biomimetics

MDPI

Article

Vegetation Evolution with Dynamic Maturity Strategy and Diverse Mutation Strategy for Solving Optimization Problems

Rui Zhong [1,*], Fei Peng [2], Enzhi Zhang [1], Jun Yu [3] and Masaharu Munetomo [4]

[1] Graduate School of Information Science and Technology, Hokkaido University, Sapporo 060-0808, Japan; enzhi.zhang.n6@elms.hokudai.ac.jp
[2] Graduate School of Science and Technology, Niigata University, Niigata 950-3198, Japan; f22c128f@mail.cc.niigata-u.ac.jp
[3] Institute of Science and Technology, Niigata University, Niigata 950-3198, Japan; yujun@ie.niigata-u.ac.jp
[4] Information Initiative Center, Hokkaido University, Sapporo 060-0808, Japan; munetomo@iic.hokudai.ac.jp
* Correspondence: rui.zhong.u5@elms.hokudai.ac.jp

Abstract: We introduce two new search strategies to further improve the performance of vegetation evolution (VEGE) for solving continuous optimization problems. Specifically, the first strategy, named the dynamic maturity strategy, allows individuals with better fitness to have a higher probability of generating more seed individuals. Here, all individuals will first become allocated to generate a fixed number of seeds, and then the remaining number of allocatable seeds will be distributed competitively according to their fitness. Since VEGE performs poorly in getting rid of local optima, we propose the diverse mutation strategy as the second search operator with several different mutation methods to increase the diversity of seed individuals. In other words, each generated seed individual will randomly choose one of the methods to mutate with a lower probability. To evaluate the performances of the two proposed strategies, we run our proposal (VEGE + two strategies), VEGE, and another seven advanced evolutionary algorithms (EAs) on the CEC2013 benchmark functions and seven popular engineering problems. Finally, we analyze the respective contributions of these two strategies to VEGE. The experimental and statistical results confirmed that our proposal can significantly accelerate convergence and improve the convergence accuracy of the conventional VEGE in most optimization problems.

Keywords: evolutionary computation; diverse mutation strategies; dynamic maturity strategy; vegetation evolution

Citation: Rui, Z.; Fei, P.; Enzhi, Z.; Jun, Y.; Masaharu, M. Vegetation Evolution with Dynamic Maturity Strategy and Diverse Mutation Strategy for Solving Optimization Problems. *Biomimetics* **2023**, *8*, 454. https://doi.org/10.3390/biomimetics8060454

Academic Editors: Heming Jia, Laith Abualigah and Xuewen Xia

Received: 19 July 2023
Revised: 20 September 2023
Accepted: 20 September 2023
Published: 25 September 2023

1. Introduction

Since evolutionary computation (EC) does not depend on the characteristics of optimization problems and has the advantages of parallelism and robustness, these algorithms have been successfully applied to various real-world applications, such as drug design [1–3], Twitter bot detection [4–6], anomaly detection [7,8], and engineering [9–11]. With their popularity in the field of optimization, numerous new EC algorithms have been proposed. For example, invasive weed optimization (IWO) [12] was inspired by the strong survival capacity of colonizing weeds to imitate the robustness, adaptation, and randomness of colonizing weeds. The remora optimization algorithm (ROA) simulates the parasitic behavior of remora to continuously optimize the current population [13], beluga whale optimization (BWO) simulates three phases of beluga whales (i.e., exploration, exploitation, and whale fall) to find the global optimum [14], and plant competition optimization (PCO) [15] assumes each feasible solution during optimization as a plant and competes with its neighbors to realize optimization.

As one of the newest EC algorithms, vegetation evolution (VEGE) repeatedly simulates the behavior of plants in different periods to balance exploration and exploitation from a fresh perspective [16]. Due to the superior performance of the conventional VEGE

compared with some classic EC algorithms, e.g., differential evolution (DE) [17], particle swarm optimization (PSO) [18], and the enhanced fireworks algorithm (EFWA) [19], it has attracted widespread attention and several improved versions have been proposed. For example, Yu et al. introduced different mutation strategies into the growth period and maturity period to increase population diversity [20] and proposed multiple different generation strategies to increase the global search ability [21]. Additionally, they also analyzed the effects of various operations and parameter settings on the performance of the conventional VEGE [22]. Although many pieces in the literature have shown the effectiveness of VEGE, there is still much room for improvement.

The main objective of this paper is to introduce two new search strategies to further improve the search efficiency of VEGE and propose an improved VEGE to better balance search efficiency and population diversity. More specifically, the first strategy, i.e., the dynamic maturity strategy, appropriately introduces competition among individuals to generate more potential seed individuals. In this strategy, we consider the concept of the proximate optimal principle (POP), which suggests that well-performing solutions have similar structures. Therefore, we dynamically allocate computational resources to plants in the seeding operator, where the better individuals can generate more seeds and vice versa. In the second strategy, i.e., the diverse mutation strategy, we observe the relatively weak capacity of VEGE for getting rid of local optima, and the ensemble of the mutation module can improve the ability to escape from trapped local areas. In the numerical experiments, the 30-D and 50-D CEC2013 benchmark functions are employed to evaluate the overall optimization performance of our proposed improved VEGE, and seven engineering problems are adopted to investigate the capacity of our proposal in real-world scenarios. Seven advanced EAs and the conventional VEGE are the competitor algorithms. In addition, we also investigate contributions of the two strategies to performance improvement and their application scenarios. Finally, we provide some open topics for free discussion.

The remainder of this paper is organized as follows. We briefly introduce the optimization framework of the conventional VEGE in Section 2 and give a detailed description of two proposed strategies in Section 3. The parameter settings of the analysis experiment are given in Section 4. We then analyze the effectiveness of the two proposed strategies as well as their strengths and weaknesses in Section 5. Finally, we conclude our work in Section 6.

2. Vegetation Evolution

Since the genetic algorithm (GA) [23,24] has sparked a wave of research in the EC community, many well-known EC algorithms have been proposed one after another. Initially, practitioners borrowed biological evolution or the intelligent behavior of animal groups to design new EC algorithms. Then, many natural phenomena as well as human culture have also become sources of inspiration and developed various novel EC algorithms. However, only a few of these derive inspiration from plants to propose new algorithms, such as the flower pollination algorithm [25] and dandelion optimizer [26]. Fortunately, there has also recently been attention to developing new algorithms inspired by the behavior of different plants. As one of the latest members of this branch, the conventional VEGE simulates the common life cycle of plants derived from observations of different plants rather than simulating the behavior of a particular plant to find the global optimum.

Similarly to most heuristic evolutionary algorithms, the conventional VEGE is also population-based and iteratively improves the accuracy of individuals (candidate solutions) to converge to the global optimum. Here, a real plant is modeled as an individual, and each individual sequentially goes through two distinct life periods: growth and maturity. Among these, the growing individuals are responsible for local search, while the mature individuals are responsible for global search. Thus, the unique contribution of the conventional VEGE is to interactively switch between two different search capabilities to balance exploitation and exploration well. A visual demonstration of the the conventional VEGE is shown in Figure 1.

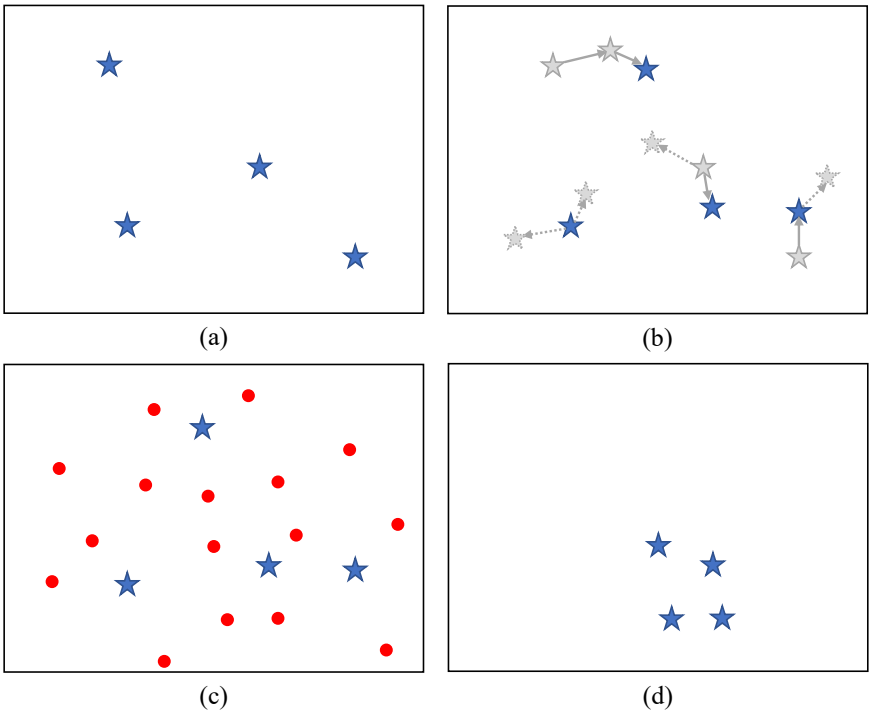

Figure 1. The optimization process of the conventional VEGE, and (**a**–**d**) demonstrate initialization, growth period, maturity period, and selection, respectively. The dashed arrows indicate the growth vector of individuals, and all red dots indicate generated seed individuals.

The general optimization process of the conventional VEGE can be summarized simply as follows. Usually, random initialization is used to generate an initial population consisting of multiple individuals. Equation (1) represents the process of random initialization.

$$P = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ \vdots \\ X_n \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ x_{31} & x_{32} & \cdots & x_{3m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix}$$
$$x_{ij} = r_1 \times (UB_j - LB_j) + LB_j \tag{1}$$

where LB_j and UB_j are the lower and upper bounds of the j^{th} dimension; n and m are the population size and the dimensionality of the optimization problem, respectively; and r_1 is a random number in (0, 1). These randomly initialized individuals first enter the growth period, and each individual generates only one offspring individual via Equation (2).

$$\vec{\boldsymbol{X}}_i^{t+1} = \vec{\boldsymbol{X}}_i^{t} + \vec{\boldsymbol{GR}} \odot \vec{\boldsymbol{GD}} \tag{2}$$

where $\vec{\boldsymbol{GR}}$ is a vector to denote the growth radius of plants in every dimension, and $\vec{\boldsymbol{GD}}$ in $(-1, 1)$ represents the growth direction of plants in every dimension. If the generated offspring individual is better than its parent individual, it will replace the parent individual, otherwise, the parent individual is directly copied to the next generation. After going through a number of growths, i.e., reaching a predefined maximum number of growths, all individuals enter the maturity period. Then, each individual will generate multiple seed individuals by the DE/cur/1-like mutation strategy [17] in Equation (3). Note that

this operation will only be performed once, and the individuals that survive to the next generation will enter the growth period again.

$$\vec{X}_{seed} = \vec{X}_i + \vec{MS} \odot (\vec{X}_{r1} - \vec{X}_{r2}) \quad (3)$$

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

3. Our Proposal: VEGE with Dynamic Maturity Strategy and Diverse Mutation Strategy

The conventional VEGE extracts and simulates some common characteristics of the plant life cycle to gradually update the population, where each individual is treated equally and allocated the same resources, such as the same number of growths and seeds. Actually, the real plant ecosystem is much more complex than observed, and differences exist between different species and even between different individuals of the same species. Here, we introduce two new search strategies, i.e., the dynamic maturity and diverse mutation strategies, into the conventional VEGE to simulate the survival mode of real plants more realistically. To better understand the procedures of our proposal, the flowchart is visualized in Figure 2.

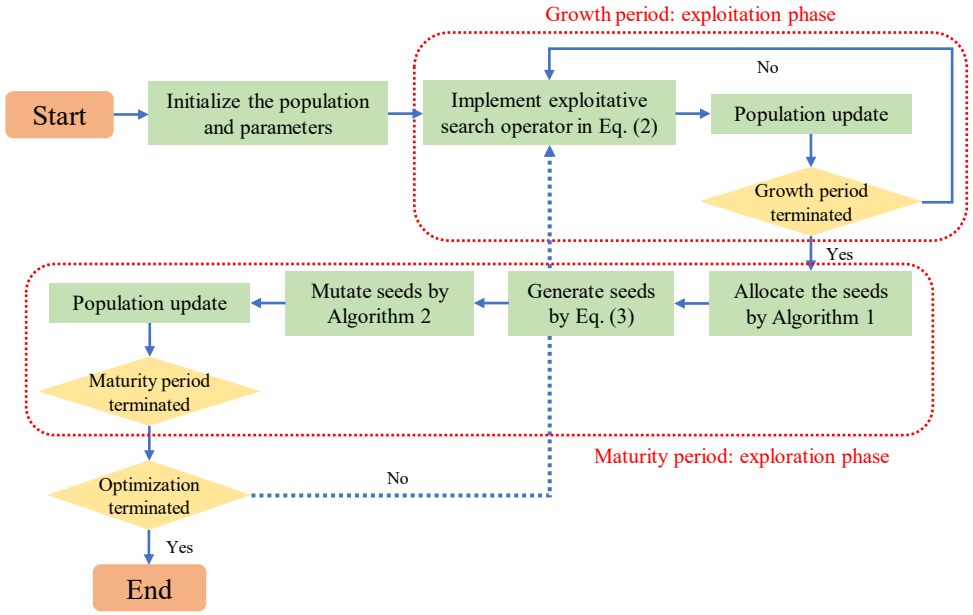

Figure 2. The flowchart of our proposal.

3.1. Dynamic Maturity Strategy

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

Suppose the total number of seed individuals that can be generated in each generation is SI. We employ the proposed dynamic maturity strategy to assign the number of seed individuals for each individual in the maturity period. First, each individual will generate the same number of k seed individuals, and then the remaining $(SI - k \times n)$ seed individuals will be allocated in a competitive manner based on the fitness of all individuals. Here, we use Equation (4) to determine the probability of each individual being selected and use a roulette wheel to allocate the remaining seed resources. Finally, individuals with better fitness have a higher probability of generating more seed individuals.

$$p_i = \text{softmax}(\frac{1}{f_i}) = \frac{\exp(\frac{1}{f_i})}{\sum_{j=1}^{n} \exp(\frac{1}{f_i})} \tag{4}$$

where p_i is the probability that the i-th individual is selected, and f_i is the fitness of the i-th individual. Algorithm 1 describes the pseudocode of this strategy.

Algorithm 1 Dynamic maturity strategy

```
Obtain the current population X and the fitness value f.
Initialize the seeding resources SR = [k, k, ..., k] for each plant.
Calculate the probability p for each plant by Equation (4).
Calculate the cumulative probability CP.
for i = 0, ..., (SI − k × n) do
    Generate a random value r in (0, 1).
    for j = 1, ..., n do
        if CP_{j−1} ≤ r < CP_j then
            SR_{j−1} ← SR_{j−1} + 1.
        end if
    end for
end for
Output the allocated seeding resources SR.
```

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

3.2. Diverse Mutation Strategy

While mutation is an important method for increasing population diversity, the conventional VEGE did not introduce any mutation strategy when it was originally designed. Later, some practitioners realized this defect and adopted different mutation methods to simulate external mutation and internal mutation for individuals in growth and maturity periods, respectively [20]. However, there are various ways of mutation in nature since the complexity of the ecosystem is far beyond our imagination. Thus, we introduce multiple different mutation methods only for seed individuals to provide more diversified potential individuals. In other words, each newly generated seed individual will randomly select one of the following three methods with a uniform probability. Note that these mutations are performed on the seed individuals before their evaluation.

- The genes of a seed individual add a Gaussian noise with a 10% probability. Here, the Gaussian noise is generated by multiplying a standard Gaussian noise by 0.05× (search upper bound—search lower bound).
- The seed individual mutates with its parent by a crossover operator with equal probability.
- The genes of a seed individual are replaced with a random value in the global space with a probability of 1%.

Algorithm 2 describes the procedure of this diverse mutation strategy.

Algorithm 2 Diverse mutation strategy

for $i = 0, \ldots, SI$ **do**
  Generate a random value r in (0, 1).
  if $r < 1/3$ **then**
    for $j = 0, \ldots, D$ **do**
      Generate a random value θ in (0, 1).
      if $\theta < 0.1$ **then**
        Generate a random value ξ, which follows the normal distribution $N(0, 1)$.
        $X_{seed,j} = X_{seed,j} + 0.05 \times \xi \times (UB_j - LB_j)$. % Mutation strategy 1
      end if
    end for
  else if $1/3 \leq r < 2/3$ **then**
    for $j = 0, \ldots, D$ **do**
      Generate a random value θ in (0, 1).
      if $\theta < 0.5$ **then**
        $X_{seed,j} = X_{seed,j}$.
      else
        $X_{seed,j} = X_{parent,j}$. % Mutation strategy 2
      end if
    end for
  else
    Generate a random value θ in (0, 1).
    if $\theta < 0.01$ **then**
      Generate a random value ς in (0, 1).
      $X_{seed,j} = LB_j + \varsigma \times (UB_j - LB_j)$. % Mutation strategy 3
    end if
  end if
end for
Output seeds after diverse mutation strategies.

So far, the two proposed strategies have been explained, and they are both improvements for the maturity period. Algorithm 3 gives the general framework of the conventional VEGE combined with the two proposed strategies.

Algorithm 3 The general framework of the conventional VEGE combined with the two proposed strategies. Steps 8 and 10 are our proposed new strategies, respectively.

1: Initialize the population randomly.
2: Evaluate the fitness of all initial individuals.
3: **if** all individuals are in the growth period **then**
4: **for** $i = 1, \ldots, n$ **do**
5: Perform the growth operation for i-th individual using the method of the conventional VEGE.
6: **end for**
7: **else**
8: Determine the number of seed individuals for each individual by Algorithm 1.
9: All individuals generate seed individuals in turn.
10: Use the proposed diverse mutation strategy in Algorithm 2 for generated seed individuals.
11: Evaluate all seed individuals after undergoing mutation.
12: Mix the current population and all seed individuals to select the next generation.
13: **end if**
14: Output the found global optimum.

4. Experimental Evaluations

Three subsections are involved: benchmark functions, competitor algorithms and the parameter setting, and experimental results.

4.1. Benchmark Functions

Since the 28 benchmark functions from the CEC2013 test suite [27] have most of the common features, we run (the conventional VEGE + two strategies), the conventional VEGE, DE, PSO, DE with self-adaptive populations (DE-SAP), phasor PSO (PPSO), social ski-driver optimization (SSDO), RIME, and snow ablation optimization (SAO) on two different dimensions of these functions to analyze the effectiveness of our proposed two strategies. Table 1 gives a detailed summary of the CEC2013 benchmark functions including the multimodal, asymmetry, non-separable variables, and the global optimum value.

Table 1. Summary of the CEC2013 suite: Uni. = unimodal function, Multi. = multimodal function, Comp. = composition function. Search space for each function is $[-100, 100]^D$.

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

Moreover, we investigate the robustness of our proposal in real-world applications. Seven popular engineering optimization problems are employed as test functions, which are listed in Table 2 and were provided by the ENOPPY library [28].

Given that the original versions of all techniques cannot solve the constrained optimization problems, we equip all EAs with the static penalty function [29], which is defined by Equation (5)

$$F(R_i) = f(R_i) + w \times \sum_{i=1}^{m} (\max(0, g_i(R_i))) \quad (5)$$

where $F(\cdot)$ is the fitness function, while $f(\cdot)$ and $g_i(\cdot)$ are the objective function and constraint function, respectively. w is a constant set to 10^7 by default in the ENOPPY library.

Table 2. Summary of seven engineering optimization problems: Dim. = dimension size.

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

4.2. Competitor Algorithms and Parameter Settings

To ensure the fairness of the comparison, we use the number of fitness evaluations to terminate the comparative experiments, and the maximum number is set to $1000\times$ *dimension* for the CEC2013 28 benchmark functions and 20,000 for engineering problems. In addition, each dimension of each function is run 30 times independently to avoid randomness. Table 3 shows the parameter settings of eight EC algorithms, and the parameter configuration of the two VEGE algorithms is kept exactly the same. In addition, all parameters of the competitor algorithms follow the recommended setting from the corresponding paper, respectively, and the compared DE, PSO, DE-SAP, PPSO, and SSDO are provided by the MEALPY library [30].

Table 3. The parameter settings of eight comparison algorithms.

EAs	Parameters	Value
DE (1995) [17]	Population size	100
	Scaling factor F	1
	Crossover rate Cr	0.9
	Mutation strategy	DE/cur-to-rand/1/bin
PSO (1995) [18]	Population size	100
	Inertia factor w	1
	Coefficients c_1 and c_2	2.05
	Max. and min. speed	2, −2
DE-SAP (2006) [31]	Population size	100
	Encoding method w	absolute encoding (Abs)
PPSO (2019) [32]	Population size	100
SSDO (2020) [33]	Population size	100
VEGE (2022) [16]	Population size	10
	Growth cycle GC	6
	Growth radius GR	2
	Growth direction GD	a random number in [−1,1]
	Total # of seed individuals	60
	Moving scaling MS	a random number in [−2,2]
RIME (2023) [34]	Population size	100
	parameter w	5
SAO (2023) [35]	Population size	100

4.3. Experimental Results

Tables 4 and 5 give the experimental and statistical results of the numerical experiments on the 30-D and 50-D CEC2013 benchmark functions. We applied the Kruskal–Wallis test and Holm's multiple comparison test to check whether there is a significant difference between these at the termination of the competitor algorithms. $+$, $\approx$, and $-$ are applied to represent that our proposal is significantly better, with no significance, and significantly worse with the compared method, and the best value is in bold. Due to the limitation of space, the optimization convergence curves of the representative functions are provided in Figures 3 and 4. Table 6 provides the detailed optimization results on seven engineering problems, and the convergence curves of the engineering problems are presented in Figure 5. Tables 7 and 8 summarize the ablation experimental results to investigate the respective contributions of our proposed two strategies to performance improvement. For the sake of simplicity, VEGE + dynamic maturity strategy is referred to as VEGE-i, VEGE + diverse mutation strategy is referred to as VEGE-ii, and VEGE + two strategies is referred to as the Proposal.

Table 4. Experimental and statistical results on 30-D CEC2013 benchmark functions. f_1–f_5: unimodal functions; f_6–f_{20}: basic multimodal functions; f_{21}–f_{28}: composition functions (Proposal: VEGE + two proposed strategies).

Func.		DE	PSO	DE-SAP	PPSO	SSDO	RIME	SAO	VEGE	Proposal
f_1	mean	2.44×10^{4} +	2.98×10^{4} +	6.71×10^{4} +	-2.71×10^{2} +	4.62×10^{4} +	-1.38×10^{3} +	9.49×10^{3} +	-1.40×10^{3} +	$\mathbf{-1.40}\times10^{3}$
	std	2.97×10^{3}	1.14×10^{4}	8.84×10^{3}	6.02×10^{2}	3.06×10^{3}	6.43×10^{0}	4.06×10^{3}	8.37×10^{-1}	$\mathbf{1.63}\times10^{-1}$
f_2	mean	2.79×10^{8} +	3.82×10^{8} +	1.25×10^{9} +	5.84×10^{7} +	6.36×10^{8} +	2.62×10^{7} +	1.04×10^{8} +	5.43×10^{6} ≈	$\mathbf{5.35}\times10^{6}$
	std	8.22×10^{7}	2.54×10^{8}	4.14×10^{8}	2.56×10^{7}	1.25×10^{8}	1.15×10^{7}	4.91×10^{7}	2.36×10^{6}	$\mathbf{2.50}\times10^{6}$
f_3	mean	9.14×10^{10} +	2.19×10^{14} +	3.97×10^{17} +	4.97×10^{10} +	7.34×10^{15} +	1.27×10^{8} +	8.36×10^{10} +	2.98×10^{7} +	$\mathbf{5.73}\times10^{6}$
	std	1.12×10^{10}	8.96×10^{14}	1.64×10^{18}	2.91×10^{10}	1.11×10^{16}	1.52×10^{8}	5.66×10^{10}	4.89×10^{7}	$\mathbf{9.05}\times10^{6}$
f_4	mean	1.57×10^{5} +	1.95×10^{5} +	1.18×10^{5} +	7.54×10^{4} +	6.27×10^{4} +	3.29×10^{4} +	5.73×10^{4} +	1.54×10^{4} ≈	$\mathbf{1.39}\times10^{4}$
	std	1.85×10^{4}	6.80×10^{4}	2.32×10^{4}	1.61×10^{4}	2.21×10^{3}	1.03×10^{4}	6.35×10^{3}	6.52×10^{3}	$\mathbf{5.15}\times10^{3}$
f_5	mean	6.44×10^{3} +	5.19×10^{4} +	7.78×10^{4} +	6.52×10^{2} +	4.23×10^{4} +	-8.18×10^{2} +	7.37×10^{3} +	-9.97×10^{2} +	$\mathbf{-9.99}\times10^{2}$
	std	1.23×10^{3}	1.79×10^{4}	2.44×10^{4}	1.91×10^{3}	8.02×10^{3}	5.99×10^{1}	4.18×10^{3}	4.32×10^{0}	$\mathbf{2.07}\times10^{-1}$
f_6	mean	1.12×10^{3} +	4.26×10^{3} +	1.35×10^{4} +	-6.05×10^{2} +	7.16×10^{3} +	-8.19×10^{2} +	-2.77×10^{2} +	-8.32×10^{2} ≈	$\mathbf{-8.40}\times10^{2}$
	std	4.32×10^{2}	3.41×10^{3}	4.60×10^{3}	1.20×10^{2}	1.04×10^{3}	2.88×10^{1}	2.39×10^{2}	3.04×10^{1}	$\mathbf{3.62}\times10^{1}$
f_7	mean	2.10×10^{5} +	3.38×10^{6} +	1.83×10^{8} +	3.58×10^{6} +	4.37×10^{7} +	$\mathbf{1.08}\times10^{5}$ −	2.14×10^{5} +	3.42×10^{6} +	1.40×10^{5}
	std	1.32×10^{4}	4.68×10^{6}	1.85×10^{8}	5.59×10^{6}	2.89×10^{7}	$\mathbf{1.66}\times10^{4}$	1.74×10^{5}	1.24×10^{7}	4.08×10^{4}
f_8	mean	$\mathbf{-6.79}\times10^{2}$ ≈	-6.79×10^{2} ≈	-6.79×10^{2} ≈	-6.79×10^{2} ≈	-6.79×10^{2} ≈	-6.79×10^{2} ≈	-6.79×10^{2} ≈	-6.79×10^{2} ≈	-6.79×10^{2}
	std	$\mathbf{6.05}\times10^{-2}$	4.48×10^{-2}	5.20×10^{-2}	7.12×10^{-2}	6.03×10^{-2}	6.93×10^{-2}	8.65×10^{-2}	4.44×10^{-2}	4.68×10^{-2}
f_9	mean	-5.59×10^{2} +	-5.62×10^{2} +	-5.57×10^{2} +	-5.61×10^{2} +	-5.58×10^{2} +	$\mathbf{-5.75}\times10^{2}$ −	-5.70×10^{2} ≈	-5.68×10^{2} ≈	-5.71×10^{2}
	std	1.44×10^{0}	3.92×10^{0}	1.30×10^{0}	2.58×10^{0}	1.01×10^{0}	$\mathbf{5.03}\times10^{0}$	3.17×10^{0}	3.98×10^{0}	4.13×10^{0}
f_{10}	mean	2.74×10^{3} +	4.82×10^{3} +	9.35×10^{3} +	-1.02×10^{1} +	6.07×10^{3} +	-4.46×10^{2} +	1.07×10^{3} +	-4.96×10^{2} +	$\mathbf{-4.97}\times10^{2}$
	std	5.09×10^{2}	2.05×10^{3}	1.89×10^{3}	2.76×10^{2}	6.06×10^{2}	2.19×10^{1}	4.90×10^{2}	6.48×10^{-1}	$\mathbf{6.01}\times10^{-1}$
f_{11}	mean	1.04×10^{2} +	3.71×10^{2} +	6.69×10^{2} +	5.85×10^{1} +	3.94×10^{2} +	-2.88×10^{2} +	-4.05×10^{1} +	1.03×10^{2} +	$\mathbf{-3.84}\times10^{2}$
	std	3.34×10^{1}	1.66×10^{2}	1.60×10^{2}	9.32×10^{1}	4.50×10^{1}	1.77×10^{1}	7.24×10^{1}	1.03×10^{2}	$\mathbf{4.27}\times10^{0}$
f_{12}	mean	2.65×10^{2} +	4.39×10^{2} +	7.95×10^{2} +	1.89×10^{2} +	4.47×10^{2} +	$\mathbf{-1.55}\times10^{2}$ −	6.56×10^{1} ≈	1.54×10^{2} +	5.69×10^{1}
	std	4.69×10^{1}	1.67×10^{2}	1.27×10^{2}	1.09×10^{2}	3.80×10^{1}	$\mathbf{3.68}\times10^{1}$	7.28×10^{1}	1.10×10^{2}	9.84×10^{1}
f_{13}	mean	3.21×10^{2} +	5.17×10^{2} +	7.94×10^{2} +	3.62×10^{2} +	5.03×10^{2} +	$\mathbf{1.06}\times10^{1}$ −	1.66×10^{2} ≈	3.70×10^{2} +	1.91×10^{2}
	std	3.82×10^{1}	1.60×10^{2}	1.62×10^{2}	9.39×10^{1}	4.31×10^{1}	$\mathbf{2.59}\times10^{1}$	6.33×10^{1}	1.24×10^{2}	7.60×10^{1}
f_{14}	mean	6.03×10^{3} +	8.31×10^{3} +	8.35×10^{3} +	5.18×10^{3} +	8.37×10^{3} +	2.42×10^{3} +	5.32×10^{3} +	4.59×10^{3} +	$\mathbf{2.49}\times10^{2}$
	std	3.85×10^{2}	2.80×10^{2}	2.86×10^{2}	8.16×10^{2}	2.19×10^{2}	4.21×10^{2}	7.98×10^{2}	4.93×10^{2}	$\mathbf{1.66}\times10^{2}$

Table 4. *Cont.*

Func.		DE	PSO	DE-SAP	PPSO	SSDO	RIME	SAO	VEGE	Proposal
f_{15}	mean	8.11×10^{3} +	8.57×10^{3} +	8.46×10^{3} +	6.32×10^{3} +	7.82×10^{3} +	5.08×10^{3} +	5.80×10^{3} +	4.52×10^{3} ≈	$\mathbf{4.20}\times10^{3}$
	std	3.64×10^{2}	4.61×10^{2}	3.11×10^{2}	8.14×10^{2}	2.94×10^{2}	6.66×10^{2}	5.23×10^{2}	4.96×10^{2}	$\mathbf{5.88}\times10^{2}$
f_{16}	mean	2.03×10^{2} +	2.03×10^{2} +	2.03×10^{2} +	2.02×10^{2} +	2.03×10^{2} +	2.02×10^{2} +	$\mathbf{2.01}\times10^{2}$ ≈	2.02×10^{2} +	2.01×10^{2}
	std	3.23×10^{-1}	3.48×10^{-1}	3.15×10^{-1}	5.98×10^{-1}	3.54×10^{-1}	5.34×10^{-1}	$\mathbf{3.59}\times10^{-1}$	4.48×10^{-1}	3.57×10^{-1}
f_{17}	mean	1.75×10^{3} +	1.50×10^{3} +	1.63×10^{3} +	1.00×10^{3} +	1.18×10^{3} +	5.03×10^{2} +	7.43×10^{2} +	1.15×10^{3} +	$\mathbf{3.54}\times10^{2}$
	std	1.68×10^{2}	2.41×10^{2}	2.53×10^{2}	1.01×10^{2}	3.80×10^{1}	2.76×10^{1}	8.99×10^{1}	1.94×10^{2}	$\mathbf{5.91}\times10^{0}$
f_{18}	mean	1.71×10^{3} +	2.04×10^{3} +	3.31×10^{3} +	1.70×10^{3} +	2.60×10^{3} +	$\mathbf{6.28}\times10^{2}$ −	1.23×10^{3} +	1.92×10^{3} +	7.08×10^{2}
	std	1.23×10^{2}	4.13×10^{2}	3.50×10^{2}	2.82×10^{2}	1.07×10^{2}	$\mathbf{3.70}\times10^{1}$	1.94×10^{2}	3.64×10^{2}	8.71×10^{1}
f_{19}	mean	1.13×10^{5} +	2.93×10^{5} +	2.95×10^{6} +	2.72×10^{3} +	6.88×10^{5} +	$\mathbf{5.19}\times10^{2}$ ≈	8.66×10^{3} +	5.32×10^{2} +	5.19×10^{2}
	std	5.39×10^{4}	3.10×10^{5}	1.44×10^{6}	5.16×10^{3}	2.20×10^{5}	$\mathbf{3.79}\times10^{0}$	1.19×10^{4}	5.45×10^{0}	6.41×10^{0}
f_{20}	mean	6.14×10^{2} ≈	6.15×10^{2} +	6.15×10^{2} +	6.15×10^{2} +	6.15×10^{2} +	$\mathbf{6.13}\times10^{2}$ −	6.15×10^{2} +	6.15×10^{2} +	6.14×10^{2}
	std	1.94×10^{-1}	4.84×10^{-1}	2.26×10^{-7}	1.81×10^{-1}	1.95×10^{-4}	$\mathbf{7.51}\times10^{-1}$	4.62×10^{-1}	1.96×10^{-1}	7.55×10^{-1}
f_{21}	mean	4.10×10^{3} +	3.71×10^{3} +	4.50×10^{3} +	2.05×10^{3} +	3.20×10^{3} +	$\mathbf{1.67}\times10^{3}$ −	2.59×10^{3} +	1.83×10^{3} +	1.69×10^{3}
	std	1.88×10^{2}	6.52×10^{2}	5.64×10^{2}	1.24×10^{2}	8.10×10^{1}	$\mathbf{2.63}\times10^{2}$	1.85×10^{2}	2.45×10^{2}	2.69×10^{2}
f_{22}	mean	7.31×10^{3} +	9.77×10^{3} +	9.86×10^{3} +	6.56×10^{3} +	1.01×10^{4} +	3.78×10^{3} +	7.44×10^{3} +	6.26×10^{3} +	$\mathbf{1.21}\times10^{3}$
	std	3.42×10^{2}	5.71×10^{2}	4.61×10^{2}	8.86×10^{2}	2.56×10^{2}	4.96×10^{2}	9.26×10^{2}	7.09×10^{2}	$\mathbf{1.40}\times10^{2}$
f_{23}	mean	9.06×10^{3} +	9.52×10^{3} +	9.69×10^{3} +	7.59×10^{3} +	9.62×10^{3} +	6.14×10^{3} ≈	7.86×10^{3} +	5.86×10^{3} ≈	$\mathbf{5.85}\times10^{3}$
	std	3.24×10^{2}	3.09×10^{2}	3.76×10^{2}	7.65×10^{2}	2.87×10^{2}	9.04×10^{2}	1.11×10^{3}	6.87×10^{2}	$\mathbf{7.33}\times10^{2}$
f_{24}	mean	1.31×10^{3} +	1.31×10^{3} +	1.36×10^{3} +	1.32×10^{3} +	1.36×10^{3} +	$\mathbf{1.26}\times10^{3}$ −	1.29×10^{3} +	1.30×10^{3} +	1.28×10^{3}
	std	3.56×10^{0}	8.40×10^{0}	1.77×10^{1}	7.86×10^{0}	1.35×10^{1}	$\mathbf{1.02}\times10^{1}$	1.28×10^{1}	1.34×10^{1}	1.12×10^{1}
f_{25}	mean	1.41×10^{3} +	1.42×10^{3} +	1.45×10^{3} +	1.43×10^{3} +	1.47×10^{3} +	$\mathbf{1.38}\times10^{3}$ −	1.42×10^{3} +	1.43×10^{3} +	1.40×10^{3}
	std	6.05×10^{0}	8.92×10^{0}	9.25×10^{0}	1.34×10^{1}	1.52×10^{1}	$\mathbf{8.43}\times10^{0}$	1.31×10^{1}	1.42×10^{1}	1.62×10^{1}
f_{26}	mean	1.52×10^{3} ≈	1.58×10^{3} +	1.61×10^{3} +	1.59×10^{3} +	1.60×10^{3} +	1.52×10^{3} ≈	1.56×10^{3} +	1.53×10^{3} ≈	$\mathbf{1.51}\times10^{3}$
	std	7.56×10^{1}	4.77×10^{1}	3.51×10^{1}	5.02×10^{1}	3.29×10^{1}	7.11×10^{1}	5.99×10^{1}	8.66×10^{1}	$\mathbf{8.33}\times10^{1}$
f_{27}	mean	2.71×10^{3} +	2.72×10^{3} +	3.11×10^{3} +	2.75×10^{3} +	3.21×10^{3} +	$\mathbf{2.26}\times10^{3}$ −	2.55×10^{3} +	2.67×10^{3} +	2.43×10^{3}
	std	2.90×10^{1}	8.05×10^{1}	1.33×10^{2}	7.33×10^{1}	1.20×10^{2}	$\mathbf{1.00}\times10^{2}$	9.49×10^{1}	1.12×10^{2}	8.71×10^{1}
f_{28}	mean	4.91×10^{3} +	6.10×10^{3} +	7.41×10^{3} +	5.04×10^{3} +	5.70×10^{3} +	$\mathbf{2.32}\times10^{3}$ −	4.45×10^{3} +	4.93×10^{3} +	3.64×10^{3}
	std	2.10×10^{2}	7.46×10^{2}	6.90×10^{2}	5.27×10^{2}	1.83×10^{2}	$\mathbf{4.34}\times10^{2}$	3.91×10^{2}	4.48×10^{2}	1.22×10^{3}
		+/≈/−: 25/3/0	+/≈/−: 27/1/0	+/≈/−: 27/1/0	+/≈/−: 27/1/0	+/≈/−: 27/1/0	+/≈/−: 13/4/11	+/≈/−: 23/5/0	+/≈/−: 20/8/0	

Table 5. Experimental and statistical results on 50-D CEC2013 benchmark functions.

Func.		DE	PSO	DE-SAP	PPSO	SSDO	RIME	SAO	VEGE	Proposal
f_1	mean	7.75×10^{4} +	6.46×10^{4} +	9.55×10^{4} +	1.34×10^{3} +	7.03×10^{4} +	-1.28×10^{3} +	2.36×10^{4} +	-1.39×10^{3} +	$-\mathbf{1.40}\times10^{3}$
	std	7.23×10^{3}	1.47×10^{4}	1.11×10^{4}	9.91×10^{2}	3.31×10^{3}	2.69×10^{1}	5.14×10^{3}	2.03×10^{0}	$\mathbf{1.07}\times10^{0}$
f_2	mean	1.11×10^{9} +	1.63×10^{9} +	3.68×10^{9} +	1.47×10^{8} +	1.83×10^{9} +	7.47×10^{7} +	2.06×10^{8} +	$\mathbf{9.29}\times10^{6}$ −	1.15×10^{7}
	std	1.34×10^{8}	8.33×10^{8}	8.69×10^{8}	4.46×10^{7}	3.93×10^{8}	1.95×10^{7}	6.91×10^{7}	$\mathbf{2.54}\times10^{6}$	2.89×10^{6}
f_3	mean	2.92×10^{11} +	9.75×10^{14} +	5.17×10^{17} +	6.95×10^{10} +	4.29×10^{13} +	3.33×10^{9} +	1.19×10^{11} +	2.21×10^{8} +	$\mathbf{7.26}\times10^{7}$
	std	2.16×10^{10}	3.15×10^{15}	1.79×10^{18}	2.88×10^{10}	4.17×10^{13}	4.83×10^{9}	4.18×10^{10}	4.31×10^{8}	$\mathbf{1.70}\times10^{8}$
f_4	mean	2.76×10^{5} +	2.95×10^{5} +	1.85×10^{5} +	1.20×10^{5} +	8.59×10^{4} +	6.49×10^{4} +	8.72×10^{4} +	$\mathbf{1.06}\times10^{4}$ −	1.47×10^{4}
	std	2.24×10^{4}	9.53×10^{4}	5.35×10^{4}	1.95×10^{4}	3.71×10^{3}	1.35×10^{4}	6.38×10^{3}	$\mathbf{4.20}\times10^{3}$	4.77×10^{3}
f_5	mean	4.19×10^{4} +	1.06×10^{5} +	9.11×10^{4} +	1.51×10^{3} +	3.70×10^{4} +	-4.03×10^{2} +	9.54×10^{3} +	-9.95×10^{2} +	$-\mathbf{9.98}\times10^{2}$
	std	4.96×10^{3}	6.02×10^{4}	3.08×10^{4}	1.90×10^{3}	6.90×10^{3}	1.34×10^{2}	3.43×10^{3}	1.36×10^{0}	$\mathbf{6.83}\times10^{-1}$
f_6	mean	6.88×10^{3} +	5.05×10^{3} +	1.28×10^{4} +	-4.65×10^{2} +	6.72×10^{3} +	-7.86×10^{2} ≈	4.62×10^{2} +	-8.04×10^{2} ≈	$-\mathbf{8.11}\times10^{2}$
	std	9.37×10^{2}	2.49×10^{3}	3.80×10^{3}	1.17×10^{2}	5.61×10^{2}	5.00×10^{1}	4.09×10^{2}	4.49×10^{1}	$\mathbf{4.28}\times10^{1}$
f_7	mean	8.19×10^{5} +	2.86×10^{7} +	3.61×10^{8} +	4.97×10^{6} +	1.02×10^{7} +	$\mathbf{4.38}\times10^{5}$ ≈	5.85×10^{5} ≈	3.95×10^{6} +	4.69×10^{5}
	std	7.36×10^{4}	4.76×10^{7}	3.59×10^{8}	7.04×10^{6}	3.79×10^{6}	$\mathbf{6.13}\times10^{4}$	2.37×10^{5}	8.69×10^{6}	8.88×10^{4}
f_8	mean	-6.79×10^{2} ≈	-6.79×10^{2} ≈	-6.79×10^{2} ≈	-6.79×10^{2} ≈	-6.79×10^{2} ≈	$-\mathbf{6.79}\times10^{2}$ ≈	-6.79×10^{2} ≈	-6.79×10^{2} ≈	-6.79×10^{2}
	std	3.54×10^{-2}	3.53×10^{-2}	4.84×10^{-2}	6.62×10^{-2}	3.84×10^{-2}	$\mathbf{5.28}\times10^{-2}$	5.53×10^{-2}	4.25×10^{-2}	3.67×10^{-2}
f_9	mean	-5.24×10^{2} +	-5.30×10^{2} +	-5.22×10^{2} +	-5.28×10^{2} +	-5.23×10^{2} +	$-\mathbf{5.47}\times10^{2}$ −	-5.40×10^{2} +	-5.38×10^{2} +	-5.43×10^{2}
	std	1.23×10^{0}	4.42×10^{0}	1.91×10^{0}	3.59×10^{0}	1.33×10^{0}	$\mathbf{5.63}\times10^{0}$	3.54×10^{0}	4.62×10^{0}	5.21×10^{0}
f_{10}	mean	8.37×10^{3} +	9.74×10^{3} +	1.67×10^{4} +	5.27×10^{2} +	1.00×10^{4} +	-2.63×10^{2} +	2.51×10^{3} +	-4.88×10^{2} ≈	$-\mathbf{4.89}\times10^{2}$
	std	1.32×10^{3}	2.64×10^{3}	2.67×10^{3}	3.40×10^{2}	6.53×10^{2}	7.04×10^{1}	7.06×10^{2}	1.88×10^{0}	$\mathbf{2.29}\times10^{0}$
f_{11}	mean	9.78×10^{2} +	9.69×10^{2} +	1.08×10^{3} +	4.05×10^{2} +	7.17×10^{2} +	-7.11×10^{1} +	2.72×10^{2} +	5.33×10^{2} +	$-\mathbf{3.66}\times10^{2}$
	std	7.25×10^{1}	2.36×10^{2}	2.16×10^{2}	1.12×10^{2}	4.65×10^{1}	4.88×10^{1}	7.64×10^{1}	1.51×10^{2}	$\mathbf{7.80}\times10^{0}$
f_{12}	mean	1.10×10^{3} +	9.74×10^{2} +	1.31×10^{3} +	5.70×10^{2} +	8.89×10^{2} +	$\mathbf{8.24}\times10^{1}$ −	3.89×10^{2} ≈	5.69×10^{2} +	4.08×10^{2}
	std	1.07×10^{2}	2.32×10^{2}	1.68×10^{2}	1.08×10^{2}	4.11×10^{1}	$\mathbf{6.73}\times10^{1}$	9.51×10^{1}	1.55×10^{2}	1.30×10^{2}
f_{13}	mean	1.18×10^{3} +	1.02×10^{3} +	1.31×10^{3} +	7.92×10^{2} +	8.97×10^{2} +	$\mathbf{2.89}\times10^{2}$ −	5.05×10^{2} −	7.81×10^{2} +	5.80×10^{2}
	std	1.08×10^{2}	1.87×10^{2}	1.61×10^{2}	8.04×10^{1}	3.70×10^{1}	$\mathbf{7.01}\times10^{1}$	8.49×10^{1}	1.59×10^{2}	1.35×10^{2}
f_{14}	mean	1.12×10^{4} +	1.53×10^{4} +	1.49×10^{4} +	1.02×10^{4} +	1.52×10^{4} +	6.11×10^{3} +	1.08×10^{4} +	8.28×10^{3} +	$\mathbf{3.27}\times10^{2}$
	std	4.80×10^{2}	7.13×10^{2}	5.68×10^{2}	1.24×10^{3}	4.71×10^{2}	7.76×10^{2}	1.21×10^{3}	9.40×10^{2}	$\mathbf{1.62}\times10^{2}$

Table 5. *Cont.*

Func.		DE	PSO	DE-SAP	PPSO	SSDO	RIME	SAO	VEGE	Proposal
f_{15}	mean	1.50×10^{4} +	1.56×10^{4} +	1.58×10^{4} +	1.27×10^{4} +	1.51×10^{4} +	1.08×10^{4} +	1.23×10^{4} +	8.77×10^{3} ≈	**8.67**$\times10^{3}$
	std	3.75×10^{2}	4.18×10^{2}	3.84×10^{2}	1.39×10^{3}	5.10×10^{2}	1.10×10^{3}	9.49×10^{2}	7.62×10^{2}	**8.16**$\times10^{2}$
f_{16}	mean	2.04×10^{2} +	2.04×10^{2} +	2.04×10^{2} +	2.03×10^{2} +	2.04×10^{2} +	2.03×10^{2} +	**2.02**$\times10^{2}$ ≈	2.03×10^{2} +	2.02×10^{2}
	std	3.34×10^{-1}	3.08×10^{-1}	5.68×10^{-1}	7.99×10^{-1}	2.64×10^{-1}	6.25×10^{-1}	**4.24**$\times10^{-1}$	5.45×10^{-1}	6.19×10^{-1}
f_{17}	mean	4.36×10^{3} +	2.69×10^{3} +	2.34×10^{3} +	1.60×10^{3} +	1.69×10^{3} +	8.28×10^{2} +	1.19×10^{3} +	2.12×10^{3} +	**4.09**$\times10^{2}$
	std	4.57×10^{2}	5.31×10^{2}	2.83×10^{2}	1.54×10^{2}	2.83×10^{1}	6.32×10^{1}	1.05×10^{2}	3.14×10^{2}	**9.50**$\times10^{0}$
f_{18}	mean	3.99×10^{3} +	4.00×10^{3} +	4.75×10^{3} +	3.02×10^{3} +	3.78×10^{3} +	**9.54**$\times10^{2}$ ≈	2.00×10^{3} +	3.45×10^{3} +	1.02×10^{3}
	std	2.33×10^{2}	6.43×10^{2}	4.75×10^{2}	2.93×10^{2}	1.14×10^{2}	**8.31**$\times10^{1}$	2.36×10^{2}	5.13×10^{2}	1.25×10^{2}
f_{19}	mean	2.71×10^{6} +	6.27×10^{5} +	2.07×10^{6} +	4.33×10^{3} +	3.27×10^{5} +	5.56×10^{2} +	1.59×10^{4} +	5.68×10^{2} +	**5.39**$\times10^{2}$
	std	8.77×10^{5}	7.00×10^{5}	1.62×10^{6}	3.63×10^{3}	5.09×10^{4}	9.34×10^{0}	1.08×10^{4}	1.10×10^{1}	**6.63**$\times10^{0}$
f_{20}	mean	6.24×10^{2} ≈	6.25×10^{2} +	6.25×10^{2} +	6.25×10^{2} +	6.25×10^{2} +	**6.24**$\times10^{2}$ ≈	6.24×10^{2} ≈	6.24×10^{2} ≈	6.24×10^{2}
	std	2.35×10^{-1}	2.20×10^{-1}	1.91×10^{-6}	7.82×10^{-2}	6.78×10^{-2}	**6.35**$\times10^{-1}$	3.66×10^{-1}	3.66×10^{-1}	5.46×10^{-1}
f_{21}	mean	7.93×10^{3} +	6.90×10^{3} +	6.36×10^{3} +	1.78×10^{3} +	4.58×10^{3} +	1.36×10^{3} +	3.48×10^{3} +	1.19×10^{3} +	**1.14**$\times10^{3}$
	std	5.48×10^{2}	1.43×10^{3}	8.57×10^{2}	3.50×10^{2}	9.06×10^{1}	4.11×10^{2}	2.71×10^{2}	2.17×10^{2}	**3.17**$\times10^{0}$
f_{22}	mean	1.29×10^{4} +	1.73×10^{4} +	1.72×10^{4} +	1.30×10^{4} +	1.75×10^{4} +	7.88×10^{3} +	1.43×10^{4} +	1.12×10^{4} +	**1.31**$\times10^{3}$
	std	5.04×10^{2}	6.23×10^{2}	4.88×10^{2}	1.41×10^{3}	3.98×10^{2}	9.87×10^{2}	1.43×10^{3}	1.17×10^{3}	**1.33**$\times10^{2}$
f_{23}	mean	1.61×10^{4} +	1.67×10^{4} +	1.73×10^{4} +	1.48×10^{4} +	1.71×10^{4} +	1.23×10^{4} +	1.50×10^{4} +	1.11×10^{4} ≈	**1.09**$\times10^{4}$
	std	3.57×10^{2}	7.21×10^{2}	4.83×10^{2}	1.21×10^{3}	4.20×10^{2}	1.17×10^{3}	1.42×10^{3}	1.06×10^{3}	**9.74**$\times10^{2}$
f_{24}	mean	1.40×10^{3} +	1.41×10^{3} +	1.67×10^{3} +	1.44×10^{3} +	1.62×10^{3} +	**1.33**$\times10^{3}$ −	1.41×10^{3} +	1.41×10^{3} +	1.36×10^{3}
	std	5.83×10^{0}	1.68×10^{1}	1.38×10^{2}	3.20×10^{1}	7.81×10^{1}	**1.57**$\times10^{1}$	2.44×10^{1}	1.29×10^{1}	1.47×10^{1}
f_{25}	mean	1.51×10^{3} ≈	1.52×10^{3} +	1.59×10^{3} +	1.54×10^{3} +	1.62×10^{3} +	**1.46**$\times10^{3}$ −	1.53×10^{3} +	1.58×10^{3} +	1.50×10^{3}
	std	8.24×10^{0}	1.63×10^{1}	2.61×10^{1}	1.66×10^{1}	2.50×10^{1}	**1.44**$\times10^{1}$	1.93×10^{1}	2.38×10^{1}	1.84×10^{1}
f_{26}	mean	1.69×10^{3} +	1.68×10^{3} +	1.74×10^{3} +	1.70×10^{3} +	1.73×10^{3} +	**1.62**$\times10^{3}$ −	1.64×10^{3} +	1.67×10^{3} +	1.64×10^{3}
	std	5.49×10^{0}	1.17×10^{1}	4.18×10^{1}	1.41×10^{1}	7.34×10^{0}	**4.17**$\times10^{1}$	6.64×10^{1}	1.30×10^{1}	1.10×10^{1}
f_{27}	mean	3.63×10^{3} +	3.64×10^{3} +	4.77×10^{3} +	3.82×10^{3} +	4.52×10^{3} +	**2.86**$\times10^{3}$ −	3.47×10^{3} +	3.57×10^{3} +	3.25×10^{3}
	std	5.36×10^{1}	1.31×10^{2}	5.24×10^{2}	2.13×10^{2}	1.93×10^{2}	**1.35**$\times10^{2}$	1.77×10^{2}	2.51×10^{2}	1.28×10^{2}
f_{28}	mean	9.18×10^{3} +	1.27×10^{4} +	1.33×10^{4} +	9.65×10^{3} +	9.95×10^{3} +	**3.21**$\times10^{3}$ ≈	7.66×10^{3} +	8.88×10^{3} +	4.93×10^{3}
	std	4.78×10^{2}	1.74×10^{3}	1.57×10^{3}	9.01×10^{2}	3.26×10^{2}	**9.33**$\times10^{2}$	5.01×10^{2}	1.16×10^{3}	2.08×10^{3}
		+/≈/−: 25/3/0	+/≈/−: 27/1/0	+/≈/−: 27/1/0	+/≈/−: 27/1/0	+/≈/−: 27/1/0	+/≈/−: 15/6/7	+/≈/−:v 22/5/1	+/≈/−: 20/6/2	

Table 6. Experimental and statistical results on seven engineering problems.

Func.		DE	PSO	DE-SAP	PPSO	SSDO	RIME	SAO	VEGE	Proposal
WBP	mean	$2.0480\times10^{0}+$	$2.0480\times10^{0}+$	$2.3833\times10^{0}+$	$2.2624\times10^{0}+$	$1.4229\times10^{5}+$	$1.9252\times10^{0}+$	$1.9104\times10^{0}+$	$1.7238\times10^{0}+$	$\mathbf{1.6916}\times10^{0}$
	std	2.0936×10^{-1}	2.0936×10^{-1}	3.4810×10^{-1}	5.3722×10^{-1}	7.6623×10^{5}	2.2225×10^{-1}	2.5516×10^{-1}	$\mathbf{3.1670}\times10^{-2}$	4.3967×10^{-2}
	worst	2.5772×10^{0}	2.5772×10^{0}	3.1889×10^{0}	3.8419×10^{0}	4.2686×10^{6}	2.8182×10^{0}	2.8941×10^{0}	$\mathbf{1.8171}\times10^{0}$	1.9271×10^{0}
	best	1.7887×10^{0}	1.7887×10^{0}	1.8893×10^{0}	1.7229×10^{0}	2.6867×10^{0}	1.7036×10^{0}	1.6829×10^{0}	1.6848×10^{0}	$\mathbf{1.6826}\times10^{0}$
CSP	mean	$6.0775\times10^{-3}+$	$6.0775\times10^{-3}+$	$6.5158\times10^{-3}+$	$6.0990\times10^{-3}\approx$	$6.0818\times10^{-3}+$	$6.0785\times10^{-3}+$	$6.1013\times10^{-3}+$	$6.0871\times10^{-3}+$	$\mathbf{6.0761}\times10^{-3}$
	std	1.5168×10^{-6}	1.5168×10^{-6}	5.2997×10^{-4}	1.2300×10^{-4}	5.9274×10^{-6}	4.0493×10^{-6}	1.2203×10^{-4}	1.2788×10^{-5}	$\mathbf{1.3320}\times10^{-8}$
	worst	6.0811×10^{-3}	6.0811×10^{-3}	8.5403×10^{-3}	6.7614×10^{-3}	6.0986×10^{-3}	6.0978×10^{-3}	6.7582×10^{-3}	6.1218×10^{-3}	$\mathbf{6.0762}\times10^{-3}$
	best	$\mathbf{6.0761}\times10^{-3}$	$\mathbf{6.0761}\times10^{-3}$	6.0777×10^{-3}	$\mathbf{6.0761}\times10^{-3}$	6.0763×10^{-3}	6.0762×10^{-3}	$\mathbf{6.0761}\times10^{-3}$	$\mathbf{6.0761}\times10^{-3}$	$\mathbf{6.0761}\times10^{-3}$
SRD	mean	$3.0695\times10^{3}+$	$3.0695\times10^{3}+$	$3.2475\times10^{3}+$	$3.0741\times10^{3}+$	$1.2766\times10^{6}+$	$2.9963\times10^{3}+$	$2.9990\times10^{3}+$	$3.0450\times10^{3}+$	$\mathbf{2.9869}\times10^{3}$
	std	5.3510×10^{1}	5.3510×10^{1}	4.3277×10^{1}	9.4483×10^{1}	1.7862×10^{6}	7.0867×10^{0}	9.1124×10^{0}	2.7661×10^{1}	$\mathbf{1.0080}\times10^{-3}$
	worst	3.2223×10^{3}	3.2223×10^{3}	3.3639×10^{3}	3.3639×10^{3}	6.1677×10^{6}	3.0180×10^{3}	3.0377×10^{3}	3.1019×10^{3}	$\mathbf{2.9869}\times10^{3}$
	best	3.0298×10^{3}	3.0298×10^{3}	3.1683×10^{3}	2.9879×10^{3}	3.2053×10^{3}	2.9876×10^{3}	2.9886×10^{3}	3.0000×10^{3}	$\mathbf{2.9869}\times10^{3}$
TBTD	mean	$2.6419\times10^{2}+$	$2.6419\times10^{2}+$	$2.6864\times10^{2}+$	$2.6413\times10^{2}+$	$2.6999\times10^{2}+$	$2.6413\times10^{2}+$	$2.6423\times10^{2}+$	$\mathbf{2.6390}\times10^{2}\approx$	2.6390×10^{2}
	std	1.8476×10^{-1}	1.8476×10^{-1}	2.0998×10^{0}	3.8377×10^{-1}	5.0347×10^{0}	3.8677×10^{-1}	1.1078×10^{0}	$\mathbf{2.4545}\times10^{-5}$	2.2217×10^{-3}
	worst	2.6475×10^{2}	2.6475×10^{2}	2.7107×10^{2}	2.6522×10^{2}	2.8284×10^{2}	2.6553×10^{2}	2.6985×10^{2}	$\mathbf{2.6390}\times10^{2}$	2.6391×10^{2}
	best	2.6395×10^{2}	2.6395×10^{2}	2.6419×10^{2}	$\mathbf{2.6390}\times10^{2}$	2.6400×10^{2}	$\mathbf{2.6390}\times10^{2}$	$\mathbf{2.6390}\times10^{2}$	$\mathbf{2.6390}\times10^{2}$	$\mathbf{2.6390}\times10^{2}$
CBD	mean	$2.7914\times10^{0}+$	$2.7914\times10^{0}+$	$4.0830\times10^{0}+$	$2.8157\times10^{0}+$	$2.5804\times10^{0}+$	$1.3574\times10^{0}+$	$1.3415\times10^{0}+$	$1.3432\times10^{0}+$	$\mathbf{1.3402}\times10^{0}$
	std	5.7027×10^{-1}	5.7027×10^{-1}	1.8919×10^{0}	1.1259×10^{0}	6.3506×10^{-1}	1.5191×10^{-2}	1.7858×10^{-3}	1.8963×10^{-3}	$\mathbf{4.4879}\times10^{-4}$
	worst	4.2211×10^{0}	4.2211×10^{0}	8.3554×10^{0}	5.5371×10^{0}	3.8432×10^{0}	1.4100×10^{0}	1.3480×10^{0}	1.3474×10^{0}	$\mathbf{1.3423}\times10^{0}$
	best	2.1447×10^{0}	2.1447×10^{0}	1.5797×10^{0}	1.4229×10^{0}	1.3684×10^{0}	1.3409×10^{0}	1.3401×10^{0}	1.3405×10^{0}	$\mathbf{1.3400}\times10^{0}$
TCD	mean	$3.0542\times10^{1}+$	$3.0542\times10^{1}+$	$3.0904\times10^{1}+$	$3.0254\times10^{1}+$	$3.1549\times10^{1}+$	$3.0328\times10^{1}+$	$3.0153\times10^{1}+$	$\mathbf{3.0150}\times10^{1}\approx$	3.0152×10^{1}
	std	2.3954×10^{-1}	2.3954×10^{-1}	4.4253×10^{-1}	3.8837×10^{-1}	6.3834×10^{-1}	2.0913×10^{-1}	1.0563×10^{-2}	$\mathbf{5.3976}\times10^{-4}$	1.1610×10^{-2}
	worst	3.1181×10^{1}	3.1181×10^{1}	3.2113×10^{1}	3.2213×10^{1}	3.2711×10^{1}	3.0857×10^{1}	3.0208×10^{1}	$\mathbf{3.0153}\times10^{1}$	3.0214×10^{1}
	best	3.0192×10^{1}	3.0192×10^{1}	3.0287×10^{1}	$\mathbf{3.0150}\times10^{1}$	3.0289×10^{1}	3.0151×10^{1}	$\mathbf{3.0150}\times10^{1}$	$\mathbf{3.0150}\times10^{1}$	$\mathbf{3.0150}\times10^{1}$
CBHD	mean	$7.8378\times10^{0}+$	$7.8378\times10^{0}+$	$8.5143\times10^{0}+$	$7.2506\times10^{0}+$	$1.0699\times10^{1}+$	$6.8542\times10^{0}+$	$6.9393\times10^{0}+$	$6.8762\times10^{0}+$	$\mathbf{6.8430}\times10^{0}$
	std	7.1741×10^{-1}	7.1741×10^{-1}	6.6499×10^{-1}	4.0215×10^{-1}	1.2514×10^{0}	7.7909×10^{-3}	1.5303×10^{-1}	4.5820×10^{-2}	$\mathbf{3.0522}\times10^{-8}$
	worst	1.0352×10^{1}	1.0352×10^{1}	1.0069×10^{1}	8.4715×10^{0}	1.3665×10^{1}	6.8709×10^{0}	7.5109×10^{0}	7.0912×10^{0}	$\mathbf{6.8430}\times10^{0}$
	best	7.0410×10^{0}	7.0410×10^{0}	7.7228×10^{0}	6.8774×10^{0}	8.2201×10^{0}	6.8444×10^{0}	6.8431×10^{0}	6.8451×10^{0}	$\mathbf{6.8430}\times10^{0}$
		$+/\approx/-$: 7/0/0	$+/\approx/-$: 7/0/0	$+/\approx/-$: 7/0/0	$+/\approx/-$: 6/1/0	$+/\approx/-$: 7/0/0	$+/\approx/-$: 7/0/0	$+/\approx/-$: 7/0/0	$+/\approx/-$: 5/2/0	

Table 7. Ablation experimental results on 30-D CEC2013 benchmark functions.

Func	VEGE		VEGE-i		VEGE-ii		Proposal	
	mean	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**
f_1	-1.40×10^{3} +	8.37×10^{-1}	-1.40×10^{3} +	6.88×10^{-1}	$-\mathbf{1.40}\times10^{3}$ ≈	$\mathbf{1.32}\times10^{-1}$	-1.40×10^{3}	1.63×10^{-1}
f_2	5.43×10^{6} ≈	2.36×10^{6}	$\mathbf{4.84}\times10^{6}$ ≈	$\mathbf{2.22}\times10^{6}$	4.90×10^{6} ≈	1.92×10^{6}	5.35×10^{6}	2.50×10^{6}
f_3	2.98×10^{7} +	4.89×10^{7}	6.16×10^{7} +	1.89×10^{8}	$\mathbf{4.43}\times10^{6}$ ≈	$\mathbf{8.65}\times10^{6}$	5.73×10^{6}	9.05×10^{6}
f_4	1.54×10^{4} ≈	6.52×10^{3}	1.47×10^{4} ≈	6.42×10^{3}	1.44×10^{4} ≈	6.29×10^{3}	$\mathbf{1.39}\times10^{4}$	$\mathbf{5.15}\times10^{3}$
f_5	-9.97×10^{2} +	4.32×10^{0}	-9.97×10^{2} +	2.62×10^{0}	-9.99×10^{2} ≈	2.55×10^{-1}	$-\mathbf{9.99}\times10^{2}$	$\mathbf{2.07}\times10^{-1}$
f_6	-8.32×10^{2} ≈	3.04×10^{1}	-8.36×10^{2} ≈	3.15×10^{1}	-8.39×10^{2} ≈	2.99×10^{1}	$-\mathbf{8.40}\times10^{2}$	$\mathbf{3.62}\times10^{1}$
f_7	3.42×10^{6} +	1.24×10^{7}	2.45×10^{6} +	8.68×10^{6}	1.40×10^{5} ≈	2.95×10^{4}	$\mathbf{1.40}\times10^{5}$	$\mathbf{4.08}\times10^{4}$
f_8	-6.79×10^{2} ≈	4.44×10^{-2}	-6.79×10^{2} ≈	4.98×10^{-2}	$-\mathbf{6.79}\times10^{2}$ ≈	$\mathbf{6.07}\times10^{-2}$	-6.79×10^{2}	4.68×10^{-2}
f_9	-5.68×10^{2} ≈	3.98×10^{0}	-5.67×10^{2} +	3.50×10^{0}	$-\mathbf{5.73}\times10^{2}$ ≈	$\mathbf{3.57}\times10^{0}$	-5.71×10^{2}	4.13×10^{0}
f_{10}	-4.96×10^{2} ≈	6.48×10^{-1}	-4.96×10^{2} +	7.67×10^{-1}	-4.97×10^{2} ≈	9.79×10^{-1}	$-\mathbf{4.97}\times10^{2}$	$\mathbf{6.01}\times10^{-1}$
f_{11}	1.03×10^{2} +	1.03×10^{2}	6.81×10^{1} +	1.12×10^{2}	-3.83×10^{2} ≈	5.52×10^{0}	$-\mathbf{3.84}\times10^{2}$	$\mathbf{4.27}\times10^{0}$
f_{12}	1.54×10^{2} +	1.10×10^{2}	1.95×10^{2} +	8.51×10^{1}	$\mathbf{3.07}\times10^{1}$ ≈	$\mathbf{1.06}\times10^{2}$	5.69×10^{1}	9.84×10^{1}
f_{13}	3.70×10^{2} +	1.24×10^{2}	3.91×10^{2} +	9.34×10^{1}	1.98×10^{2} ≈	8.52×10^{1}	$\mathbf{1.91}\times10^{2}$	$\mathbf{7.60}\times10^{1}$
f_{14}	4.59×10^{3} +	4.93×10^{2}	4.52×10^{3} +	5.85×10^{2}	$\mathbf{2.39}\times10^{2}$ ≈	$\mathbf{1.21}\times10^{2}$	2.49×10^{2}	1.66×10^{2}
f_{15}	4.52×10^{3} ≈	4.96×10^{2}	4.55×10^{3} ≈	5.89×10^{2}	4.23×10^{3} ≈	5.42×10^{2}	$\mathbf{4.20}\times10^{3}$	$\mathbf{5.88}\times10^{2}$
f_{16}	2.02×10^{2} +	4.48×10^{-1}	2.02×10^{2} +	4.48×10^{-1}	$\mathbf{2.01}\times10^{2}$ ≈	$\mathbf{6.01}\times10^{-1}$	2.01×10^{2}	3.57×10^{-1}
f_{17}	1.15×10^{3} +	1.94×10^{2}	1.22×10^{3} +	1.42×10^{2}	$\mathbf{3.51}\times10^{2}$ ≈	$\mathbf{5.10}\times10^{0}$	3.54×10^{2}	5.91×10^{0}
f_{18}	1.92×10^{3} +	3.64×10^{2}	1.93×10^{3} +	3.67×10^{2}	7.10×10^{2} ≈	8.06×10^{1}	$\mathbf{7.08}\times10^{2}$	$\mathbf{8.71}\times10^{1}$
f_{19}	5.32×10^{2} +	5.45×10^{0}	5.33×10^{2} +	4.19×10^{0}	$\mathbf{5.19}\times10^{2}$ ≈	$\mathbf{4.93}\times10^{0}$	5.19×10^{2}	6.41×10^{0}
f_{20}	6.15×10^{2} +	1.96×10^{-1}	6.14×10^{2} ≈	1.30×10^{-1}	6.14×10^{2} ≈	6.85×10^{-1}	$\mathbf{6.14}\times10^{2}$	$\mathbf{7.55}\times10^{-1}$
f_{21}	1.83×10^{3} +	2.45×10^{2}	1.75×10^{3} +	2.96×10^{2}	1.75×10^{3} ≈	2.16×10^{2}	$\mathbf{1.69}\times10^{3}$	$\mathbf{2.69}\times10^{2}$
f_{22}	6.26×10^{3} +	7.09×10^{2}	6.30×10^{3} +	7.92×10^{2}	$\mathbf{1.21}\times10^{3}$ ≈	$\mathbf{1.82}\times10^{2}$	1.21×10^{3}	1.40×10^{2}
f_{23}	5.86×10^{3} ≈	6.87×10^{2}	6.03×10^{3} ≈	6.28×10^{2}	$\mathbf{5.70}\times10^{3}$ ≈	$\mathbf{6.43}\times10^{2}$	5.85×10^{3}	7.33×10^{2}
f_{24}	1.30×10^{3} +	1.34×10^{1}	1.30×10^{3} +	1.22×10^{1}	1.28×10^{3} ≈	9.28×10^{0}	$\mathbf{1.28}\times10^{3}$	$\mathbf{1.12}\times10^{1}$
f_{25}	1.43×10^{3} +	1.42×10^{1}	1.44×10^{3} +	1.72×10^{1}	$\mathbf{1.40}\times10^{3}$ ≈	$\mathbf{1.29}\times10^{1}$	1.40×10^{3}	1.62×10^{1}
f_{26}	1.53×10^{3} +	8.66×10^{1}	1.54×10^{3} +	8.37×10^{1}	1.52×10^{3} ≈	8.05×10^{1}	$\mathbf{1.51}\times10^{3}$	$\mathbf{8.33}\times10^{1}$
f_{27}	2.67×10^{3} +	1.12×10^{2}	2.74×10^{3} +	1.55×10^{2}	2.43×10^{3} ≈	1.24×10^{2}	$\mathbf{2.43}\times10^{3}$	$\mathbf{8.71}\times10^{1}$
f_{28}	4.93×10^{3} +	4.48×10^{2}	4.72×10^{3} +	6.73×10^{2}	$\mathbf{3.57}\times10^{3}$ ≈	$\mathbf{1.03}\times10^{3}$	3.64×10^{3}	1.22×10^{3}
	+/≈/−:20/8/0		+/≈/−:21/7/0		+/≈/−:0/28/0			

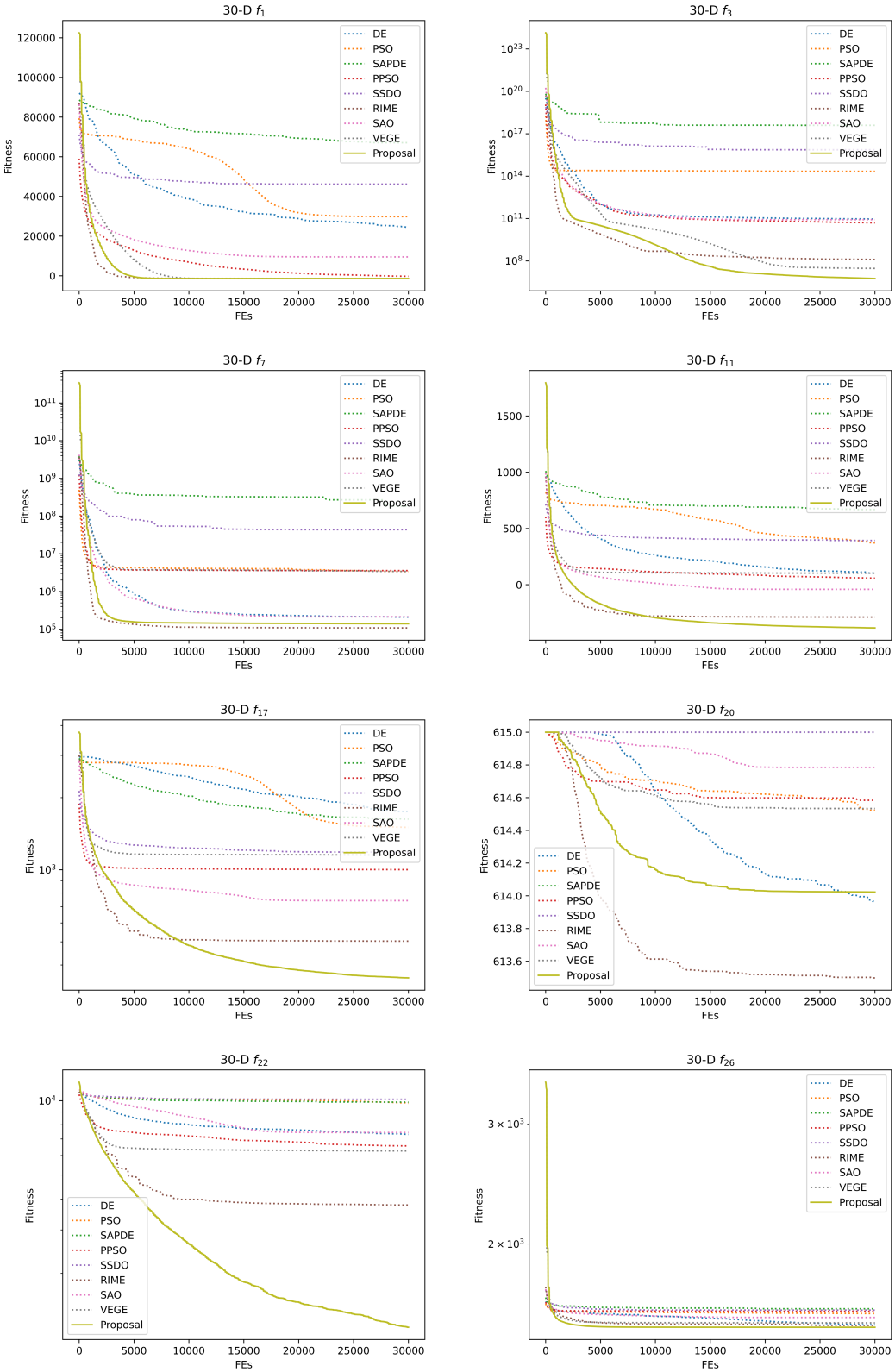

Figure 3. Convergence curves of competitor algorithms on 30-D CEC2013 representative benchmark functions (f_1 and f_3: unimodal functions; f_7, f_{11}, f_{17}, and f_{20}: multimodal functions; f_{22} and f_{26}: composite functions).

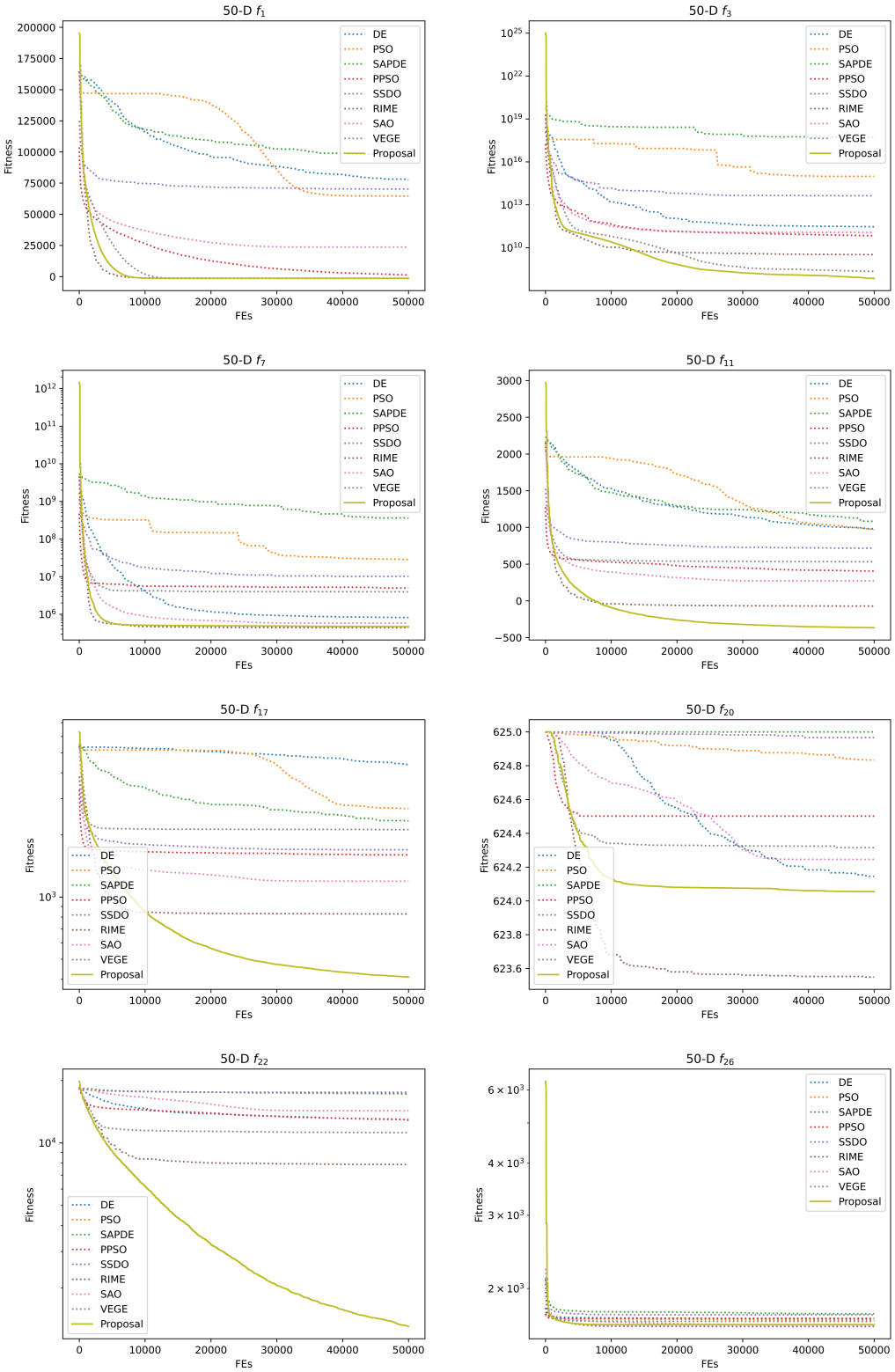

Figure 4. Convergence curves of competitor algorithms on 50-D CEC2013 representative benchmark functions.

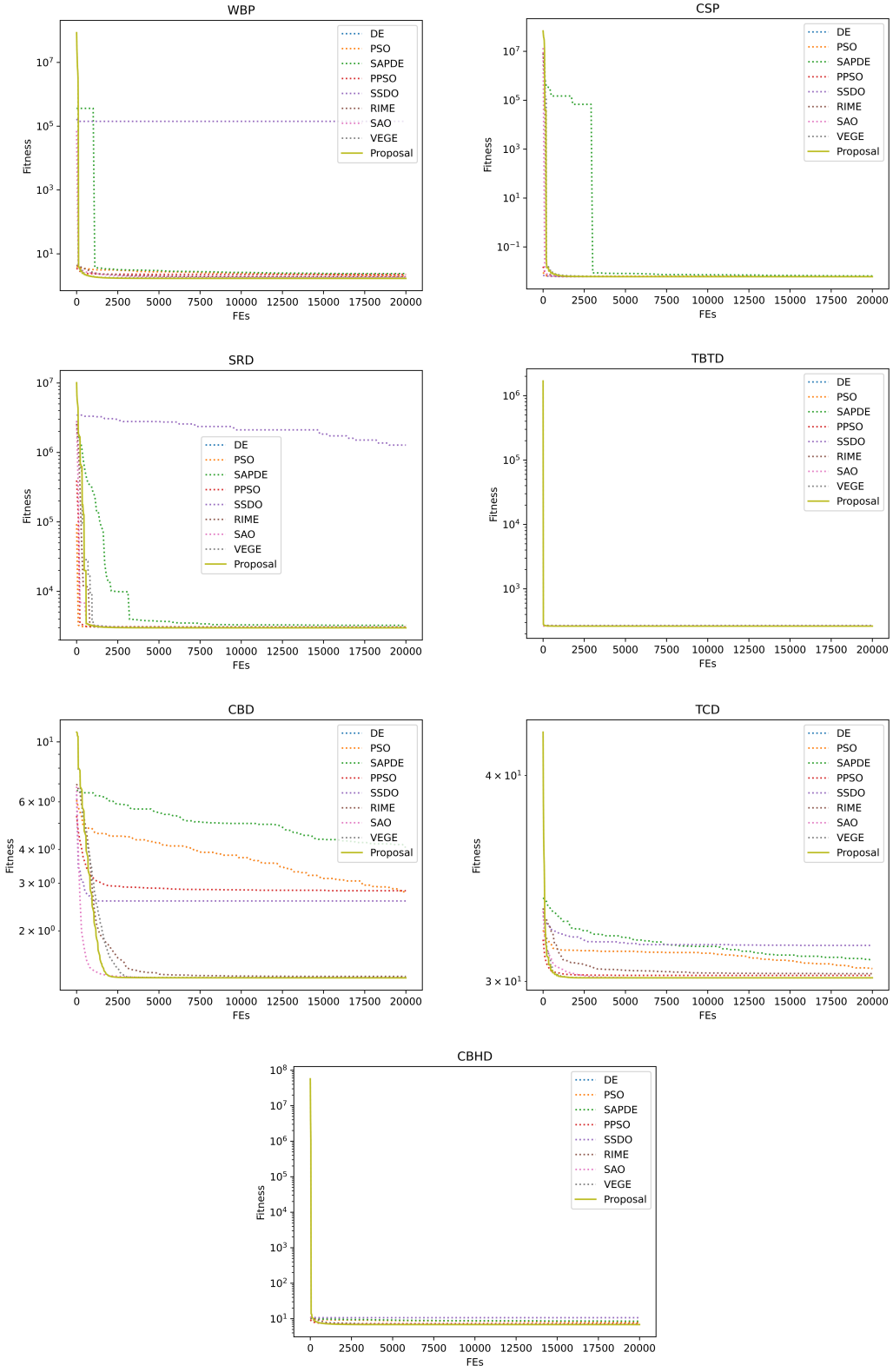

Figure 5. Convergence curves of competitor algorithms on seven engineering optimization problems.

Table 8. Ablation experimental results on 50-D CEC2013 benchmark functions.

Func	VEGE		VEGE-i		VEGE-ii		Proposal	
	mean	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**
f_1	-1.39×10^{3} +	2.03×10^{0}	-1.39×10^{3} +	1.79×10^{0}	$\mathbf{-1.40}\times10^{3}$ ≈	$\mathbf{8.74}\times10^{-1}$	-1.40×10^{3}	1.07×10^{0}
f_2	9.29×10^{6} −	2.54×10^{6}	$\mathbf{8.94}\times10^{6}$ −	$\mathbf{2.44}\times10^{6}$	1.02×10^{7} ≈	2.72×10^{6}	1.15×10^{7}	2.89×10^{6}
f_3	2.21×10^{8} +	4.31×10^{8}	2.83×10^{8} +	9.82×10^{8}	7.89×10^{7} ≈	1.36×10^{8}	$\mathbf{7.26}\times10^{7}$	$\mathbf{1.70}\times10^{8}$
f_4	$\mathbf{1.06}\times10^{4}$ −	$\mathbf{4.20}\times10^{3}$	1.08×10^{4} −	3.81×10^{3}	1.52×10^{4} ≈	6.96×10^{3}	1.47×10^{4}	4.77×10^{3}
f_5	-9.95×10^{2} +	1.36×10^{0}	-9.95×10^{2} +	1.13×10^{0}	$\mathbf{-9.98}\times10^{2}$ ≈	$\mathbf{5.06}\times10^{-1}$	-9.98×10^{2}	6.83×10^{-1}
f_6	-8.04×10^{2} ≈	4.49×10^{1}	$\mathbf{-8.15}\times10^{2}$ ≈	$\mathbf{3.37}\times10^{1}$	-8.09×10^{2} ≈	3.48×10^{1}	-8.11×10^{2}	4.28×10^{1}
f_7	3.95×10^{6} +	8.69×10^{6}	1.45×10^{6} +	1.38×10^{6}	5.28×10^{5} ≈	2.43×10^{5}	$\mathbf{4.69}\times10^{5}$	$\mathbf{8.88}\times10^{4}$
f_8	-6.79×10^{2} ≈	4.25×10^{-2}	-6.79×10^{2} ≈	3.46×10^{-2}	-6.79×10^{2} ≈	4.17×10^{-2}	$\mathbf{-6.79}\times10^{2}$	$\mathbf{3.67}\times10^{-2}$
f_9	-5.38×10^{2} +	4.62×10^{0}	-5.38×10^{2} +	4.32×10^{0}	$\mathbf{-5.45}\times10^{2}$ ≈	$\mathbf{4.91}\times10^{0}$	-5.43×10^{2}	5.21×10^{0}
f_{10}	-4.88×10^{2} ≈	1.88×10^{0}	-4.88×10^{2} ≈	2.23×10^{0}	$\mathbf{-4.90}\times10^{2}$ ≈	$\mathbf{2.42}\times10^{0}$	-4.89×10^{2}	2.29×10^{0}
f_{11}	5.33×10^{2} +	1.51×10^{2}	4.72×10^{2} +	1.39×10^{2}	-3.65×10^{2} ≈	8.90×10^{0}	$\mathbf{-3.66}\times10^{2}$	$\mathbf{7.80}\times10^{0}$
f_{12}	5.69×10^{2} +	1.55×10^{2}	5.84×10^{2} +	1.58×10^{2}	$\mathbf{3.41}\times10^{2}$ ≈	$\mathbf{9.74}\times10^{1}$	4.08×10^{2}	1.30×10^{2}
f_{13}	7.81×10^{2} +	1.59×10^{2}	7.61×10^{2} +	1.48×10^{2}	5.85×10^{2} ≈	1.52×10^{2}	$\mathbf{5.80}\times10^{2}$	$\mathbf{1.35}\times10^{2}$
f_{14}	8.28×10^{3} +	9.40×10^{2}	8.31×10^{3} +	8.13×10^{2}	$\mathbf{3.21}\times10^{2}$ ≈	$\mathbf{1.68}\times10^{2}$	3.27×10^{2}	1.62×10^{2}
f_{15}	8.77×10^{3} ≈	7.62×10^{2}	8.90×10^{3} ≈	1.04×10^{3}	$\mathbf{8.49}\times10^{3}$ ≈	$\mathbf{1.06}\times10^{3}$	8.67×10^{3}	8.16×10^{2}
f_{16}	2.03×10^{2} ≈	5.45×10^{-1}	2.03×10^{2} ≈	4.64×10^{-1}	$\mathbf{2.02}\times10^{2}$ ≈	$\mathbf{5.70}\times10^{-1}$	2.02×10^{2}	6.19×10^{-1}
f_{17}	2.12×10^{3} +	3.14×10^{2}	2.06×10^{3} +	2.77×10^{2}	$\mathbf{4.08}\times10^{2}$ ≈	$\mathbf{9.63}\times10^{0}$	4.09×10^{2}	9.50×10^{0}
f_{18}	3.45×10^{3} +	5.13×10^{2}	3.32×10^{3} +	4.18×10^{2}	1.10×10^{3} ≈	1.41×10^{2}	$\mathbf{1.02}\times10^{3}$	$\mathbf{1.25}\times10^{2}$
f_{19}	5.68×10^{2} +	1.10×10^{1}	5.71×10^{2} +	1.00×10^{1}	5.41×10^{2} ≈	6.26×10^{0}	$\mathbf{5.39}\times10^{2}$	$\mathbf{6.63}\times10^{0}$
f_{20}	6.24×10^{2} ≈	3.66×10^{-1}	6.24×10^{2} ≈	2.83×10^{-1}	6.24×10^{2} ≈	5.92×10^{-1}	$\mathbf{6.24}\times10^{2}$	$\mathbf{5.46}\times10^{-1}$
f_{21}	1.19×10^{3} +	2.17×10^{2}	1.15×10^{3} +	1.31×10^{0}	$\mathbf{1.13}\times10^{3}$ ≈	$\mathbf{3.75}\times10^{0}$	1.14×10^{3}	3.17×10^{0}
f_{22}	1.12×10^{4} +	1.17×10^{3}	1.10×10^{4} +	1.23×10^{3}	1.32×10^{3} ≈	1.77×10^{2}	$\mathbf{1.31}\times10^{3}$	$\mathbf{1.33}\times10^{2}$
f_{23}	1.11×10^{4} ≈	1.06×10^{3}	1.12×10^{4} ≈	9.20×10^{2}	$\mathbf{1.07}\times10^{4}$ ≈	$\mathbf{1.12}\times10^{3}$	1.09×10^{4}	9.74×10^{2}
f_{24}	1.41×10^{3} +	1.29×10^{1}	1.42×10^{3} +	2.55×10^{1}	$\mathbf{1.36}\times10^{3}$ ≈	$\mathbf{1.79}\times10^{1}$	1.36×10^{3}	1.47×10^{1}
f_{25}	1.58×10^{3} +	2.38×10^{1}	1.58×10^{3} +	2.92×10^{1}	1.51×10^{3} ≈	2.02×10^{1}	$\mathbf{1.50}\times10^{3}$	$\mathbf{1.84}\times10^{1}$
f_{26}	1.67×10^{3} +	1.30×10^{1}	1.67×10^{3} +	1.45×10^{1}	$\mathbf{1.64}\times10^{3}$ ≈	$\mathbf{4.53}\times10^{1}$	1.64×10^{3}	1.10×10^{1}
f_{27}	3.57×10^{3} +	2.51×10^{2}	3.60×10^{3} +	2.49×10^{2}	3.25×10^{3} ≈	1.59×10^{2}	$\mathbf{3.25}\times10^{3}$	$\mathbf{1.28}\times10^{2}$
f_{28}	8.88×10^{3} +	1.16×10^{3}	8.96×10^{3} +	1.81×10^{3}	$\mathbf{4.65}\times10^{3}$ ≈	$\mathbf{1.83}\times10^{3}$	4.93×10^{3}	2.08×10^{3}
+/≈/−: 19/7/2			+/≈/−: 19/7/2		+/≈/−: 0/28/0			

5. Discussion

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

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

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

Next, we want to discuss the potential of the proposed two strategies and provide some open topics. Not limited to the conventional VEGE, our proposal can be easily combined with other improved versions of the VEGE. Since the two strategies are separable, we can also use either of them to combine with VEGE. Thus, a topic worthy of further research is to dynamically select the combination of strategies according to the characteristics of the optimization problem. We simply used three different mutation methods to simulate the mutation patterns of real plants, which is far from enough because the real situation is more diverse and complex. Thus, how to add more diverse mutation methods is also one of our future topics. Since the distribution of individuals is constantly changing with the convergence of the population, the probabilities of different mutation methods being executed should also be different. Therefore, how to efficiently use mutations to guide the convergence of the algorithm is also a promising topic. Although we have only given a few topics, there are still many other interesting topics and hope to give some inspiration to the latecomers.

Subsequently, we apply the Kruskal–Wallis test and Holm's multiple comparison test to check for significant differences among the compared algorithms on the CEC2013 benchmark functions. The results of the statistical tests show that our two proposed strategies can further improve the performance of the conventional VEGE on most optimization problems, and the deterioration situation is rare and only happens in 50-D f_2 and f_4, which fully proves the effectiveness of our two proposed strategies. Moreover, from the experimental and statistical results compared with optimization algorithms in Tables 4 and 5, our proposal is quite competitive with these state-of-the-art optimization algorithms, and this conclusion can be also observed from the convergence curve of the optimization processes. Due to the population size in VEGE, our proposal being variable, and the initial population size of our proposal being 10 while the other compared algorithms are set to 100, the beginning of the convergence curve in VEGE and our proposal is worse than that of other algorithms, but the excellent exploration and exploitation ability drives our proposal to outperform the compared algorithms rapidly, and the final optimum found by our proposal is also superior, which shows the domination of our proposal over the competitor algorithms in practice.

However, in some multimodal functions, our proposed VEGE with two strategies is significantly inferior to RIME (e.g., 30-D f_7 and f_9), and we attempt to explain this degeneration by the No Free Lunch Theory (NFL) [36]. The NFL states that any pair of black-box optimization algorithms has an identical averaging performance in all possible problems, and, if an algorithm performs well on a certain category of problems, it must degenerate on the rest of the problems since it is the only way to achieve identical averaging performance. Although the improvement from VEGE to our proposal can be observed, the original skeleton limits the performance of VEGE in these functions, and we will further analyze the reasons for the failure and give corresponding countermeasures in our future work.

In addition, the ablation experiment results in Tables 7 and 8 show that the second strategy brings a greater performance improvement than the first strategy on many functions, and the difference between the combination of the two and the second strategy alone is not obvious. This also supports our previous topic, that is, how to reasonably select search strategies is one of the important means by which to improve performance.

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

6. Conclusions

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

We will continue to analyze the ecosystem of real plants and use new findings to continuously improve the performance of the conventional VEGE. In future research, we will focus on extending VEGE to various complex optimization tasks such as multi-objective problems [37,38], multimodal problems [39,40], large-scale global optimization problems [41,42], expensive optimization problems [43,44], and feature selection tasks [45,46]. Furthermore, we will also try to apply these improved algorithms to various real-world optimization problems [47,48].

Author Contributions: Conceptualization, R.Z. and J.Y.; Data curation, R.Z. and J.Y.; Formal analysis, R.Z. and E.Z.; Funding acquisition, R.Z. and M.M.; Investigation, R.Z. and F.P.; Methodology, R.Z., F.P. and J.Y.; Project administration, M.M.; Resources, R.Z., J.Y. and M.M.; Software, R.Z. and M.M.; Supervision, J.Y.; Validation, R.Z., F.P., E.Z., J.Y. and M.M.; Visualization, R.Z.; Writing—original draft, R.Z.; Writing – review and editing, R.Z., F.P., E.Z., J.Y. and M.M. All authors have read and agreed to the published version of the manuscript.

Funding: This work was founded by JSPS KAKENHI Grant Number JP20K11967 and JST SPRING Grant Number JPMJSP2119.

Informed Consent Statement: Not applicable.

Data Availability Statement: The code of this research can be found at https://github.com/RuiZhong961230/IVEGE.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Devi, R.V.; Sathya, S.S.; Coumar, M.S. Evolutionary algorithms for de novo drug design—A survey. *Appl. Soft Comput.* **2015**, *27*, 543–552. [CrossRef]
2. Houssein, E.H.; Hosney, M.E.; Oliva, D.; Mohamed, W.M.; Hassaballah, M. A novel hybrid Harris hawks optimization and support vector machines for drug design and discovery. *Comput. Chem. Eng.* **2020**, *133*, 106656. . [CrossRef]
3. Lambrinidis, G.; Tsantili-Kakoulidou, A. Multi-objective optimization methods in novel drug design. *Expert Opin. Drug Discov.* **2021**, *16*, 647–658. [CrossRef] [PubMed]
4. Rovito, L.; Bonin, L.; Manzoni, L.; De Lorenzo, A. An Evolutionary Computation Approach for Twitter Bot Detection. *Appl. Sci.* **2022**, *12*, 5915. [CrossRef]
5. Lingam, G.; Ranjan Rout, R.; Somayajulu, D. Deep Q-Learning and Particle Swarm Optimization for Bot Detection in Online Social Networks. In Proceedings of the 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kanpur, India, 6–8 July 2019; pp. 1–6. [CrossRef]
6. Lingam, G.; Rout, R.R.; Somayajulu, D.V.L.N.; Ghosh, S.K. Particle Swarm Optimization on Deep Reinforcement Learning for Detecting Social Spam Bots and Spam-Influential Users in Twitter Network. *IEEE Syst. J.* **2021**, *15*, 2281–2292. [CrossRef]
7. Akhmedova, S.; Stanovov, V.; Kamiya, Y. A Hybrid Clustering Approach Based on Fuzzy Logic and Evolutionary Computation for Anomaly Detection. *Algorithms* **2022**, *15*, 342. [CrossRef]
8. Ghezelbash, R.; Maghsoudi, A.; Carranza, E.J.M. Optimization of geochemical anomaly detection using a novel genetic K-means clustering (GKMC) algorithm. *Comput. Geosci.* **2020**, *134*, 104335. [CrossRef]
9. Di Pasquale, G.; Saracino, A.; Bosso, L.; Russo, D.; Moroni, A.; Bonanomi, G.; Allevato, E. Coastal Pine-Oak Glacial Refugia in the Mediterranean Basin: A Biogeographic Approach Based on Charcoal Analysis and Spatial Modelling. *Forests* **2020**, *11*, 673. [CrossRef]
10. Buonincontri, M.P.; Bosso, L.; Smeraldo, S.; Chiusano, M.L.; Pasta, S.; Di Pasquale, G. Shedding light on the effects of climate and anthropogenic pressures on the disappearance of Fagus sylvatica in the Italian lowlands: evidence from archaeo-anthracology and spatial analyses. *Sci. Total Environ.* **2023**, *877*, 162893. [CrossRef]
11. Rivera-Lopez, R.; Canul-Reich, J.; Mezura-Montes, E.; Cruz-Chávez, M.A. Induction of decision trees as classification models through metaheuristics. *Swarm Evol. Comput.* **2022**, *69*, 101006. [CrossRef]

12. Mehrabian, A.; Lucas, C. A novel numerical optimization algorithm inspired from weed colonization. *Ecol. Inform.* **2006**, *1*, 355–366. [CrossRef]
13. Jia, H.; Peng, X.; Lang, C. Remora optimization algorithm. *Expert Syst. Appl.* **2021**, *185*, 115665. [CrossRef]
14. Zhong, C.; Li, G.; Meng, Z. Beluga whale optimization: A novel nature-inspired metaheuristic algorithm. *Knowl. Based Syst.* **2022**, *251*, 109215. [CrossRef]
15. Rahmani, A.M.; AliAbdi, I. Plant competition optimization: A novel metaheuristic algorithm. *Expert Syst.* **2022**, *39*, e12956. [CrossRef]
16. Yu, J. Vegetation Evolution: An Optimization Algorithm Inspired by the Life Cycle of Plants. *Int. J. Comput. Intell. Appl.* **2022**, *21*, 2250010. [CrossRef]
17. Storn, R.; Price, K.V. Differential Evolution—A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
18. Kennedy, J.; Eberhart, R. Particle Swarm Optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948. [CrossRef]
19. Zheng, S.; Janecek, A.; Tan, Y. Enhanced Fireworks Algorithm. In Proceedings of the 2013 IEEE Congress on Evolutionary Computation, Sendai, Japan, 25–28 May 2013; pp. 2069–2077. [CrossRef]
20. Yu, J.; Takagi, H. Accelerating Vegetation Evolution with Mutation Strategy and Gbased Growth Strategy. In Proceedings of the 2019 IEEE Symposium Series on Computational Intelligence (SSCI), Xiamen, China, 6–9 December 2019; pp. 3033–3039. [CrossRef]
21. Yu, J.; Takagi, H. Multi-Species Generation Strategy-Based Vegetation Evolution. In Proceedings of the 2020 IEEE Congress on Evolutionary Computation (CEC), Giza, Egypt , 24–26 October 2020; pp. 1–6. [CrossRef]
22. Yu, J.; Takagi, H. Performance Analysis of Vegetation Evolution. In Proceedings of the 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), Bari, Italy, 6–9 October 2019; pp. 2214–2219. [CrossRef]
23. Holland, J.H. Outline for a Logical Theory of Adaptive Systems. *J. ACM* **1962**, *9*, 297–314. [CrossRef]
24. Katoch, S.; Chauhan, S.S.; Kumar, V. A review on genetic algorithm: past, present, and future. *Multimed. Tools Appl.* **2021**, *80*, 8091–8126. [CrossRef]
25. Yang, X.S. Flower Pollination Algorithm for Global Optimization. In *Proceedings of the Unconventional Computation and Natural Computation*; Durand-Lose, J., Jonoska, N., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 240–249. [CrossRef]
26. Zhao, S.; Zhang, T.; Ma, S.; Chen, M. Dandelion Optimizer: A Nature-Inspired Metaheuristic Algorithm for Engineering Applications. *Eng. Appl. Artif. Intell.* **2022**, *114*, 105075. [CrossRef]
27. Liang, J.; Qu, B.; Suganthan, P.; Hernández-Díaz, A. *Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization*; Technical Report 201212; Computational Intelligence Laboratory, Zhengzhou University: Zhengzhou, China, 2013.
28. Thieu, N.V. *ENOPPY: A Python Library for Engineering Optimization Problems*; Zenodo: Geneva, Switzerland, 2023. [CrossRef]
29. Coello Coello, C.A. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. *Comput. Methods Appl. Mech. Eng.* **2002**, *191*, 1245–1287. . [CrossRef]
30. Van Thieu, N.; Mirjalili, S. MEALPY: An open-source library for latest meta-heuristic algorithms in Python. *J. Syst. Archit.* **2023**, *139*, 102871. [CrossRef]
31. Teo, J. Exploring dynamic self-adaptive populations in differential evolution. *Soft Comput.* **2006**, *10*, 673–686. [CrossRef]
32. Ghasemi, M.; Akbari, E.; Rahimnejad, A.; Razavi, E.; Ghavidel, S.; Li, L. Phasor particle swarm optimization: A simple and efficient variant of PSO. *Soft Comput.* **2019**, *23*, 9701–9718. [CrossRef]
33. Tharwat, A.; Gabel, T. Parameters optimization of support vector machines for imbalanced data using social ski driver algorithm. *Neural Comput. Appl.* **2020**, *32*. [CrossRef]
34. Su, H.; Zhao, D.; Heidari, A.A.; Liu, L.; Zhang, X.; Mafarja, M.; Chen, H. RIME: A physics-based optimization. *Neurocomputing* **2023**, *532*, 183–214. [CrossRef]
35. Deng, L.; Liu, S. Snow ablation optimizer: A novel metaheuristic technique for numerical optimization and engineering design. *Expert Syst. Appl.* **2023**, *225*, 120069. [CrossRef]
36. Wolpert, D.; Macready, W. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
37. Premkumar, M.; Jangir, P.; Sowmya, R.; Alhelou, H.H.; Heidari, A.A.; Chen, H. MOSMA: Multi-Objective Slime Mould Algorithm Based on Elitist Non-Dominated Sorting. *IEEE Access* **2021**, *9*, 3229–3248. [CrossRef]
38. Houssein, E.H.; Mahdy, M.A.; Shebl, D.; Manzoor, A.; Sarkar, R.; Mohamed, W.M. An efficient slime mould algorithm for solving multi-objective optimization problems. *Expert Syst. Appl.* **2022**, *187*, 115870. [CrossRef]
39. Ahmed, R.; Nazir, A.; Mahadzir, S.; Shorfuzzaman, M.; Islam, J. Niching Grey Wolf Optimizer for Multimodal Optimization Problems. *Appl. Sci.* **2021**, *11*. [CrossRef]
40. Wang, B.; Liu, L.; Li, Y.; Khishe, M. Robust Grey Wolf Optimizer for Multimodal Optimizations: A Cross-Dimensional Coordination Approach. *J. Sci. Comput.* **2022**, *92*, 110. [CrossRef]
41. Sun, Y.; Wang, X.; Chen, Y.; Liu, Z. A modified whale optimization algorithm for large-scale global optimization problems. *Expert Syst. Appl.* **2018**, *114*, 563–577. [CrossRef]
42. Chakraborty, S.; Saha, A.K.; Chakraborty, R.; Saha, M. An enhanced whale optimization algorithm for large scale optimization problems. *Knowl. Based Syst.* **2021**, *233*, 107543. [CrossRef]

43. Berkan Aydilek, İ. A hybrid firefly and particle swarm optimization algorithm for computationally expensive numerical problems. *Appl. Soft Comput.* **2018**, *66*, 232–249. [CrossRef]
44. Liu, Y.; Liu, J.; Jin, Y. Surrogate-Assisted Multipopulation Particle Swarm Optimizer for High-Dimensional Expensive Optimization. *IEEE Trans. Syst. Man, Cybern. Syst.* **2022**, *52*, 4671–4684. [CrossRef]
45. Elminaam, D.S.A.; Nabil, A.; Ibraheem, S.A.; Houssein, E.H. An Efficient Marine Predators Algorithm for Feature Selection. *IEEE Access* **2021**, *9*, 60136–60153. [CrossRef]
46. Abd Elaziz, M.; Ewees, A.A.; Yousri, D.; Abualigah, L.; Al-qaness, M.A.A. Modified Marine Predators Algorithm for Feature Selection: Case Study Metabolomics. *Knowl. Inf. Syst.* **2022**, *64*, 261–287. [CrossRef]
47. Xiao, Q.; Li, C.; Tang, Y.; Pan, J.; Yu, J.; Chen, X. Multi-component energy modeling and optimization for sustainable dry gear hobbing. *Energy* **2019**, *187*, 115911. [CrossRef]
48. Alsalibi, B.; Mirjalili, S.; Yahya, R.; Gandomi, A.; Abualigah, L. A Comprehensive Survey on the Recent Variants and Applications of Membrane-Inspired Evolutionary Algorithms. *Arch. Comput. Methods Eng.* **2022**, *29*, 1–17. [CrossRef]