritz, "Breaking the cycle: Exploring the advantages of novel evolutionary cycles," in *Proceedings of the 2023 IEEE Symposium Series on Computational Intelligence*, 2023.

[38] W. Yi, R. Qu, L. Jiao, and B. Niu, "Automated design of metaheuristics using reinforcement learning within a novel general search framework," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 4, pp. 1072–1084, 2023.

[39] Y. Tian, Y. Liu, S. Yang, and X. Zhang, "Deep reinforcement learning based on search space independent operators for black-box continuous optimization," *IEEE/CAA Journal of Automatica Sinica*, 2025.

[40] J. Kudela, "The evolutionary computation methods no one should use," *arXiv:2301.01984*, 2023.

[41] N. Hansen, R. Ros, N. Mauny, M. Schoenauer, and A. Auger, "Impacts of invariance in search: When CMA-ES and PSO face ill-conditioned and non-separable problems," *Applied Soft Computing*, vol. 11, no. 8, pp. 5755–5769, 2011.

[42] K. Helsgaun, "General k-opt submoves for the lin–kernighan TSP heuristic," *Mathematical Programming Computation*, vol. 1, pp. 119–163, 2009.

[43] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, "An efficient approach to non-dominated sorting for evolutionary multi-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 2, pp. 201–213, 2015.

[44] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization," in *Proceedings of the Fifth Conference on Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, 2001, pp. 95–100.

[45] M. Li, S. Yang, and X. Liu, "Shift-based density estimation for Pareto-based algorithms in many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 348–365, 2014.

[46] S. Yang, Y. Tian, C. He, X. Zhang, K. C. Tan, and Y. Jin, "A gradient guided evolutionary approach to training deep neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 9, pp. 4861–4875, 2022.

[47] B. Li, J. Li, K. Tang, and X. Yao, "Many-objective evolutionary algorithms: A survey," *ACM Computing Surveys*, vol. 48, no. 1, p. 13, 2015.

[48] L. Liao, H. Li, W. Shang, and L. Ma, "An empirical study of the impact of hyperparameter tuning and model optimization on the performance properties of deep neural networks," *ACM Transactions on Software Engineering and Methodology*, vol. 31, no. 3, p. 53, 2022.

[49] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *Journal of Machine Learning Research*, vol. 20, no. 55, pp. 1–21, 2019.

[50] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[51] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Erciyes University, Tech. Rep. tr06, 2005.

[52] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, pp. 159–195, 2001.

[53] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[54] L. Pan, W. Xu, L. Li, C. He, and R. Cheng, "Adaptive simulated binary crossover for rotated multi-objective optimization," *Swarm and Evolutionary Computation*, vol. 60, p. 100759, 2021.

[55] N. Hansen, S. Finck, R. Ros, and A. Auger, "Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions," RR-6829, INRIA, Tech. Rep., 2009.

[56] K. M. Sallam, S. M. Elsayed, R. K. Chakrabortty, and M. J. Ryan, "Improved multi-operator differential evolution algorithm for solving unconstrained problems," in *Proceedings of the 2020 IEEE Congress on Evolutionary Computation*, 2020.

[57] H. Ye, J. Wang, Z. Cao, F. Berto, C. Hua, H. Kim, J. Park, and G. Song, "ReEvo: Large language models as hyper-heuristics with reflective evolution," in *Advances in neural information processing systems*, no. 37, 2024, pp. 43 571–43 608.

[58] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[59] D. F. Shanno, "Conditioning of quasi-Newton methods for function minimization," *Mathematics of Computation*, vol. 24, pp. 647–656, 1970.

[60] R. Fletcher and C. M. Reeves, "Function minimization by conjugate gradients," *The Computer Journal*, vol. 7, no. 2, pp. 149–154, 1964.

[61] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," COURSERA: Neural networks for machine learning, Tech. Rep., 2012.

[62] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming," *Acta Numerica*, vol. 4, no. 1, pp. 1–51, 1995.

[63] C. Sun, Y. Jin, R. Cheng, J. Ding, and J. Zeng, "Surrogate-assisted cooperative swarm optimization of high-dimensional expensive problems," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 4, pp. 644–660, 2017.

[64] J. Blanchard, C. Beauthier, and T. Carletti, "A surrogate-assisted cooperative co-evolutionary algorithm using recursive differential grouping as decomposition strategy," in *Proceedings of the 2019 IEEE Congress on Evolutionary Computation*, 2019.

[65] G. Chen, K. Zhang, X. Xue, L. Zhang, J. Yao, H. Sun, L. Fan, and Y. Yang, "Surrogate-assisted evolutionary algorithm with dimensionality reduction method for water flooding production optimization," *Journal of Petroleum Science and Engineering*, vol. 185, p. 106633, 2020.

[66] F. Li, X. Cai, L. Gao, and W. Shen, "A surrogate-assisted multiswarm optimization algorithm for high-dimensional computationally expensive problems," *IEEE Transactions on Cybernetics*, vol. 51, no. 3, pp. 1390–1402, 2021.

[67] L. Si, X. Zhang, Y. Tian, S. Yang, L. Zhang, and Y. Jin, "Linear subspace surrogate modeling for large-scale expensive single/multi-objective optimization," *IEEE Transactions on Evolutionary Computation*, 2023.

[68] H. M. Salkin and C. A. D. Kluyver, "The knapsack problem: A survey," *Naval Research Logistics Quarterly*, vol. 22, no. 1, pp. 127–144, 1975.

[69] Y. Su, N. Guo, Y. Tian, and X. Zhang, "A non-revisiting genetic algorithm based on a novel binary space partition tree," *Journal of Petroleum Science and Engineering*, vol. 512, pp. 661–674, 2020.

[70] M. Dorigo and G. D. Caro, "Ant colony optimization: a new meta-heuristic," in *Proceedings of the 1999 IEEE Congress on Evolutionary Computation*, 1999.

[71] B. Gavish and S. C. Graves, "The travelling salesman problem and related problems," Operations Research Center Working Paper;OR 078-78, Tech. Rep., 1978.

[72] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[73] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 284–302, 2009.

[74] Y. Tian, X. Zheng, X. Zhang, and Y. Jin, "Efficient large-scale multiobjective optimization based on a competitive swarm optimizer," *IEEE Transactions on Cybernetics*, vol. 50, no. 8, pp. 3696–3708, 2020.

[75] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multiobjective test problems and a scalable test problem toolkit," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 5, pp. 477–506, 2006.

[76] H. Chen, R. Cheng, J. Wen, H. Li, and J. Weng, "Solving large-scale many-objective optimization problems by covariance matrix adaptation evolution strategy with scalable small subpopulations," *Information Sciences*, vol. 509, pp. 457–469, 2020.

[77] J. Jiang, J. Wu, J. Luo, X. Yang, and Z. Huang, "MOBCA: multiobjective besiege and conquer algorithm," *Biomimetics*, vol. 9, p. 316, 2024.

[78] X. Yang, J. Zou, S. Yang, J. Zheng, and Y. Liu, "A fuzzy decision variables framework for large-scale multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 3, pp. 445–459, 2023.

[79] G. Liu, Z. Pei, N. Liu, and Y. Tian, "Subspace segmentation based co-evolutionary algorithm for balancing convergence and diversity in many-objective optimization," *Swarm and Evolutionary Computation*, vol. 83, p. 101410, 2023.

[80] R. Cheng, Y. Jin, K. Narukawa, and B. Sendhoff, "A multiobjective evolutionary algorithm using Gaussian process-based inverse modeling," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 6, pp. 838–856, 2015.

[81] L. Sun and K. Li, "Adaptive operator selection based on dynamic Thompson sampling for MOEA/D," in *Proceedings of the 2020 International Conference on Parallel Problem Solving from Nature*, 2020, pp. 271–284.

[82] L. Pan, C. He, Y. Tian, H. Wang, X. Zhang, and Y. Jin, "A classification based surrogate-assisted evolutionary algorithm for expensive many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 1, pp. 74–88, 2019.

[83] H. Bian, J. Tian, J. Yu, and H. Yu, "Bayesian co-evolutionary optimization based entropy search for high-dimensional many-objective optimization," *Knowledge-Based Systems*, vol. 274, p. 110630, 2023.

[84] H. Gu, H. Wang, C. He, B. Yuan, and Y. Jin, "Large-scale multiobjective evolutionary algorithm guided by low-dimensional surrogates of scalarization functions," *Evolutionary Computation*, 2024.

[85] A. F. R. Araújo, L. R. C. Farias, and A. R. C. Gonçalves, "Self-organizing surrogate-assisted non-dominated sorting differential evolution," *Swarm and Evolutionary Computation*, vol. 91, p. 101703, 2024.

[86] C. A. C. Coello and N. C. Cortes, "Solving multiobjective optimization problems using an artificial immune system," *Genetic Programming and Evolvable Machines*, vol. 6, no. 2, pp. 163–190, 2005.

[87] Y. Tian, S. Peng, S. Yang, X. Zhang, K. C. Tan, and Y. Jin, “Action command encoding for surrogate assisted neural architecture search,” *IEEE Transactions on Cognitive and Developmental Systems*, vol. 14, no. 3, pp. 1129–1142, 2021.

[88] R. S. Olson, R. J. Urbanowicz, and P. C. Andrews, “Automating biomedical data science through tree-based pipeline optimization,” in *Proceedings of the European Conference on the Applications of Evolutionary Computation*, 2016, pp. 123–137.

[89] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, “SMASH: One-shot model architecture search through hypernetworks,” in *Proceedings of the 2018 International Conference on Learning Representations*, 2018.

[90] S. Liu, H. Zhang, and Y. Jin, “A survey on computationally efficient neural architecture search,” *Journal of Automation and Intelligence*, vol. 1, no. 1, p. 100002, 2022.

[91] Y. He and L. Xiao, “Structured pruning for deep convolutional neural networks: A survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 5, pp. 2900–2919, 2024.

[92] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in Neural Information Processing Systems 27*, 2014.

[93] K. Tang, S. Liu, P. Yang, and X. Yao, “Few-shots parallel algorithm portfolio construction via co-evolution,” *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 3, pp. 595–607, 2021.

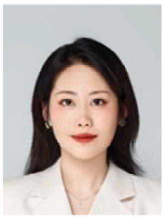
[94] Y. Tian, J. Pan, S. Yang, X. Zhang, S. He, and Y. Jin, “Imperceptible and sparse adversarial attacks via a dual-population based constrained evolutionary algorithm,” *IEEE Transactions on Artificial Intelligence*, vol. 4, no. 2, pp. 268–281, 2022.

[95] L. Zhang, R. Wang, Y. Tian, and X. Zhang, “Attacking evolutionary algorithms via SparseEA,” in *Proceedings of the Fifteenth International Conference on Swarm Intelligence*, 2024.

[96] Y. Tian, R. Cheng, X. Zhang, and Y. Jin, “Techniques for accelerating multi-objective evolutionary algorithms in PlatEMO,” in *Proceedings of the 2020 IEEE Congress on Evolutionary Computation*, 2020.

Ye Tian (Senior Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees from Anhui University, Hefei, China, in 2012, 2015, and 2018, respectively.

He is currently a Professor with the School of Computer Science and Technology, Anhui University, Hefei, China. His current research interests include evolutionary computation and its applications. He is the recipient of the 2018, 2021, and 2024 IEEE Transactions on Evolutionary Computation Outstanding Paper Award, the 2020 IEEE Computational Intelligence Magazine Outstanding Paper Award, and the 2022 IEEE Computational Intelligence Society Outstanding PhD Dissertation Award.


Xuhong Qi received the B.Sc. degree from Luoyang Normal University, Luoyang, China, in 2019, and the MA.Eng degree from Anhui University, Hefei, China, in 2025.

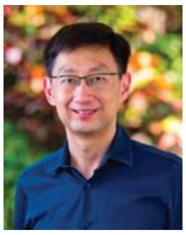
During her master’s study, her research focused on the automatic design methods of evolutionary algorithms.

Shangshang Yang received the B.Sc. and Ph.D. degrees from Anhui University, Hefei, China, in 2017 and 2022, respectively.

He is currently a postdoctoral researcher at the School of Computer Science and Technology, Anhui University, Hefei, China. His current research interests include evolutionary multi-objective optimization, automated machine learning, and intelligent education. He is the recipient of the 2023 International Conference on Data-driven Optimization of Complex Systems Best Paper Award. He received the Postdoctoral Fellowship Program (Grade B) and was awarded a Research Performance Assessment Grant (Second Grade) from the China Postdoctoral Science Foundation. He served as a reviewer/PC member for multiple top-tier conferences, including NeurIPS, ICML, ICLR, AAAI, and IJCAI.

Cheng He (Senior Member, IEEE) received the B.Eng. degree from the Wuhan University of Science and Technology, Wuhan, China, in 2012, and the Ph.D. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2018.

He is currently an Associate Professor with the School of Electrical and Electronic Engineering, Huazhong University of Science and Technology, Wuhan, China. He is an Associate Editor of Complex & Intelligent Systems and an Editorial Board Member of Evolutionary Computation. His current research interests include computational intelligence and its application in IntelliSense.


Kay Chen Tan (Fellow, IEEE) received the B.Eng. (First Class Hons.) degree in electronics and electrical engineering and the Ph.D. degree from the University of Glasgow, Glasgow, U.K., in 1994 and 1997, respectively.

Kay Chen Tan is a Chair Professor (Computational Intelligence) of the Department of Data Science and Artificial Intelligence, The Hong Kong Polytechnic University. He has coauthored 8 books and published over 300 peer-reviewed journal articles. Prof. Tan served as the Vice-President (Publications) of the IEEE Computational Intelligence Society, USA, from 2021 to 2024. He was the Editor-in-Chief of IEEE Transactions on Evolutionary Computation from 2015-2020 and IEEE Computational Intelligence Magazine from 2010-2013. Prof. Tan is an IEEE Fellow and an Honorary Professor at the University of Nottingham in the UK. He is also the Chief Co-Editor of the Springer Book Series on Machine Learning: Foundations, Methodologies, and Applications.

Yaochu Jin (Fellow, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in automatic control from Zhejiang University, Hangzhou, China, in 1988, 1991, and 1996, respectively, and the Dr.-Ing. degree in neuroinformatics from Ruhr-University Bochum, Bochum, Germany, in 2001.

He is presently a Chair Professor for AI with the School of Engineering, Westlake University, Hangzhou, China, Head of the Department of Artificial Intelligence and Director of the Trustworthy and General AI Laboratory. He was an Alexander von Humboldt Professor for AI with the Faculty of Technology, Bielefeld University, Bielefeld, Germany, and Surrey Distinguished Chair of Computational Intelligence with the Department of Computer Science, University of Surrey, Guildford, U.K. He was a Finland Distinguished Professor in Finland and a Changjiang Distinguished Visiting Professor in China. His main research interests include data-driven evolutionary optimization, trustworthy machine learning, multiobjective evolutionary learning, and evolutionary developmental systems. Dr. Jin is the recipient of the 2018, 2021 and 2023 IEEE Transactions on Evolutionary Computation Outstanding Paper Award, and the 2015, 2017, and 2020 IEEE Computational Intelligence Magazine Outstanding Paper Award. He is the recipient of the 2025 IEEE Frank Rosenblatt Award. He was named “Highly Cited Researcher” for 2019-2025 by Clarivate Analytics. He is currently the President of the IEEE Computational Intelligence Society and Editor-in-Chief of Complex & Intelligent Systems. He is a member of Academia Europaea.

Xingyi Zhang (Fellow, IEEE) received the B.Sc. degree from Fuyang Normal College, Fuyang, China, in 2003, and the M.Sc. and Ph.D. degrees from Huazhong University of Science and Technology, Wuhan, China, in 2006 and 2009, respectively.

He is currently a Professor with the School of Computer Science and Technology, Anhui University, Hefei, China. His current research interests include unconventional models and algorithms of computation, evolutionary multi-objective optimization, and logistic scheduling. He is the recipient of the 2018, 2021, and 2024 IEEE Transactions on Evolutionary Computation Outstanding Paper Award and the 2020 IEEE Computational Intelligence Magazine Outstanding Paper Award.

Supplementary Materials of "A Universal Framework for Automatically Generating Single- and Multi-Objective Evolutionary Algorithms"

Ye Tian, *Senior Member, IEEE*, Xuhong Qi, Shangshang Yang, Cheng He, *Senior Member, IEEE*, Kay Chen Tan, *Fellow, IEEE*, Yaochu Jin, *Fellow, IEEE*, and Xingyi Zhang, *Fellow, IEEE*

SI. Pseudocode of the Execution Procedure of NeuroEAs

Algorithm S1 presents the execution procedure of NeuroEAs, where a NeuroEA receives the population size N, the set of blocks $Blocks$, and the adjacency matrix Adj. To begin with, a population is initialized with N randomly generated solutions (Line 1), and is assigned to the outputs of all population blocks (Lines 2–4). At each generation, all the blocks are first flagged as *inactivated* (Lines 7–8). Then, the first inactivated block whose predecessors are all activated blocks or population blocks (Line 11) receives solutions from its predecessors through the function $Gather$ (Line 12). Afterwards, the block is executed to generate output solutions (Line 13) and changes its flag to *activated* (Line 14). The current generation finishes until all blocks' flags are changed to *activated* (Line 9), and the algorithm returns the output of the first population block after the terminal condition is fulfilled (Lines 15–16).

Algorithm S2 presents the procedure of receiving solutions from predecessors, where the function $Gather$ receives the block b to execute, the set of predecessors Pre of the block, and the adjacency matrix Adj. For each predecessor $c \in Pre$, if b asks it to evaluate solutions and the solutions in $c.output$ have not been evaluated yet (Line 3), the solutions in $c.output$ will be evaluated first (Line 4). Then, a number of solutions are passed from $c.output$ to b from the pointer $c.pointer$ (Lines 5–7), and $c.pointer$ is increased accordingly (Line 8). Here the operator mod ensures that the pointer will be reset to 1 when it exceeds the maximum number of solutions in $c.output$. After receiving solutions from all the predecessors, the received solutions are interleaved (Line 9) and returned (Line 10).

Algorithm S1: Procedure of NeuroEA

```
Input: $N$ (population size), $Blocks$ (a set of blocks), $Adj$ (adjacency matrix)
Output: $P$ (final population)
1  $P \leftarrow$ Randomly initialize $N$ solutions;
2  foreach $b \in Blocks$ do
3      if $b$ is a population block then
4          $b.output \leftarrow P$;
5      $Pre.b \leftarrow$ Obtain the set of predecessors of $b$ from $Adj$;
6  while terminal condition is not fulfilled do
7      foreach $b \in Blocks$ do
8          $b.activated \leftarrow$ false;
9      while $\exists b \in Blocks : b.activated =$ false do
10         foreach $b \in Blocks : b.activated =$ false do
11             if $\forall c \in Pre.b : c.activated =$ true or $c$ is a population block then
12                 $input \leftarrow Gather(b, Pre.b, Adj)$;
13                 $b.output \leftarrow b.Execution(input)$;
14                 $b.activated \leftarrow$ true;
15 $P \leftarrow$ Output of the first population block in $Blocks$;
16 return $P$;
```

SII. Parameter Settings of the Compared Algorithms

The 36 compared algorithms follow their suggested parameter settings in the literature as much as possible, which have verified promising performance in the experiments. Detailed settings are given as follows.

Metaheuristics:

- *Genetic algorithm (GA)* [1]: The simulated binary crossover [2] and polynomial mutation [3] with a distribution index of 20 are used as the genetic operators for continuous optimization, the uniform crossover and bit-wise mutation are used as the genetic operators for solving KPs, and the sequence based crossover and mutation are used as the genetic operators for solving TSPs, where the crossover probability is set to 1 and the mutation probability is set to $1/D$ (D is the number of variables).
- *Particle swarm optimization (PSO)* [4]: The inertia weight is set to 0.4.
- *Artificial bee colony (ABC)* [5]: The number of trials for releasing a food source is set to 20.
- *Covariance matrix adaptation evolution strategy (CMA-ES)* [6]: The initial $\sigma = 0.1(\mathbf{u} - \mathbf{l})$ and $\mu = 0.5N$, where $\mathbf{u}$ is the upper bound of the search space, $\mathbf{l}$ is the lower bound of the search space, and N is the

Algorithm S2: $Gather(b, Pre, Adj)$

```
Input: $b$ (block to execute), $Pre$ (predecessors of $b$), $Adj$ (adjacency matrix)
Output: $input$ (input of $b$)
1  $input \leftarrow \emptyset$;
2  foreach $c \in Pre$ do
3      if $b$ needs function evaluations and $c.output$ is not evaluated then
4          $c.output \leftarrow Evaluation(c.output)$;
5      $len \leftarrow \lceil |c.output| \cdot Adj(b, c) \rceil$;
6      $index \leftarrow \mathrm{mod}(\{c.pointer - 1, ..., c.pointer + len - 2\}, |c.output|) + 1$;
7      $input \leftarrow input \cup c.output_{index}$;
8      $c.pointer \leftarrow \mathrm{mod}(c.pointer + len - 1, |c.output|) + 1$;
9  Interleave the solutions in $input$;
10 return $input$;
```

population size.

- *Differential evolution (DE)* [7]: The crossover constant $CR = 0.9$ and the mutation factor $F = 0.5$.

Metaheuristics with automated design methods:

- *GA+ARSBX* [8]: Parameter settings are the same as GA.
- *IMODE* [9]: The minimum population size is set to 4 and the ratio of archive size to population size is set to 2.6.
- *LDE* [10]: The number of bins is set to 5, the number of previous generations is set to 5, the training problems are the same as NeuroEAs.
- *AutoV* [11]: A search space independent operator receiving two parents and having ten parameter sets is adopted, where the training problems and training algorithm are the same as NeuroEAs.
- *ReEvo* [12]: It uses DeepSeek-V3.2-Exp to generate a genetic algorithm, adopting a hyper-heuristic with a population size of 20 and 100 generations.

Gradient methods:

- *Adam* [13]: α is set to 1, β_1 is set to 0.9, and β_2 is set to 0.999.
- *BFGS* [14]: β is set to 0.6 and σ is set to 0.4 in backtracking line search.
- *FRCG* [15]: β is set to 0.6 and σ is set to 0.4 in backtracking line search.
- *RMSProp* [16]: α is set to 1 and ρ is set to 0.9.
- *Sequential quadratic programming (SQP)* [17]: No algorithm-specific setting.

Surrogate-assisted metaheuristics:

- *SACOSO* [18]: The population size of FES-assisted PSO is set to 30 and the population size of RBFNN-assisted PSO is set to 200.
- *SACC-EAM-II* [19]: The number of subcomponents of separable variables is set to 50.
- *SADE-Sammon* [20]: No algorithm-specific setting.
- *SAMSO* [21]: No algorithm-specific setting.
- *L2SMEA* [22]: The number of linear subspaces is set to 8.

Combinatorial metaheuristics:

- *BSPGA* [23]: The crossover probability is set to 0.5, the mutation probability is set to $1/D$, and the tree based learning probability is set to 0.05.
- *Ant colony optimization (ACO)* [24]: No algorithm-specific setting.

Multi-objective evolutionary algorithms (MOEAs):

- *NSGA-II* [25]: Parameter settings are the same as GA.
- *MOEA/D-DE* [26]: The crossover constant $CR = 0.9$, the mutation factor $F = 0.5$, the distribution index of polynomial mutation is set to 20, the probability of polynomial mutation is set to $1/D$, the probability of choosing parents locally is set to 0.9, and the maximum number of solutions replaced by each offspring is set to 2.
- *LMOCSO* [27]: The distribution index of polynomial mutation is set to 20 and the probability of polynomial mutation is set to $1/D$.
- *S^3-CMA-ES* [28]: The sample size to divide variables is set to 50 and the group size for separative variables is set to 35.
- *MOBCA* [29]: The parameter BCB is set to 0.2, the number of soldiers for each army is set to 3, and the division number of grids is set to 10.
- *FDV* [30]: The fuzzy evolution rate is set to 0.8, the step acceleration is set to 0.4, and LMOCSO is adopted as the optimizer.
- *SSCEA* [31]: The number of selected solutions for variable clustering is set to 5 and the number of perturbations on each solution is set to 50.

MOEAs with automated design methods:

- *MOEA/D-FRRMAB* [32]: The scaling factor in bandit-based operator selection is set to 5, the size of sliding window is set to $0.5N$, and the decaying factor in calculating credit value is set to 1.
- *MOEA/D-DYTS* [33]: Parameter settings are the same as MOEA/D-DE.

Surrogate-assisted MOEAs:

- *CSEA* [34]: The number of reference solutions is set to 6 and the number of solutions evaluated by surrogate model is set to 3 000.
- *ESB-CEO* [35]: The number of function evaluations at each generation is set to 5, the probability of choosing parents locally is set to 0.9, and the maximum number of solutions replaced by each offspring is set to 2.
- *LDS-AF* [36]: Parameter settings are the same as MOEA/D-DE.
- *SSDE* [37]: The number of neurons in each dimension of the latent space is set to N, the initial learning rate is set to 0.2, and the size of neighborhood mating pools is set to N.
- *MO-L2SMEA* [22]: The number of linear subspaces is set to 8.

TABLE SI
MINIMUM OBJECTIVE VALUES OBTAINED BY A NEUROEA (WITHOUT SURROGATE) AND FIVE SURROGATE-ASSISTED METAHEURISTICS ON 30 SINGLE-OBJECTIVE OPTIMIZATION PROBLEMS WITH 10 & 30 VARIABLES AND 500 FUNCTION EVALUATIONS

Problem	D	SACOSO	SACC-EAM-II	SADE-Sammon	SAMSO	L2SMEA	NeuroEA
f_1	10	1.1471e+3 (3.78e+2) -	4.4805e+3 (7.57e+2) -	3.8263e+3 (1.33e+3) -	1.1673e+2 (1.27e+2) =	2.2362e+3 (1.31e+3) -	8.2043e+1 (2.99e+1)
	30	7.4982e+3 (1.31e+3) -	1.8000e+4 (0.00e+0) -	4.0112e+4 (6.11e+3) -	4.0585e+3 (3.22e+3) =	1.3786e+4 (2.59e+3) -	3.0620e+3 (5.34e+2)
f_2	10	1.3023e+1 (2.56e+0) -	2.4023e+1 (9.63e+0) -	1.6755e+1 (3.50e+0) -	1.8725e+1 (6.33e+0) -	1.5097e+1 (4.43e+0) -	2.9314e+0 (7.95e-1)
	30	4.2169e+3 (1.27e+4) -	2.7429e+9 (2.01e+9) -	1.9973e+7 (3.81e+7) -	1.8776e+9 (5.25e+9) -	3.2274e+2 (6.48e+2) -	2.8115e+1 (3.93e+0)
f_3	10	5.0414e+3 (1.90e+3) -	6.3662e+3 (2.09e+3) -	6.7430e+3 (1.81e+3) -	4.0639e+3 (4.20e+3) -	1.8982e+3 (5.65e+2) -	5.8148e+2 (2.37e+2)
	30	8.3689e+4 (4.46e+4) -	1.3804e+5 (5.47e+4) -	6.9558e+4 (1.04e+4) -	1.2271e+5 (5.42e+4) -	2.1174e+4 (7.68e+3) -	1.0626e+4 (2.88e+3)
f_4	10	2.1983e+1 (3.57e+0) -	5.8487e+1 (1.80e+1) -	3.9984e+1 (6.19e+0) -	4.6065e+1 (7.68e+0) -	3.0565e+1 (8.75e+0) -	8.1001e+0 (1.96e+0)
	30	3.2433e+1 (5.95e+0) -	7.9981e+1 (5.96e-2) -	7.9220e+1 (1.64e+0) -	8.6981e+1 (7.81e+0) -	4.8511e+1 (7.01e+0) -	2.5871e+1 (2.62e+0)
f_5	10	2.6838e+5 (1.19e+5) -	2.1858e+6 (1.08e+6) -	2.9544e+6 (1.52e+6) -	6.9825e+3 (6.10e+3) =	8.3050e+5 (8.59e+5) -	3.8380e+3 (2.27e+3)
	30	1.8048e+6 (3.73e+5) -	3.5011e+7 (0.00e+0) -	9.8364e+7 (1.77e+7) -	1.5317e+6 (1.42e+6) =	1.3394e+7 (6.48e+6) -	7.2479e+5 (3.09e+5)
f_6	10	1.0360e+3 (2.99e+2) -	4.3295e+3 (7.47e+2) -	4.1052e+3 (1.22e+3) -	1.7370e+2 (1.03e+2) =	2.7501e+3 (1.29e+3) -	1.5320e+2 (8.65e+1)
	30	7.0594e+3 (9.87e+2) -	1.8000e+4 (0.00e+0) -	3.8550e+4 (8.24e+3) -	8.0984e+3 (7.07e+3) =	1.1534e+4 (2.56e+3) -	3.0904e+3 (8.43e+2)
f_7	10	1.7977e-1 (1.32e-1) -	6.3020e-1 (2.47e-1) -	7.5973e-1 (3.17e-1) -	2.5250e-1 (1.25e-1) -	7.2291e-1 (2.54e-1) -	3.5907e-2 (1.50e-2)
	30	2.3225e+0 (1.49e+0) -	3.6159e+0 (2.70e-1) -	5.2136e+1 (1.83e+1) -	2.0806e+0 (5.90e-1) -	5.3214e+0 (1.58e+0) -	6.1078e-1 (2.28e-1)
f_8	10	-1.6246e+3 (1.54e+2) -	-1.6472e+3 (3.07e+2) -	-2.2993e+3 (2.87e+2) =	-1.6088e+3 (2.03e+2) -	-1.4472e+3 (1.43e+2) -	-2.3044e+3 (1.02e+2)
	30	-2.7977e+3 (4.42e+2) -	-2.7041e+3 (3.76e+2) -	-3.2334e+3 (5.21e+2) =	-2.5711e+3 (4.37e+2) -	-2.6350e+3 (2.28e+2) -	-3.4135e+3 (3.00e+2)
f_9	10	6.6768e+1 (1.20e+1) -	2.9000e+1 (0.00e+0) +	6.0749e+1 (1.31e+1) -	3.3745e+1 (1.65e+1) =	7.4603e+1 (1.03e+1) -	4.1448e+1 (7.15e+0)
	30	2.0527e+2 (3.05e+1) +	5.2242e+1 (7.49e-15) +	3.2665e+2 (2.55e+1) -	1.6004e+2 (3.95e+1) +	2.8439e+2 (1.15e+1) -	2.2482e+2 (1.07e+1)
f_{10}	10	1.2748e+1 (7.77e-1) -	1.6182e+1 (6.57e-1) -	1.1535e+1 (1.05e+0) -	1.8096e+1 (8.65e-1) -	1.5135e+1 (2.51e+0) -	5.4505e+0 (8.53e-1)
	30	1.4583e+1 (6.31e-1) -	1.8102e+1 (3.74e-15) -	1.9140e+1 (5.86e-1) -	1.7268e+1 (1.51e+0) -	1.6558e+1 (1.14e+0) -	1.1273e+1 (9.23e-1)
f_{11}	10	1.1509e+1 (3.48e+0) -	3.8306e+1 (1.05e+1) -	3.2430e+1 (1.74e+1) -	2.7190e+0 (1.02e+0) =	2.2088e+1 (1.14e+1) -	2.1035e+0 (4.75e-1)
	30	6.0490e+1 (9.58e+0) -	1.6300e+2 (3.00e-14) -	3.5898e+2 (7.17e+1) -	3.9204e+1 (3.96e+1) =	1.2198e+2 (3.39e+1) -	2.9984e+1 (8.21e+0)
f_{12}	10	9.3904e+3 (1.99e+4) -	1.2412e+6 (7.70e+5) -	1.4311e+6 (1.41e+6) -	5.4591e+1 (5.54e+1) =	5.4494e+5 (5.23e+5) -	2.4447e+1 (1.38e+1)
	30	4.7320e+5 (6.88e+5) -	8.1000e+7 (1.57e-8) -	1.7241e+8 (4.97e+7) -	3.1085e+5 (4.43e+5) -	1.0044e+7 (1.05e+7) -	3.6406e+4 (5.52e+4)
f_{13}	10	1.1840e+5 (2.37e+5) -	6.6924e+6 (3.95e+6) -	5.7873e+6 (4.93e+6) -	1.8358e+1 (3.34e+1) =	2.8778e+6 (3.37e+6) -	6.0892e+0 (1.64e+0)
	30	1.5351e+6 (5.59e+5) -	1.5188e+8 (0.00e+0) -	5.2346e+8 (1.99e+8) -	4.1224e+6 (6.04e+6) =	3.6139e+7 (2.55e+7) -	8.2424e+5 (5.32e+5)
f_{14}	10	4.5239e-1 (4.54e-1) =	1.0408e+1 (2.53e+0) -	3.2454e+0 (2.56e+0) -	9.5566e-2 (1.49e-1) +	6.6068e+0 (2.30e+0) -	2.5797e-1 (1.36e-1)
	30	2.9701e-1 (2.74e-1) +	2.5000e+1 (0.00e+0) -	9.9076e+1 (1.13e+1) -	9.0346e+0 (8.67e+0) =	3.5064e+1 (9.93e+0) -	6.1449e+0 (9.89e-1)
f_{15}	10	2.9890e+4 (1.61e+4) -	8.4951e+4 (2.96e+5) -	3.2975e+4 (1.69e+4) -	8.8943e+4 (3.70e+4) -	2.0950e+4 (2.73e+4) -	1.1709e+3 (6.27e+2)
	30	9.6998e+4 (9.33e+4) =	5.5882e+5 (3.65e+5) -	1.1375e+6 (4.32e+5) -	4.7814e+5 (8.64e+4) -	4.6018e+5 (1.61e+5) -	6.6672e+4 (1.91e+4)
f_{16}	10	7.6830e+1 (1.61e+1) -	9.8688e+1 (3.76e+1) -	1.1985e+2 (2.08e+1) -	5.0812e+1 (1.34e+1) =	1.0199e+2 (1.58e+1) -	4.3161e+1 (8.24e+0)
	30	1.3857e+2 (6.27e+1) +	2.9997e+2 (6.17e+1) -	6.7078e+2 (9.94e+1) -	3.3955e+2 (5.95e+1) -	3.8831e+2 (4.71e+1) -	2.5502e+2 (2.38e+1)
f_{17}	10	1.8483e+2 (3.44e+1) -	2.0677e+2 (1.41e+1) -	1.6059e+2 (1.67e+1) -	1.2074e+2 (4.22e+1) -	1.4857e+2 (2.38e+1) -	7.0759e+1 (1.14e+1)
	30	6.3545e+2 (9.32e+1) -	5.7694e+2 (1.85e+0) -	1.0070e+3 (8.99e+1) -	6.4188e+2 (2.37e+2) -	5.8018e+2 (6.09e+1) -	3.6495e+2 (3.33e+1)
f_{18}	10	3.2370e+1 (1.07e+1) +	5.2518e+1 (5.30e+0) =	2.7115e+0 (1.70e+0) +	1.2391e+2 (3.80e-6) -	2.2978e+1 (1.56e+1) +	5.5276e+1 (1.19e+1)
	30	2.3671e+2 (2.83e+1) +	2.9794e+2 (7.20e+0) +	1.0148e+2 (2.64e+1) +	2.8766e+2 (1.18e-7) +	3.0022e+2 (3.61e+1) =	3.1476e+2 (1.59e+1)
f_{19}	10	3.4701e+0 (2.43e+0) -	3.2531e+1 (1.65e+1) -	2.3252e+1 (3.27e+0) -	2.1065e+0 (3.07e+0) =	1.1116e+1 (4.54e+0) -	1.3575e+0 (6.21e-1)
	30	1.5411e+1 (1.08e+1) =	5.7549e+1 (0.00e+0) -	1.9497e+2 (2.95e+1) -	5.7634e+1 (2.87e+1) -	6.3498e+1 (1.77e+1) -	1.5475e+1 (4.06e+0)
f_{20}	10	1.0968e+1 (6.05e+0) -	4.4949e+1 (8.11e+0) -	5.2632e+1 (1.56e+1) -	1.2614e+1 (6.57e+0) -	2.9422e+1 (1.17e+1) -	2.7719e+0 (1.79e+0)
	30	3.6528e+1 (4.32e+1) +	3.9866e+1 (7.49e-15) =	4.5040e+2 (5.49e+1) -	1.6545e+2 (6.63e+1) -	1.4317e+2 (3.71e+1) -	3.9940e+1 (5.56e+0)
f_{21}	10	2.8806e+2 (1.59e+2) -	3.0563e+3 (9.01e+2) -	2.5588e+3 (1.08e+3) -	4.1239e+2 (2.27e+2) -	1.1527e+3 (1.12e+3) -	6.6036e+1 (2.71e+1)
	30	7.6696e+2 (2.06e+2) +	2.2525e+4 (0.00e+0) -	7.9377e+4 (2.22e+4) -	7.8856e+3 (2.85e+3) -	1.0201e+4 (5.74e+3) -	1.2178e+3 (3.85e+2)
f_{22}	10	1.0906e+2 (4.76e+1) =	2.4929e+3 (1.06e+3) -	3.7361e+3 (2.48e+3) -	2.4498e+3 (1.73e+3) -	1.1511e+3 (5.38e+2) -	7.4567e+1 (3.79e+1)
	30	3.3546e+3 (4.16e+3) =	5.9168e+3 (9.59e-13) -	9.4143e+4 (3.83e+4) -	6.2276e+4 (3.14e+4) -	1.2761e+4 (6.13e+3) -	9.1524e+2 (2.99e+2)
f_{23}	10	9.0542e+4 (5.04e+4) -	1.0952e+5 (3.24e+4) -	9.1110e+4 (4.58e+4) -	3.6270e+4 (2.38e+4) -	1.1878e+4 (1.25e+4) =	1.2520e+4 (5.26e+3)
	30	1.1789e+6 (3.35e+5) -	3.3076e+5 (8.01e+4) -	1.3146e+6 (3.00e+5) -	7.6277e+5 (3.52e+5) -	3.4294e+5 (1.18e+5) -	2.1692e+5 (7.56e+4)
f_{24}	10	2.5146e+2 (2.27e+2) -	2.4942e+2 (3.03e+2) -	2.1905e+2 (1.29e+2) -	3.4592e+2 (2.31e+2) -	1.2796e+1 (4.53e+0) +	3.3378e+1 (1.21e+1)
	30	8.2622e+2 (7.80e+2) -	2.1153e+3 (2.93e+3) -	3.0591e+2 (8.56e+1) -	2.3601e+3 (3.67e+3) -	4.9816e+1 (1.00e+1) +	1.0733e+2 (2.70e+1)
f_{25}	10	6.4119e+6 (2.25e+6) -	1.4004e+7 (3.63e+6) -	7.7889e+6 (2.63e+6) -	1.8615e+7 (1.54e+7) -	5.1302e+6 (2.60e+6) -	2.7406e+5 (1.36e+5)
	30	3.5104e+7 (1.37e+7) -	2.2703e+7 (0.00e+0) -	1.1615e+8 (2.99e+7) -	1.0073e+8 (2.94e+7) -	3.7170e+7 (1.41e+7) -	7.6239e+6 (1.94e+6)
f_{26}	10	1.9238e+2 (9.91e+1) -	5.4488e+2 (6.08e+1) -	5.2174e+2 (9.48e+1) -	6.0010e+1 (5.58e+1) +	3.7069e+2 (7.84e+1) -	1.2282e+2 (3.61e+1)
	30	5.1313e+2 (2.14e+2) =	9.3401e+2 (2.40e-13) -	1.8137e+3 (1.61e+2) -	7.6079e+2 (3.24e+2) =	1.0634e+3 (1.06e+2) -	4.5554e+2 (4.48e+1)
f_{27}	10	2.3206e+0 (1.17e+0) -	5.1865e+0 (1.15e+0) -	4.7132e+0 (1.39e+0) -	2.8410e+0 (1.49e+0) -	2.1406e+0 (1.07e+0) -	3.6389e-1 (2.24e-1)
	30	8.2708e+0 (1.97e+0) -	7.0952e+0 (0.00e+0) -	2.6389e+1 (4.80e+0) -	1.7231e+1 (4.84e+0) -	9.7684e+0 (1.31e+0) -	2.4421e+0 (4.50e-1)
f_{28}	10	8.5380e+1 (2.12e+1) -	1.1908e+2 (1.98e+1) -	1.3403e+2 (1.86e+1) -	9.4485e+1 (1.43e+1) -	1.0836e+2 (1.75e+1) -	5.2932e+1 (6.06e+0)
	30	3.0096e+2 (4.99e+1) -	3.8156e+2 (5.99e-14) -	6.8170e+2 (7.63e+1) -	4.7188e+2 (8.03e+1) -	4.0431e+2 (5.26e+1) -	2.4540e+2 (1.75e+1)
f_{29}	10	2.1239e+1 (4.02e+0) =	2.2136e+1 (3.43e+0) =	1.9333e+1 (2.73e+0) =	1.4622e+1 (7.49e+0) =	1.8025e+1 (4.46e+0) =	2.0084e+1 (2.78e+0)
	30	3.8477e+1 (5.05e+0) =	3.6369e+1 (5.83e-1) =	3.6472e+1 (4.03e+0) =	4.1602e+1 (7.65e+0) =	3.8062e+1 (4.10e+0) =	3.6239e+1 (5.59e+0)
f_{30}	10	4.3450e+0 (9.85e-1) -	4.9913e+0 (9.16e-1) -	4.6130e+0 (8.71e-1) -	3.7525e+0 (7.26e-1) -	3.9838e+0 (9.89e-1) -	1.1815e+0 (2.77e-1)
	30	6.4043e+0 (1.56e+0) -	4.4956e+0 (0.00e+0) -	9.8265e+0 (1.31e+0) -	5.8008e+0 (9.16e-1) -	5.4164e+0 (6.14e-1) -	2.4619e+0 (2.97e-1)
+/-/=		7/45/8	3/53/4	2/54/4	4/38/18	3/53/4	

It should be noted that the five gradient methods Adam, BFGS, FRCG, RMSProp, and SQP do not calculate actual gradients, but estimate gradients through finite difference [38] to mimic black-box scenarios. More specifically, for each element in the gradient $\left(\frac{\partial f(\mathbf{x})}{\partial x_1}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_D}\right)$, it is calculated by

$$\frac{\partial f(\mathbf{x})}{\partial x_i} \approx \frac{f(\mathbf{x}_{i+\epsilon}) - f(\mathbf{x})}{\epsilon}, \tag{1}$$

where $i = 1, \ldots, D$, $\mathbf{x}_{i+\epsilon} = (x_1, \ldots, x_{i-1}, x_i + \epsilon, x_{i+1}, \ldots, x_D)$, and $\epsilon = 10^{-6} \times x_j$. Therefore, the calculation of one gradient consumes D function evaluations, where D is the number of variables.

SIII. ADDITIONAL EXPERIMENTAL RESULTS

In terms of expensive single-objective optimization, Table SI lists the minimum objective values obtained by five surrogate-assisted metaheuristics and a NeuroEA on f_1-f_{30} with 10 and 30 variables. In terms of combinatorial optimization, Table SII lists the minimum objective values obtained by five metaheuristics and a NeuroEA on four KPs and four TSPs. In terms of multi-objective optimization, Table SIII lists the IGD values obtained by seven multi-objective evolutionary algorithms (MOEAs), three MOEAs with automated design methods, and a NeuroEA on 3-objective WFG1–WFG9 and IMF1–IMF10 with 10 and 30 variables. In terms of expensive multi-objective optimization, Table SIV lists the IGD values obtained by five surrogate-assisted MOEAs and a NeuroEA on 3-objective WFG1–WFG9 and IMF1–IMF10 with 10 and 30 variables.

REFERENCES

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems.* MIT Press, 1992.

TABLE SII
MINIMUM OBJECTIVE VALUES OBTAINED BY A NEUROEA AND FIVE METAHEURISTICS ON TWO COMBINATORIAL OPTIMIZATION PROBLEMS WITH D=100 TO 1 000 VARIABLES AND 50×D FUNCTION EVALUATIONS

Problem	D	GA	PSO	IMODE	BSPGA	ACO	NeuroEA
KP	100	1.2619e+3 (1.74e+1) =	1.9262e+3 (1.40e+2) -	1.8339e+3 (9.61e+1) -	1.4428e+3 (3.97e+1) -	—	1.2547e+3 (1.22e+1)
	300	3.9478e+3 (3.08e+1) -	6.6073e+3 (1.95e+2) -	5.5449e+3 (1.75e+2) -	4.4395e+3 (7.61e+1) -	—	3.8827e+3 (1.84e+1)
	500	7.2691e+3 (5.70e+1) -	1.2044e+4 (2.33e+2) -	9.8540e+3 (2.73e+2) -	8.1426e+3 (1.27e+2) -	—	7.1729e+3 (2.93e+1)
	1 000	1.3880e+4 (1.04e+2) =	2.4348e+4 (3.69e+2) -	1.8434e+4 (3.51e+2) -	1.5533e+4 (2.17e+2) -	—	1.3872e+4 (8.29e+1)
TSP	10	3.7376e+0 (1.55e-1) -	3.5185e+0 (2.28e-1) -	3.3746e+0 (1.36e-15) =	—	4.6271e+0 (2.15e-1) -	3.3746e+0(1.47e-15)
	30	4.6918e+0 (2.76e-1) -	8.7862e+0 (7.55e-1) -	4.6750e+0 (2.53e-1) -	—	4.4196e+0 (1.55e-1) -	4.4124e+0 (2.30e-1)
	50	6.3385e+0 (3.66e-1) -	1.6485e+1 (1.22e+0) -	7.8456e+0 (4.27e-0) -	—	5.9921e+0 (1.73e-1) -	5.7950e+0 (7.96e-2)
	100	1.2729e+1 (1.12e+0) -	3.5407e+1 (1.81e+0) -	1.7207e+1 (5.76e-1) -	—	8.3624e+0 (3.56e-1) +	1.1555e+1 (3.60e+0)
+/ − / =		0/6/2	0/8/0	0/7/1	0/4/0	1/3/0	

TABLE SIII
IGD VALUES OBTAINED BY A NEUROEA, SEVEN MOEAS, AND THREE AUTOMATED MOEAS ON 19 MULTI-OBJECTIVE OPTIMIZATION PROBLEMS WITH 10 & 30 VARIABLES AND 10 000 FUNCTION EVALUATIONS

Problem	M, D	NSGA-II	MOEA/D-DE	LMOCSO	S^3-CMA-ES	MOBCA	FDV	SSCEA	MOEA/D-FRRMAB	MOEA/D-DYTS	AutoV	NeuroEA
WFG1	3,10	4.5681e-1 (6.26e-2) +	1.4487e+0 (7.34e-2) −	1.3698e+0 (1.20e-1) −	1.8828e+0 (2.55e-1) −	1.4423e+0 (1.33e-1) −	1.5162e+0 (3.00e-1) −	1.0043e+0 (1.94e-1) +	1.6745e+0 (5.45e-2) −	1.6338e+0 (7.27e-2) −	1.8248e+0 (3.08e-2) −	1.2619e+0 (6.32e-2)
	3,30	1.0576e+0 (6.34e-2) +	1.5589e+0 (3.09e-2) −	1.6201e+0 (3.90e-2) −	1.8632e+0 (4.96e-2) −	1.5608e+0 (2.10e-2) −	1.8380e+0 (2.34e-1) −	1.8738e+0 (1.01e-1) −	1.6015e+0 (3.29e-2) −	1.5887e+0 (3.77e-2) −	1.7304e+0 (1.64e-2) −	1.4383e+0 (3.34e-2)
WFG2	3,10	2.3454e-1 (5.49e-2) −	3.4444e-1 (2.34e-2) −	2.1295e-1 (1.67e-2) −	1.6961e-1 (2.19e-2) −	2.3101e-1 (9.75e-3) −	2.0289e-1 (7.96e-3) −	1.7262e-1 (7.89e-3) −	3.7291e-1 (2.32e-2) −	3.7387e-1 (1.91e-2) −	2.2591e-1 (7.34e-3) −	1.6904e-1 (4.22e-3)
	3,30	2.6860e-1 (4.93e-2) −	4.1599e-1 (2.62e-2) −	3.3827e-1 (3.16e-2) −	6.8543e-1 (8.71e-2) −	6.1161e-1 (2.70e-2) −	2.5875e-1 (1.91e-2) −	5.5704e-1 (7.48e-2) −	4.1347e-1 (2.40e-2) −	4.1795e-1 (2.69e-2) −	2.5677e-1 (1.22e-2) −	2.3884e-1 (1.48e-2)
WFG3	3,10	1.1762e-1 (1.70e-2) +	2.2361e-1 (3.75e-2) −	2.3137e-1 (2.78e-2) −	1.2418e-1 (2.69e-1) +	2.9242e-1 (2.53e-2) −	2.3852e-1 (2.82e-2) −	1.3097e-1 (2.14e-2) +	2.8684e-1 (3.92e-2) −	2.5882e-1 (4.50e-2) −	1.4389e-1 (2.15e-2) ≈	1.4716e-1 (2.18e-2)
	3,30	2.3835e-1 (2.37e-2) +	4.6492e-1 (4.44e-2) −	3.9414e-1 (3.45e-2) −	7.3162e-1 (1.02e-1) −	7.6892e-1 (4.11e-2) −	3.8823e-1 (3.92e-2) −	5.8156e-1 (3.89e-2) −	4.7491e-1 (2.67e-2) −	4.3981e-1 (5.20e-2) −	2.2398e-1 (2.59e-2) +	3.1406e-1 (2.91e-2)
WFG4	3,10	2.7673e-1 (8.87e-3) −	3.9008e-1 (1.25e-2) −	2.6980e-1 (6.23e-3) −	2.4506e-1 (1.06e-1) +	3.6959e-1 (1.74e-2) −	2.7996e-1 (6.80e-3) −	2.2352e-1 (5.25e-3) +	4.0874e-1 (1.63e-2) −	4.0737e-1 (1.67e-2) −	3.0507e-1 (9.15e-3) −	2.5104e-1 (5.98e-3)
	3,30	3.0351e-1 (1.08e-2) −	4.2226e-1 (1.24e-2) −	3.0817e-1 (9.67e-3) −	5.5406e-1 (4.23e-2) −	6.1822e-1 (2.21e-2) −	2.9067e-1 (1.07e-2) −	4.2643e-1 (2.62e-2) −	4.2458e-1 (1.54e-2) −	4.1655e-1 (1.30e-2) −	3.2255e-1 (8.72e-3) −	2.8108e-1 (1.02e-2)
WFG5	3,10	2.8172e-1 (1.05e-2) −	3.3491e-1 (3.96e-3) −	2.4450e-1 (1.92e-3) −	2.4016e-1 (2.97e-2) −	2.9405e-1 (1.35e-2) −	2.5047e-1 (2.59e-3) −	2.3376e-1 (4.66e-3) ≈	3.3598e-1 (3.63e-3) −	3.3726e-1 (3.64e-3) −	3.4080e-1 (2.05e-2) −	2.3300e-1 (5.12e-3)
	3,30	3.0687e-1 (8.56e-3) −	3.3829e-1 (3.23e-3) −	2.5583e-1 (4.71e-3) −	2.6923e-1 (2.27e-2) −	4.3240e-1 (1.57e-2) −	2.5032e-1 (2.45e-3) −	6.1403e-1 (2.22e-2) −	3.3846e-1 (4.24e-3) −	3.3702e-1 (4.13e-3) −	6.0581e-1 (4.77e-2) −	2.4383e-1 (7.72e-3)
WFG6	3,10	3.2164e-1 (2.25e-2) −	4.2081e-1 (3.75e-2) −	2.8885e-1 (2.07e-2) −	2.4252e-1 (6.63e-2) −	3.9481e-1 (3.89e-2) −	2.6023e-1 (1.19e-2) −	2.6859e-1 (1.74e-2) −	4.7228e-1 (2.80e-2) −	4.7169e-1 (4.01e-2) −	3.0596e-1 (1.28e-2) −	2.4095e-1 (2.99e-2)
	3,30	3.4769e-1 (1.43e-2) −	3.8142e-1 (1.58e-2) −	3.5184e-1 (7.05e-2) −	4.5264e-1 (1.04e-1) −	4.9433e-1 (2.87e-2) −	3.1883e-1 (3.46e-2) −	7.4510e-1 (2.65e-2) −	3.7498e-1 (1.08e-2) −	3.9100e-1 (2.70e-2) −	3.6001e-1 (1.79e-2) −	2.5343e-1 (5.63e-3)
WFG7	3,10	2.8501e-1 (1.24e-2) −	3.7506e-1 (1.19e-2) −	2.7892e-1 (1.14e-2) −	8.9458e-1 (8.12e-1) ≈	3.5687e-1 (2.11e-2) −	2.9190e-1 (1.64e-2) −	3.3494e-1 (2.79e-2) −	4.0719e-1 (2.31e-2) −	3.9059e-1 (2.08e-2) −	2.8455e-1 (9.93e-3) −	2.2066e-1 (3.45e-3)
	3,30	3.3129e-1 (3.76e-2) −	4.6808e-1 (1.25e-2) −	3.8806e-1 (1.78e-2) −	1.0486e+0 (4.66e-1) −	7.1971e-1 (1.89e-2) −	3.7532e-1 (1.74e-2) −	6.3486e-1 (4.23e-2) −	4.6052e-1 (2.23e-2) −	4.4751e-1 (2.30e-2) −	3.1505e-1 (1.21e-2) −	2.8908e-1 (1.94e-2)
WFG8	3,10	3.9310e-1 (8.59e-3) −	4.8289e-1 (3.98e-2) −	3.8108e-1 (1.45e-2) −	4.8787e-1 (2.16e-1) −	5.0212e-1 (2.40e-2) −	3.7371e-1 (1.38e-2) −	3.5457e-1 (1.08e-2) ≈	5.0954e-1 (4.70e-2) −	6.3069e-1 (1.35e-1) −	4.4714e-1 (1.36e-2) −	3.5826e-1 (1.10e-2)
	3,30	3.6185e-1 (9.83e-3) +	5.4693e-1 (2.43e-2) −	4.5592e-1 (1.83e-2) −	5.5410e-1 (4.47e-2) −	7.7257e-1 (1.89e-2) −	3.9779e-1 (1.16e-2) ≈	6.3770e-1 (2.49e-2) −	5.2924e-1 (1.91e-2) −	5.0925e-1 (2.41e-2) −	4.3878e-1 (1.96e-2) −	3.9813e-1 (1.95e-2)
WFG9	3,10	2.7349e-1 (1.16e-2) −	3.4037e-1 (1.67e-2) −	2.4947e-1 (7.14e-3) −	2.1948e-1 (3.69e-2) −	4.1823e-1 (6.60e-2) −	2.4534e-1 (3.15e-3) −	3.0712e-1 (4.43e-2) −	3.7640e-1 (4.96e-2) −	3.6355e-1 (5.08e-2) −	2.6807e-1 (8.20e-3) −	2.1841e-1 (6.42e-3)
	3,30	3.3331e-1 (3.16e-2) −	3.6606e-1 (1.47e-2) −	3.1850e-1 (1.89e-2) −	2.9702e-1 (2.89e-2) −	3.4553e-1 (1.06e-2) −	2.9016e-1 (1.19e-2) −	7.5568e-1 (8.17e-2) −	3.4666e-1 (5.85e-3) −	3.5558e-1 (1.39e-2) −	3.0269e-1 (1.35e-2) −	2.7067e-1 (1.40e-2)
IMF1	2,10	1.1003e-1 (7.93e-2) −	4.6202e-3 (3.00e-4) ≈	4.9024e-3 (4.25e-4) −	6.3333e-2 (2.48e-2) −	3.2280e-1 (1.04e-1) −	1.0690e-2 (1.05e-2) −	1.3761e-1 (7.35e-2) −	7.0288e-3 (1.23e-3) −	5.0527e-3 (4.41e-4) −	4.2878e-2 (2.13e-2) −	4.5017e-3 (2.52e-4)
	2,30	2.4131e-1 (4.91e-2) −	1.3662e-1 (5.44e-2) −	4.0245e-2 (2.31e-2) ≈	1.7474e+0 (5.10e-1) −	4.8424e-1 (3.21e-2) −	1.7148e-1 (1.22e-1) −	2.3986e+0 (1.98e+0) −	1.0295e-1 (6.55e-2) −	4.4548e-2 (2.60e-2) ≈	9.9213e-2 (2.89e-2) −	3.6811e-2 (1.65e-2)
IMF2	2,10	1.5165e-1 (1.15e-1) −	4.4811e-2 (1.53e-1) ≈	1.8600e-1 (2.82e-1) ≈	1.0669e-1 (4.94e-2) −	5.4114e-1 (1.77e-1) −	4.8903e-1 (2.45e-1) −	2.3920e-1 (1.64e-1) −	1.3542e-1 (2.41e-1) −	7.0586e-2 (1.83e-1) −	9.8228e-2 (7.58e-2) −	2.4744e-2 (1.10e-1)
	2,30	4.2341e-1 (9.21e-2) −	1.4560e-1 (8.41e-2) −	9.0087e-2 (1.77e-1) −	2.7269e+0 (7.83e-1) −	6.0617e-1 (5.31e-3) −	5.7069e-1 (1.48e-1) −	3.3191e+0 (1.69e+0) −	2.5430e-1 (1.79e-1) −	7.7211e-2 (5.52e-2) −	1.8665e-1 (4.22e-2) −	5.3165e-2 (4.57e-2)
IMF3	2,10	7.0431e-3 (2.56e-3) −	1.0808e-2 (1.70e-2) −	3.6937e-3 (4.48e-4) +	1.1704e-1 (1.55e-1) −	9.7130e-3 (1.67e-3) −	4.1909e-3 (2.30e-4) −	3.8936e-2 (3.96e-2) −	5.5536e-2 (6.37e-2) −	1.0401e-1 (7.42e-2) −	8.2653e-2 (2.92e-2) −	3.9699e-3 (7.28e-4)
	2,30	7.1967e-2 (2.27e-2) +	3.7021e-1 (2.11e-2) +	1.1327e-1 (4.96e-2) +	2.7968e+0 (8.41e-1) −	1.7849e-1 (1.46e-1) +	1.6491e-2 (1.78e-2) +	3.3647e+0 (1.80e+0) −	3.6249e-1 (3.00e-2) +	3.4076e-1 (1.72e-2) +	2.7466e-1 (3.41e-2) +	4.6839e-1 (1.82e-1)
IMF4	3,10	1.4093e-1 (4.77e-2) −	7.7379e-2 (1.20e-3) −	1.1725e-1 (2.25e-1) −	1.3914e-1 (7.26e-2) −	1.8981e-1 (7.83e-2) −	6.4915e-2 (3.93e-3) ≈	1.7306e-1 (5.03e-2) −	1.1773e-1 (8.09e-2) −	9.1864e-2 (2.04e-2) −	1.0227e-1 (5.63e-3) −	6.4598e-2 (2.28e-3)
	3,30	4.4769e-1 (3.90e-2) −	8.1521e-1 (3.26e-1) −	1.4943e+0 (1.05e+0) −	1.7941e+0 (1.12e+0) −	5.5218e-1 (9.69e-2) −	1.4879e+0 (1.64e+0) −	1.1487e+1 (6.54e+0) −	6.2354e-1 (3.61e-1) −	7.3689e-1 (4.57e-1) −	5.0953e-1 (6.27e-2) −	2.5567e-1 (5.88e-2)
IMF5	2,10	1.9046e-2 (6.27e-3) −	4.5737e-3 (1.67e-4) +	5.2318e-3 (2.42e-4) +	6.1283e-2 (2.55e-2) −	2.9141e-2 (3.02e-3) −	7.8824e-3 (6.20e-4) −	3.6122e-2 (1.03e-2) −	1.6515e-2 (2.90e-3) −	1.4068e-2 (2.69e-3) −	4.0595e-2 (5.97e-3) −	6.7717e-3 (6.33e-4)
	2,30	7.1312e-2 (1.45e-2) −	5.5919e-2 (2.12e-2) −	9.7267e-2 (2.20e-2) −	1.2727e-1 (3.59e-2) −	1.2884e-1 (1.65e-2) −	1.1429e-2 (1.25e-3) +	1.8433e-1 (1.72e-2) −	2.8480e-2 (1.08e-2) −	1.1800e-2 (2.03e-3) +	6.8628e-2 (8.22e-3) −	2.0660e-2 (2.60e-3)
IMF6	2,10	3.8242e-2 (1.19e-2) −	2.0904e-2 (5.80e-3) −	2.4799e-2 (4.92e-3) −	9.2219e-2 (4.01e-2) −	5.8840e-2 (1.36e-2) −	3.1889e-2 (4.09e-3) −	6.3304e-2 (2.72e-2) −	3.6830e-2 (7.34e-3) −	3.4732e-2 (6.78e-3) −	7.6344e-2 (1.12e-2) −	1.5846e-2 (3.21e-3)
	2,30	1.3574e-1 (1.84e-2) −	7.6239e-2 (1.84e-2) −	1.3429e-1 (3.99e-2) −	2.2565e-1 (7.49e-2) −	1.9902e-1 (2.41e-2) −	4.3550e-2 (1.29e-2) ≈	3.1246e-1 (3.68e-2) −	7.6463e-2 (8.07e-3) −	6.5927e-2 (1.12e-2) −	1.2729e-1 (1.48e-2) −	4.2341e-2 (4.77e-3)
IMF7	2,10	2.9958e-2 (1.70e-2) −	1.4682e-2 (7.09e-3) −	8.4528e-3 (3.69e-3) ≈	6.2212e-2 (8.37e-2) −	1.3677e-2 (4.60e-3) −	2.0466e-2 (4.15e-3) −	6.4596e-2 (2.03e-2) −	6.9917e-2 (3.83e-2) −	7.1647e-2 (4.87e-2) −	2.0410e-1 (2.00e-2) −	7.3428e-3 (1.09e-3)
	2,30	1.1273e-1 (1.18e-2) −	2.5386e-1 (2.04e-2) −	2.2458e-1 (1.55e-2) −	2.3213e-1 (6.25e-2) −	7.3368e-2 (4.70e-2) −	7.4588e-2 (5.66e-2) −	3.1500e-1 (2.60e-2) −	2.5006e-1 (5.17e-2) −	2.1703e-2 (1.62e-2) ≈	2.0509e-1 (1.32e-2) −	1.5494e-2 (3.76e-3)
IMF8	3,10	9.9245e-2 (5.58e-3) −	1.2896e-1 (1.21e-2) −	9.0346e-2 (8.48e-3) −	1.3841e-1 (4.98e-2) −	1.1500e-1 (1.10e-2) −	1.0224e-1 (8.21e-3) −	1.0793e-1 (1.59e-2) −	1.4001e-1 (4.28e-2) −	1.4300e-1 (5.97e-2) −	1.4846e-1 (9.83e-3) −	7.0323e-2 (2.82e-3)
	3,30	4.2097e-1 (9.52e-2) −	4.9366e-1 (1.60e-1) −	6.6536e-1 (7.46e-2) −	5.9923e-1 (1.36e-1) −	3.6612e-1 (2.85e-2) −	5.8916e-1 (2.52e-1) −	9.7694e-1 (8.52e-2) −	3.4232e-1 (9.24e-2) −	4.7597e-1 (2.39e-1) −	4.5168e-1 (5.73e-2) −	2.8670e-1 (8.81e-2)
IMF9	2,10	8.7588e-3 (2.31e-3) −	5.8749e-3 (1.61e-3) ≈	6.3155e-3 (1.46e-3) ≈	1.9272e-1 (1.61e-1) −	1.3990e-2 (9.83e-4) −	6.0841e-3 (2.24e-4) −	1.8192e-2 (6.31e-3) −	3.0019e-2 (6.19e-2) −	2.4683e-2 (6.31e-2) −	1.6857e-2 (1.75e-3) −	5.5470e-3 (4.17e-4)
	2,30	5.6836e-2 (1.16e-2) −	1.7236e-1 (4.07e-2) −	1.7904e-1 (5.79e-2) −	2.4271e-1 (7.03e-2) −	6.0906e-2 (1.35e-2) −	1.6979e-1 (5.54e-2) −	3.5575e-1 (3.12e-2) −	1.4647e-1 (6.49e-2) −	1.0605e-1 (4.90e-2) −	7.0606e-2 (1.86e-2) −	4.7006e-2 (1.89e-2)
IMF10	2,10	5.0370e+0 (2.98e+0) −	4.6209e+0 (3.35e+0) −	1.0816e+1 (6.46e+0) −	1.2760e+1 (5.23e+0) −	8.8120e+0 (6.21e+0) −	8.5669e+0 (4.40e+0) −	8.2175e+0 (4.54e+0) −	6.2614e+0 (5.13e+0) −	4.7423e+0 (3.49e+0) ≈	8.0464e+0 (3.86e+0) −	3.4606e+0 (4.02e+0)
	2,30	3.6558e+1 (1.10e+1) ≈	5.1028e+1 (1.10e+1) −	4.3762e+1 (1.53e+1) −	1.1002e+2 (2.32e+1) −	5.7808e+1 (1.77e+1) −	4.4512e+1 (1.32e+1) −	1.6417e+2 (3.01e+1) −	4.8010e+1 (1.23e+1) −	4.3001e+1 (1.79e+1) −	6.2564e+1 (1.26e+1) −	3.3032e+1 (1.34e+1)
+/ − / ≈		6/31/1	2/33/3	3/31/4	2/35/1	1/37/0	2/33/3	3/33/2	1/37/0	2/33/3	2/35/1	

[2] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Systems*, vol. 9, no. 4, pp. 115–148, 1995.

[3] K. Deb and M. Goyal, "A combined genetic adaptive search (GeneAS) for engineering design," *Computer Science and Informatics*, vol. 26, no. 4, pp. 30–45, 1996.

[4] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, 1995, pp. 39–43.

[5] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Erciyes University, Tech. Rep. tr06, 2005.

[6] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, pp. 159–195, 2001.

[7] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[8] L. Pan, W. Xu, L. Li, C. He, and R. Cheng, "Adaptive simulated binary crossover for rotated multi-objective optimization," *Swarm and Evolutionary Computation*, vol. 60, p. 100759, 2021.

[9] K. M. Sallam, S. M. Elsayed, R. K. Chakrabortty, and M. J. Ryan, "Improved multi-operator differential evolution algorithm for solving unconstrained problems," in *Proceedings of the 2020 IEEE Congress on Evolutionary Computation*, 2020.

[10] J. Sun, X. Liu, T. Bäck, and Z. Xu, "Learning adaptive differential evolution algorithm from optimization experiences by policy gradient," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 4, pp. 666–680, 2021.

[11] Y. Tian, X. Zhang, C. He, K. C. Tan, and Y. Jin, "Principled design of translation, scale, and rotation invariant variation operators for metaheuristics," *Chinese Journal of Electronics*, vol. 32, no. 1, pp. 111–129, 2023.

[12] H. Ye, J. Wang, Z. Cao, F. Berto, C. Hua, H. Kim, J. Park, and G. Song, "ReEvo: Large language models as hyper-heuristics with reflective evolution," in *Advances in neural information processing systems*, no. 37, 2024, pp. 43 571–43 608.

[13] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[14] D. F. Shanno, "Conditioning of quasi-Newton methods for function minimization," *Mathematics of Computation*, vol. 24, pp. 647–656, 1970.

[15] R. Fletcher and C. M. Reeves, "Function minimization by conjugate gradients," *The Computer Journal*, vol. 7, no. 2, pp. 149–154, 1964.

[16] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," COURSERA: Neural networks for machine learning, Tech. Rep., 2012.

[17] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming," *Acta Numerica*, vol. 4, no. 1, pp. 1–51, 1995.

[18] C. Sun, Y. Jin, R. Cheng, J. Ding, and J. Zeng, "Surrogate-assisted cooperative swarm optimization of high-dimensional expensive problems," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 4, pp. 644–660, 2017.

[19] J. Blanchard, C. Beauthier, and T. Carletti, "A surrogate-assisted cooperative co-evolutionary algorithm using recursive differential grouping as decomposition strategy," in *Proceedings of the 2019*

TABLE SIV
IGD VALUES OBTAINED BY A NEUROEA (WITHOUT SURROGATE) AND FIVE SURROGATE-ASSISTED MOEAS ON 19 MULTI-OBJECTIVE OPTIMIZATION PROBLEMS WITH 10 & 30 VARIABLES AND 500 FUNCTION EVALUATIONS

Problem	M, D	CSEA	ESB-CEO	LDS-AF	SSDE	MO-L2SMEA	NeuroEA
WFG1	3,10	1.5776e+0 (6.59e-2) +	2.0954e+0 (7.42e-2) −	2.1568e+0 (5.34e-2) −	2.2411e+0 (5.17e-2) −	1.9114e+0 (5.93e-2) −	1.7817e+0 (7.67e-2)
	3,30	1.5702e+0 (5.15e-2) +	2.1243e+0 (7.53e-2) −	2.1822e+0 (4.14e-2) −	2.2478e+0 (4.77e-2) −	2.1110e+0 (7.18e-2) −	1.7119e+0 (7.92e-2)
WFG2	3,10	5.4291e-1 (4.72e-2) −	6.3722e-1 (3.15e-2) −	6.7204e-1 (5.11e-2) −	6.7645e-1 (3.86e-2) −	6.6275e-1 (2.94e-2) −	3.7973e-1 (6.11e-2)
	3,30	5.7135e-1 (4.79e-2) −	7.3781e-1 (5.41e-2) −	6.8614e-1 (6.27e-2) −	7.8884e-1 (2.21e-2) −	6.4890e-1 (3.02e-2) −	5.2965e-1 (1.81e-2)
WFG3	3,10	4.6137e-1 (5.04e-2) −	5.3333e-1 (3.14e-2) −	5.6539e-1 (4.35e-2) −	5.4311e-1 (3.25e-2) −	5.7352e-1 (2.60e-2) −	3.6772e-1 (5.87e-2)
	3,30	6.4318e-1 (3.30e-2) −	6.1328e-1 (3.26e-2) −	5.7946e-1 (4.94e-2) −	7.8813e-1 (3.79e-2) −	5.8181e-1 (1.58e-2) −	5.2580e-1 (3.25e-2)
WFG4	3,10	3.9587e-1 (3.10e-2) −	4.8779e-1 (2.60e-2) −	5.2385e-1 (2.36e-2) −	5.9511e-1 (4.37e-2) −	5.0310e-1 (2.51e-2) −	3.3605e-1 (2.63e-2)
	3,30	4.5839e-1 (2.64e-2) −	5.4108e-1 (1.80e-2) −	5.4441e-1 (2.47e-2) −	6.5864e-1 (2.17e-2) −	5.4419e-1 (3.30e-2) −	3.9394e-1 (1.72e-2)
WFG5	3,10	4.1722e-1 (3.48e-2) +	3.8400e-1 (2.13e-2) +	3.6658e-1 (2.64e-2) +	5.0682e-1 (2.15e-2) +	6.4077e-1 (1.42e-2) +	7.0085e-1 (7.49e-2)
	3,30	5.5372e-1 (3.06e-2) +	4.4745e-1 (4.05e-2) +	4.1518e-1 (3.58e-2) +	5.9842e-1 (2.08e-2) +	7.9740e-1 (1.40e-2) +	8.3197e-1 (1.81e-2)
WFG6	3,10	6.5746e-1 (5.08e-2) −	6.2929e-1 (3.86e-2) −	6.9832e-1 (6.06e-2) −	7.5154e-1 (2.84e-2) −	7.5832e-1 (2.59e-2) −	6.0276e-1 (3.96e-2)
	3,30	7.6486e-1 (3.57e-2) −	7.4345e-1 (4.95e-2) −	9.1980e-1 (6.29e-2) −	9.5608e-1 (1.62e-2) −	8.1858e-1 (2.03e-2) −	6.9183e-1 (3.30e-2)
WFG7	3,10	5.4349e-1 (4.17e-2) −	5.6321e-1 (2.77e-2) −	5.8963e-1 (2.23e-2) −	6.1069e-1 (2.98e-2) −	5.6252e-1 (1.57e-2) −	4.5742e-1 (1.73e-2)
	3,30	6.3966e-1 (2.99e-2) −	6.2425e-1 (1.49e-2) −	6.2120e-1 (2.74e-2) −	7.3762e-1 (2.27e-2) −	5.7302e-1 (2.49e-2) −	4.9798e-1 (2.38e-2)
WFG8	3,10	6.8921e-1 (4.42e-2) −	7.7641e-1 (1.84e-2) −	7.9010e-1 (4.24e-2) −	7.6083e-1 (2.90e-2) −	7.7045e-1 (2.87e-2) −	5.7950e-1 (3.28e-2)
	3,30	6.9386e-1 (2.96e-2) −	7.6977e-1 (2.69e-2) −	7.4638e-1 (4.05e-2) −	8.1630e-1 (2.21e-2) −	7.2110e-1 (3.25e-2) −	5.9430e-1 (2.34e-2)
WFG9	3,10	5.5429e-1 (6.50e-2) −	6.6441e-1 (3.61e-2) −	6.3746e-1 (5.26e-2) −	6.7906e-1 (4.28e-2) −	7.0951e-1 (3.38e-2) −	4.5906e-1 (3.78e-2)
	3,30	7.9398e-1 (6.63e-2) −	7.2952e-1 (5.23e-2) −	6.4662e-1 (6.37e-2) −	8.2198e-1 (5.11e-2) −	6.9166e-1 (2.25e-2) −	5.3405e-1 (4.28e-2)
IMF1	2,10	6.0627e-1 (1.61e-1) −	6.1938e-1 (1.97e-1) −	5.3297e-1 (2.44e-1) ≈	1.4400e+0 (9.83e-1) −	2.7557e+0 (9.11e-1) −	4.1860e-1 (1.08e-1)
	2,30	1.0096e+0 (3.73e-1) −	5.5113e+0 (1.64e+0) −	2.0709e+0 (8.01e-1) −	6.2109e+0 (1.68e+0) −	5.0300e+0 (1.30e+0) −	5.0742e-1 (8.18e-2)
IMF2	2,10	1.7355e+0 (7.29e-1) −	8.1139e-1 (2.88e-1) −	9.1154e-1 (5.59e-1) −	2.5171e+0 (9.38e-1) −	3.8562e+0 (1.45e+0) −	6.0949e-1 (1.13e-16)
	2,30	1.5909e+0 (4.47e-1) −	8.9828e+0 (1.72e+0) −	3.5061e+0 (1.13e+0) −	8.3780e+0 (1.97e+0) −	7.5640e+0 (1.46e+0) −	6.2251e-1 (4.77e-2)
IMF3	2,10	1.6672e+0 (8.92e-1) −	8.0462e-1 (4.89e-1) −	8.9028e-1 (8.51e-1) −	2.0214e+0 (7.40e-1) −	3.9494e+0 (1.40e+0) −	1.1419e-1 (4.84e-2)
	2,30	1.9572e+0 (6.67e-1) ≈	7.6831e+0 (1.85e+0) −	3.1880e+0 (1.22e+0) −	7.6272e+0 (1.48e+0) −	7.9897e+0 (1.33e+0) −	1.8766e+0 (7.98e-1)
IMF4	3,10	1.9940e+0 (5.62e-1) −	7.2558e-1 (1.86e-1) −	1.3385e+0 (8.93e-1) −	2.4707e+0 (1.08e+0) −	2.7909e+0 (7.39e-1) −	5.0350e-1 (9.33e-2)
	3,30	6.8386e+0 (2.10e+0) ≈	2.5225e+1 (8.01e+0) −	1.8643e+1 (7.20e+0) −	3.0265e+1 (6.45e+0) −	2.2912e+1 (4.43e+0) −	6.1048e+0 (1.76e+0)
IMF5	2,10	1.4625e-1 (1.68e-2) −	7.7104e-2 (1.03e-2) −	1.6702e-1 (1.83e-2) −	1.6207e-1 (1.08e-2) −	1.5972e-1 (1.36e-2) −	7.1870e-2 (8.84e-3)
	2,30	2.1466e-1 (1.81e-2) −	2.0512e-1 (1.51e-2) −	2.0384e-1 (1.07e-2) −	2.5161e-1 (1.49e-2) −	2.0416e-1 (8.94e-3) −	1.7102e-1 (1.42e-2)
IMF6	2,10	2.2659e-1 (2.59e-2) −	1.1588e-1 (1.21e-2) −	2.2176e-1 (3.54e-2) −	2.2087e-1 (3.52e-2) −	2.3835e-1 (1.78e-2) −	1.0822e-1 (7.43e-3)
	2,30	3.6481e-1 (2.28e-2) −	3.0081e-1 (2.36e-2) ≈	2.8851e-1 (1.64e-2) ≈	3.9963e-1 (3.46e-2) −	3.0707e-1 (2.07e-2) −	2.8801e-1 (2.70e-2)
IMF7	2,10	2.9509e-1 (2.08e-2) −	2.7719e-1 (4.13e-2) −	3.0451e-1 (1.19e-2) −	3.1280e-1 (8.73e-3) −	3.0518e-1 (1.26e-2) −	1.8199e-1 (5.27e-2)
	2,30	3.3853e-1 (2.54e-2) −	3.1648e-1 (1.10e-2) −	3.1361e-1 (8.14e-3) ≈	3.7616e-1 (2.96e-2) −	3.2571e-1 (9.88e-3) −	3.0991e-1 (8.16e-3)
IMF8	3,10	4.2507e-1 (6.37e-2) −	2.6778e-1 (3.26e-2) +	4.5020e-1 (1.26e-1) −	5.1639e-1 (6.39e-2) −	5.5426e-1 (4.29e-2) −	3.3067e-1 (4.52e-2)
	3,30	1.0980e+0 (1.08e-1) −	8.8415e-1 (9.17e-2) +	7.4921e-1 (3.88e-2) +	1.2226e+0 (1.14e-1) −	8.8779e-1 (4.48e-2) +	9.4864e-1 (6.87e-2)
IMF9	2,10	1.3499e-1 (2.35e-2) −	1.0690e-1 (1.40e-2) −	2.3682e-1 (5.90e-2) −	2.3077e-1 (3.35e-2) −	2.6037e-1 (2.68e-2) −	5.9211e-2 (7.50e-3)
	2,30	3.4269e-1 (5.50e-2) ≈	4.0998e-1 (4.73e-2) −	4.6158e-1 (3.11e-2) −	4.4040e-1 (3.23e-2) −	4.8388e-1 (2.44e-2) −	3.2291e-1 (4.67e-2)
IMF10	2,10	3.2046e+1 (7.04e+0) −	3.1430e+1 (8.76e+0) −	1.6047e+1 (4.40e+0) −	2.4701e+1 (8.76e+0) −	3.9994e+1 (6.34e+0) −	8.3605e-1 (1.10e-2)
	2,30	1.6339e+2 (2.57e+1) −	1.9612e+2 (2.26e+1) −	1.2486e+2 (2.66e+1) −	1.7046e+2 (1.88e+1) −	2.0941e+2 (1.67e+1) −	5.0045e+1 (4.99e+1)
+/ − / ≈		4/31/3	4/33/1	3/32/3	2/36/0	3/35/0	

IEEE Congress on Evolutionary Computation, 2019.

[20] G. Chen, K. Zhang, X. Xue, L. Zhang, J. Yao, H. Sun, L. Fan, and Y. Yang, "Surrogate-assisted evolutionary algorithm with dimensionality reduction method for water flooding production optimization," *Journal of Petroleum Science and Engineering*, vol. 185, p. 106633, 2020.

[21] F. Li, X. Cai, L. Gao, and W. Shen, "A surrogate-assisted multiswarm optimization algorithm for high-dimensional computationally expensive problems," *IEEE Transactions on Cybernetics*, vol. 51, no. 3, pp. 1390–1402, 2021.

[22] L. Si, X. Zhang, Y. Tian, S. Yang, L. Zhang, and Y. Jin, "Linear subspace surrogate modeling for large-scale expensive single/multi-objective optimization," *IEEE Transactions on Evolutionary Computation*, 2023.

[23] Y. Su, N. Guo, Y. Tian, and X. Zhang, "A non-revisiting genetic algorithm based on a novel binary space partition tree," *Journal of Petroleum Science and Engineering*, vol. 512, pp. 661–674, 2020.

[24] M. Dorigo and G. D. Caro, "Ant colony optimization: a new meta-heuristic," in *Proceedings of the 1999 IEEE Congress on Evolutionary Computation*, 1999.

[25] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[26] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 284–302, 2009.

[27] Y. Tian, X. Zheng, X. Zhang, and Y. Jin, "Efficient large-scale multiobjective optimization based on a competitive swarm optimizer," *IEEE Transactions on Cybernetics*, vol. 50, no. 8, pp. 3696–3708, 2020.

[28] H. Chen, R. Cheng, J. Wen, H. Li, and J. Weng, "Solving large-scale many-objective optimization problems by covariance matrix adaptation evolution strategy with scalable small subpopulations," *Information Sciences*, vol. 509, pp. 457–469, 2020.

[29] J. Jiang, J. Wu, J. Luo, X. Yang, and Z. Huang, "MOBCA: multi-objective besiege and conquer algorithm," *Biomimetics*, vol. 9, p. 316, 2024.

[30] X. Yang, J. Zou, S. Yang, J. Zheng, and Y. Liu, "A fuzzy decision variables framework for large-scale multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 3, pp. 445–459, 2023.

[31] G. Liu, Z. Pei, N. Liu, and Y. Tian, "Subspace segmentation based co-evolutionary algorithm for balancing convergence and diversity in many-objective optimization," *Swarm and Evolutionary Computation*, vol. 83, p. 101410, 2023.

[32] K. Li, A. Fialho, S. Kwong, and Q. Zhang, "Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 1, pp. 114–130, 2014.

[33] L. Sun and K. Li, "Adaptive operator selection based on dynamic Thompson sampling for MOEA/D," in *Proceedings of the 2020 International Conference on Parallel Problem Solving from Nature*, 2020, pp. 271–284.

[34] L. Pan, C. He, Y. Tian, H. Wang, X. Zhang, and Y. Jin, "A classification based surrogate-assisted evolutionary algorithm for expensive many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 1, pp. 74–88, 2019.

[35] H. Bian, J. Tian, J. Yu, and H. Yu, "Bayesian co-evolutionary optimization based entropy search for high-dimensional many-objective optimization," *Knowledge-Based Systems*, vol. 274, p. 110630, 2023.

[36] H. Gu, H. Wang, C. He, B. Yuan, and Y. Jin, "Large-scale multiobjective evolutionary algorithm guided by low-dimensional surrogates of scalarization functions," *Evolutionary Computation*, 2024.

[37] A. F. R. Araújo, L. R. C. Farias, and A. R. C. Gonçalves, "Self-organizing surrogate-assisted non-dominated sorting differential evolution," *Swarm and Evolutionary Computation*, vol. 91, p. 101703, 2024.

[38] J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*. Wadsworth Publ. Co., Belmont, USA, 1989.