mathematics

Article

An Adaptive Dimension Weighting Spherical Evolution to Solve Continuous Optimization Problems

Yifei Yang, Sichen Tao, Shibo Dong, Masahiro Nomura and Zheng Tang *

Faculty of Engineering, University of Toyama, Toyama-shi 930-8555, Japan; yyf773906764@gmail.com (Y.Y.); taosc73@hotmail.com (S.T.); m01811137@gmail.com (S.D.); d2372003@ems.u-toyama.ac.jp (M.N.)
* Correspondence: ztang@eng.u-toyama.ac.jp

Abstract: The spherical evolution algorithm (SE) is a unique algorithm proposed in recent years and widely applied to new energy optimization problems with notable achievements. However, the existing improvements based on SE are deemed insufficient due to the challenges arising from the multiple choices of operators and the utilization of a spherical search method. In this paper, we introduce an enhancement method that incorporates weights in individuals' dimensions that are affected by individual fitness during the iteration process, aiming to improve SE by adaptively balancing the tradeoff between exploitation and exploration during convergence. This is achieved by reducing the randomness of dimension selection and enhancing the retention of historical information in the iterative process of the algorithm. This new SE improvement algorithm is named DWSE. To evaluate the effectiveness of DWSE, in this study, we apply it to the CEC2017 standard test set, the CEC2013 large-scale global optimization test set, and 22 real-world problems from CEC2011. The experimental results substantiate the effectiveness of DWSE in achieving improvement.

Keywords: spherical evolution; evolutionary computation; weighting allocation; adaptive strategy

MSC: 68T01; 68T05; 68T20

check for updates

Citation: Yang, Y.; Tao, S.; Dong, S.; Nomura, M.; Tang, Z. An Adaptive Dimension Weighting Spherical Evolution to Solve Continuous Optimization Problems. *Mathematics* **2023**, *11*, 3733. https://doi.org/10.3390/math11173733

Academic Editors: Dhananjay R. Thiruvady, Su Nguyen and Yuan Sun

Received: 1 August 2023
Revised: 22 August 2023
Accepted: 29 August 2023
Published: 30 August 2023

1. Introduction

Optimization algorithms, as solutions designed to optimize complex real-world problems, have been a popular area of research for the past few decades [1]. These algorithms can generally be divided into two categories: metaheuristic algorithms (MHAs) and heuristic algorithms [2]. MHAs are often based on laws derived inductively from real-world phenomena, which are then transformed into algorithms. On the other hand, heuristic algorithms are traditional mathematical methods or improved versions of MHAs. Several early classical MHAs, including the genetic algorithm (GA), particle swarm optimization (PSO), the simulated annealing algorithm (SA) and differential evolution (DE), have significantly influenced the field of optimization due to their wide range of applications [3–7]. These algorithms stand firmly on their own as highly effective solutions. Additionally, metaphorical MHAs such as the artificial bee colony algorithm (ABC), ant colony optimization (ACO), and the whale optimization algorithm (WOA) add to the diversity of approaches [8–10]. Conversely, the sine cosine algorithm (SCA) and covariance matrix adaptation evolution strategy (CMA-ES) fall into the category of mathematically inspired MHAs, further broadening the spectrum of optimization techniques [11,12]. Despite their wide-ranging definitions and interpretations, most optimization algorithms are designed around two core components: the operators and strategies employed within the algorithms [13].

The operator, which denotes the formula used to update individuals, constitutes one of the fundamental cores of the algorithm. Operators usually contain individuals, step sizes, and operations that combine them [14,15]. The algorithm encompasses the selection of the previous generation of individuals and the multiplicity of search steps.

The selection of individuals significantly impacts the diversity of populations within the algorithm and the retention of historical information [16]. Meanwhile, the search step size plays a crucial role in determining the convergence speed of the algorithm [17]. The impact of strategies extends beyond the operator components, and it can be argued that the presence of strategies differentiates algorithms from purely mathematical methods [18]. The design of strategies often serves a specific purpose; for instance, the strategy change of individual division of labor in the swarm intelligence algorithm aims to conduct further exploitation in the valuable regions discovered by the algorithm, ultimately leading to better solutions [19]. Similarly, the mutation strategy in genetic algorithms broadens the search direction, preventing premature convergence to local optima [20]. However, the use of strategies entails both favorable and unfavorable effects. For instance, the greedy selection strategy may improve an individual's fitness, yet it also limits the algorithm's ability to escape local optima [21]. Irrespective of the operator design and strategy choice, the primary objective is to strike a delicate balance between exploitation and exploration during the search process [22]. Exploitation and exploration, as a set of antonyms, represent distinct facets of an algorithm's behavior, and the algorithm's state may lean more towards one aspect at any given time [23]. Exploitation emphasizes the local search ability and convergence speed of an algorithm, while exploration highlights its global search ability and capacity to jump out of local optima [24]. Achieving a balance between these two aspects is the overarching aim. Ideally, an algorithm should converge rapidly and find the global optimal solution effectively. However, this ideal scenario encounters a practical limitation due to MHAs often having to contend with NP-hard problem [25] that an algorithm cannot achieve both fast convergence and the best solution simultaneously. Nonetheless, through a thoughtful and rational design, it is possible to strike a compromise between the convergence speed and the quality of the solution. This embodies the essence of the balance between exploitation and exploration.

Algorithm improvements are often rooted in changes to operators or strategies or the fusion of two distinct algorithms [26–29]. Hybrid algorithms, which are designed to amalgamate algorithms with different tendencies, serve the purpose of striking a better balance between exploitation and exploration. These improvements typically have specific objectives, like enabling a more exploitation-oriented algorithm to break out of local optima or enhancing the local search capabilities of an exploration-oriented algorithm. Within the realm of numerous MHAs and their improved versions, the spherical evolution algorithm (SE) is popular due to its wide range of applications in engineering optimization, such as parameter estimation for photovoltaic models and optimization of wind farm layouts [30–34]. However, compared to other algorithms, SE has seen relatively few improvements. This is primarily attributed to its unique search strategy, which relies entirely on a geometric space search, and the highly stochastic nature of its search pattern, which is achieved by utilizing an operator that updates individual step sizes using Euclidean distance. One notable successful improvement of SE is the linear population size reduction-based SE algorithm (LSE) [35]. LSE effectively compensates for the higher randomness in SE, thereby bolstering its exploitation ability while maintaining exploration capabilities. By emulating the improvement strategy of LSHADE, the search pattern of LSE is somewhat altered, enabling a more focused exploitation during later iterations [36]. Another attempt at improvement is the cooperative coevolution wingsuit flying search algorithm with SE (CCWFSSE) [37]. However, it falls short of being deemed a successful hybrid algorithm due to its use of unreasonable population size settings in the experimental phase and the absence of a comparison with the original SE. A previous study introducing CSE employed a differential evolution approach using coefficients of chaotic mappings for elite individuals in SE iterations, which exhibited promising performance enhancements [38]. The results obtained in previous studies support the assertion that improving SE is a challenging task.

Weighting methods commonly employed in evolutionary computation, such as the introduction of weight to each algorithm in a hybrid algorithm, are simple yet effective techniques [39]. Some studies have explored the idea of assigning weights to individual dimensions of the algorithm, aiming to identify the dimensions requiring the most significant changes or those necessitating minimal adjustments [40]. However, this approach presents challenges, since estimating the value of a dimension in a black-box optimization problem is inherently difficult. Prematurely fixing the value of a dimension can inadvertently lead the algorithm towards a local optimum. Therefore, weighting is often used for deterministic problems such as image classification rather than NP-hard problems. In fact, the dimension selection mechanism employed in SE is essentially a form of weight selection. SE defines the number of dimensions (DSF) updated by an individual as a random integer in the range of $[6, 10]$, which, for a 30-dimensional problem, equates to assigning a weight ranging from 20% to 33.3% to each dimension of the individual. This stochastic weighting mechanism prevents the algorithm from prematurely converging to a local optimum, but it also compromises its exploitation capability. The primary focus of the improvement in this paper lies in modifying the original weight (DSF). Specifically, we introduce a modification by replacing the random selection approach with a more sophisticated weighted selection approach. The weight assigned to dimension selection is influenced by the quality of the solution, as well as historical information. This dimensionally weight-based improvement in SE is referred to as DWSE. To enhance the efficiency of DWSE, we incorporated a cooperative strategy by segmenting the population of SE into multiple subsets. Within each segment, individuals share the same set of weights, fostering a cooperative synergy that yields similar effects.

To evaluate the performance of DWSE, we conducted extensive evaluations utilizing three test sets sourced from the IEEE Congress on Evolutionary Computation (CEC). These sets encompass thirty single-objective continuous function optimization problems drawn from CEC2017, which encompass single-peak functions, simple multipeak functions, hybrid functions, and composite functions. Additionally, we included twenty-two real-world problems sourced from CEC2011, along with a comprehensive large-scale global optimization test set derived from CEC2013. The function expressions for these test sets are publicly accessible on the official IEEE CEC website. Furthermore, we are committed to providing the problem models, along with the corresponding open-source data. The outcomes of our experimentation unequivocally underscore the substantial enhancements achieved by DWSE, distinctly positioning it with advantages over other SE improvement algorithms.

This work aims to make the following contributions:

- For the first time in SE, we apply a weighting method to dimension selection with adaptive capability;
- DWSE has significant advantages over SE and other SE improvement algorithms for constrained single-objective optimization problems.

In Section 2, we introduce the proposed DWSE. In Section 3, experiments and analysis using DWSE based on standard test sets, real-world problems, and large-scale optimization problems are presented to validate the improvements. Finally, in Section 4, we provide a comprehensive summary of the conclusions drawn from this research paper.

2. The Proposed DWSE

2.1. Spherical Evolution

SE denotes the update unit of most MHAs in the hypercubic search (rectangular in 2D) approach as:

$$SS(C_{i,j}, D_{i,j}) = ScaleFun01_{i,j}() \cdot (ScaleFun02_{i,j}() - ScaleFun03_{i,j}()), \\ i = 1, 2, \cdots, popusize; j = 1, 2, \cdots, dim. \tag{1}$$

where $ScaleFun$ is a function for adjusting the difference between C and D, $popusize$ is the population size, and dim is the dimension of the problem. The unique innovation of SE lies in the fact that instead of the hypercubic search method used in Equation (1), it uses an original spherical (hypersphere in the high-dimensional case) search method. Its formula can be expressed as:

$$SS(C_{i,j}, D_{i,j}) = ScaleFun_{i,j}() \cdot \|C_{i,j} - D_{i,j}\| \cdot \cos\theta, \tag{2}$$

$$SS(C_{i,j}, D_{i,j}) = \begin{cases} ScaleFun_{i,j}() \cdot \|C_{i,*} - D_{i,*}\|_2 \cdot \sin\theta, & j = 1,2 \\ ScaleFun_{i,j}() \cdot \|C_{i,*} - D_{i,*}\|_2 \cdot \cos\theta, & j = 1,2 \end{cases} \tag{3}$$

$$SS(C^k_{i,j}, D^k_{i,j}) = \begin{cases} ScaleFun_{i,j}() \cdot \|C_{i,*} - D_{i,*}\|_2 \cdot \prod_{k=j}^{dim-1} \sin\theta_j, & j = 1 \\ ScaleFun_{i,j}() \cdot \|C_{i,*} - D_{i,*}\|_2 \cdot \prod_{k=j}^{dim-1} \sin\theta_j \cdot \cos\theta_{j-1}, & 1 < j < dim - 1 \\ ScaleFun_{i,j}() \cdot \|C_{i,*} - D_{i,*}\|_2 \cdot \cos\theta_{j-1}. & j = dim \end{cases} \tag{4}$$

where Equation (2) is used to search for the one-dimensional case, Equation (3) is used to search for the two-dimensional case, and Equation (4) is used to search for other cases. This search has a higher degree of directional randomization than hypercube search, giving it a considerable advantage in specific, more exploratory problems.

SE also provides seven operators based on spherical evolution and recommends a fourth operator. All operators in SE can be represented as follows:

(1) SE/current-to-best/1

$$X^{t+1}_{i,j} = X^t_{i,j} + SS_m(X^t_{i,j}, X^t_{g,j}) + SS_m(X^t_{r1,j}, X^t_{r2,j}), \tag{5}$$

(2) SE/best/1

$$X^{t+1}_{i,j} = X^t_{g,j} + SS_m(X^t_{r1,j}, X^t_{r2,j}), \tag{6}$$

(3) SE/best/2

$$X^{t+1}_{i,j} = X^t_{g,j} + SS_m(X^t_{r1,j}, X^t_{r2,j}) + SS_m(X^t_{r3,j}, X^t_{r4,j}), \tag{7}$$

(4) SE/rand/1

$$X^{t+1}_{i,j} = X^t_{r1,j} + SS_m(X^t_{r2,j}, X^t_{r3,j}), \tag{8}$$

(5) SE/rand/2

$$X^{t+1}_{i,j} = X^t_{r1,j} + SS_m(X^t_{r2,j}, X^t_{r3,j}) + SS_m(X^t_{r3,j}, X^t_{r4,j}), \tag{9}$$

(6) SE/current/1

$$X^{t+1}_{i,j} = X^t_{i,j} + SS_m(X^t_{r2,j}, X^t_{r3,j}), \tag{10}$$

(7) SE/current/2

$$X^{t+1}_{i,j} = X^t_{i,j} + SS_m(X^t_{r2,j}, X^t_{r3,j}) + SS_m(X^t_{r4,j}, X^t_{r5,j}). \tag{11}$$

2.2. Weight-Based Dimension Selection Method

As mentioned in the background, determining the weights of each dimension at each moment in a black-box optimization problem poses a challenge. To address this issue, DWSE uses an iterative method of weight grouping to obtain new weights. The population of individuals is equally divided into 10 segments, with each segment assigned a set of weights, initially set at 0.5. This weight represents the number of dimensions that can be selected for updating in the current individual, as opposed to SE's random selection process. In DWSE, the number of dimensions to be selected for updating in an individual is calculated as $DSF = dim \times W$, where DSF denotes the number of dimensions, dim represents the total number of dimensions of an individual, and W corresponds to the weight of the following individual. The method employed for selecting which specific dimensions require updating adheres to the SE approach. Among all the dimensions of an individual, the DSF dimensions that necessitate updating are selected evenly. The k value used to calibrate and select the dimensions (DSF) is then randomly chosen from $[1, dim]$.

In the weight updating process of DWSE, a combination of a greedy strategy and a standard normal distribution significantly influences the weight assignment. Specifically, during SE iterations, individuals that have been successfully updated are preserved and assigned corresponding weights. Conversely, the remaining individuals retain only the weights from their previous generation. The total weights are subsequently updated uniformly based on the standard normal distribution. Among the ten individuals, those that have not undergone updating are recorded as 0, while those that have been updated are assigned a value reflecting their progress in adaptation. The update value is calculated as $val = fitold - fit$, where fit and $fitold$ represent the current fitness and the fitness of the previous generation, respectively. Subsequently, the values of the ten individuals undergo normalization, which determines the extent of weight adjustment required. Note that the reason for choosing a normal distribution is its property of ensuring equal probability distributions for increments and decrements, with decreasing values as the center distance increases.

$$W = \sum_{i=1}^{10} \frac{W \cdot val_i}{fitold_i}, \tag{12}$$

In the context of this weight retention approach, when the majority of individuals in the group achieve successful iterations, the value of W will be larger, causing the algorithm to exhibit a stronger bias towards local search. Conversely, if the majority of individuals are not retained, the value of W will be smaller, leading to a bias in the algorithm towards updating fewer dimensions. Following the completion of the weight retention process, the weights are updated based on a standard normal distribution, expressed as follows:

$$W = \frac{1}{0.1\sqrt{2\pi}} exp\left(-\frac{(R-W)^2}{0.02}\right). \tag{13}$$

where R represents a random number within the range of 0 to 1. Equations (12) and (13) demonstrate the process of updating weights for a set of individuals. As DWSE is divided into ten groups of individuals, and the aforementioned process must be repeated ten times. Figure 1 shows the difference between DWSE and SE in terms of strategy. The weighting strategy of DWSE in the figure has an additional layer of feedback process compared to SE, which is obviously more favorable to the evolutionary process of the algorithm.

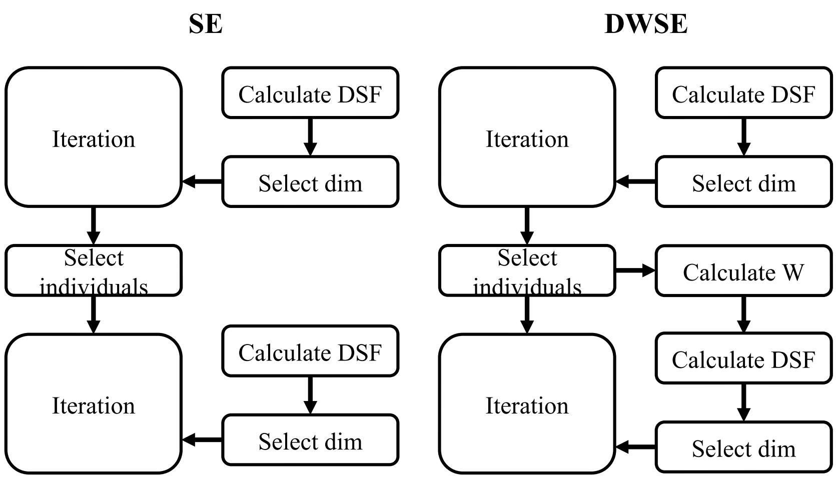

Figure 1. DWSE vs. SE in terms of strategy.

2.3. Step Size Update Method Based on Success History Information

In SE, the step size (also referred to as the scaling factor) is determined by a random number constrained by the population size, which may not provide sufficient guidance for achieving rapid convergence in continuous optimization. In DWSE, the update strategy for the step size involves utilizing the previous generation's step size (SF) under the successfully updated W as the midpoint, which is then randomly adjusted within a specific range. A probability distribution approximating a function of the Cauchy distribution is a more desirable outcome based on the need for a smaller floating range of variation. The update formula for the step size can be expressed as follows:

$$SF_i = SF_i + 0.1 \cdot \tan[\pi \cdot (R - 0.5)]. \tag{14}$$

where R is a random number from to 0 to 1. Compared to SE, which depends entirely on random numbers, the step size of DWSE retains some historical information, which facilitates the precise exploitation process of the algorithm. Figure 2 shows the float range of SF. SF does not vary in float beyond ± 0.2 in most cases, which moderates the step size variation of the algorithm and helps to maintain fast convergence.

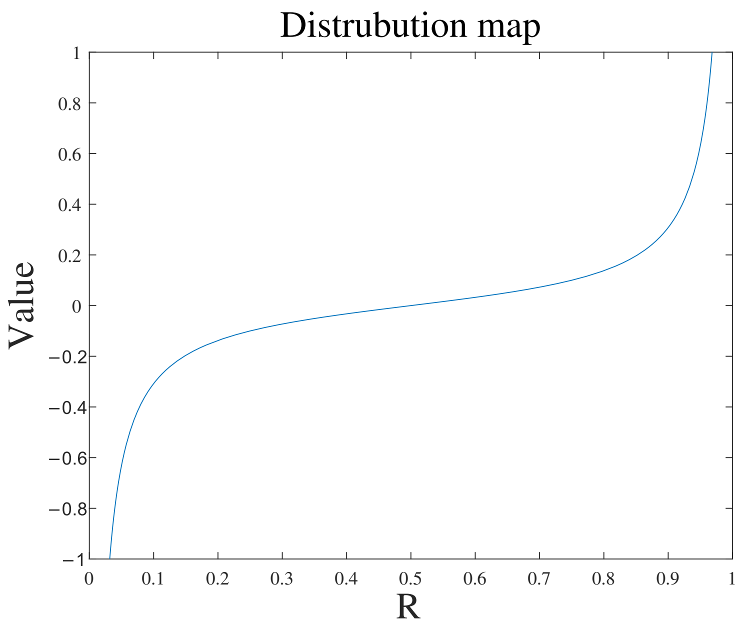

Figure 2. Distribution map of float range for SF.

2.4. Algorithm

Pseudocode Algorithm 1 shows the specific implementation of DWSE. W and SF are entered as a 1×10 matrix with an initial value of 0.5 for ease of calculation. The output is the best result obtained in the last iteration.

Algorithm 1: DWSE.

Input: N, dim, W, SF, FES;
Initialization;
$\{X_1, X_2, \ldots, X_N\}, nFES = 0, fitold = fit$;
while $nFES \leq FES$ **do**
  for $i = 1$ *to* N **do**
    $g = \lfloor \frac{i-1}{10} + 1 \rfloor$;
    $W_i \leftarrow W_g$;
    $DSF_i \leftarrow$ get the dimension number by $DSF_i = dim \times W_i$;
    $SS_{i,j} \leftarrow$ calculate the updated value by Equations (2)–(4);
    $X_i \leftarrow$ get the new individual by Equation (8);
    $fit(X_i) \leftarrow$ calculate $fit(X_i)$;
    if $fit(X_i) < fit_i$ **then**
      $fit_i \leftarrow fit(X_i)$;
    end
    $nFES \leftarrow nFES + 1$;
    $val_i \leftarrow$ update the value by $val = fitold - fit$;
  end
  for $g = 1$ *to* 10 **do**
    $W_g \leftarrow$ update by Equations (12) and (13);
  end
  $SF \leftarrow$ update the step size by Equation (14);
  $fitold \leftarrow fit$ update the $fitold$;
end
Output: The best obtained solution

3. Experimental Results and Discussion

For the experimental part, the number of evaluations was set to 10,000 $\times$ dim for the CEC2011 and CEC2017 problems and 3000 $\times$ dim for the CEC2013 LSGO problems. The population size of all SE improvement algorithms including DWSE was set to 100. In the tables, mean and std represent the average value and standard deviation, respectively, and data marked in bold represent the best mean. $W/T/L$ is the result of the comparison of $win/tie/loss$ under the Wilcoxon rank-sum test at $p = 0.05$ (In general, the threshold for determining significant differences is 5%). After std, $+/=/-$ represents the $win/tie/loss$ relationship under that problem. *Rank* denotes results based on Friedman mean rankings, and bolded font denotes the best mean under the problem. All results are based on repeated runs in MATLAB (51 times for CEC2017 and CEC2011; 30 for at CEC2013 LSGO) on a device with an i7-9700 CPU and 36GB RAM.

3.1. Experimental Results and Analysis in CEC2017

Tables 1 and 2 present the experimental results of SEs in CEC2017, encompassing results under four problem dimensions: 10, 30, 50, and 100. The results reveal that DWSE holds a significant advantage over all other SE improvement algorithms; this advantage is particularly pronounced when dealing with smaller problem dimensions. The dimension weighting mechanism of DWSE allows for a much more effective exploitation, surpassing other SE algorithms in terms of local search capabilities. Even in higher dimensions, DWSE maintains a competitive edge, despite a slight decline in performance. This is attributed to the impact of dimensional weights under grouping, which accentuates the disparity in operator weights among different groups, preserving sufficient randomness.

Table 1. Experimental results in CEC2017.

Dimension	**10**										
Fun	**DWSE**		**SE**			**LSE**			**CSE**		
	Mean	**std**	**Mean**	**std**		**Mean**	**std**		**Mean**	**std**	
F1	4.218×10^{1}	3.670×10^{1}	1.110×10^{3}	1.328×10^{3}	+	4.245×10^{2}	4.942×10^{2}	+	8.967×10^{2}	1.188×10^{3}	+
F2	3.967×10^{-1}	2.833×10^{0}	1.771×10^{3}	2.389×10^{3}	+	4.958×10^{2}	7.280×10^{2}	+	1.637×10^{3}	1.778×10^{3}	+
F3	2.525×10^{3}	3.744×10^{3}	5.462×10^{2}	2.429×10^{2}	+	1.553×10^{2}	1.288×10^{2}	+	5.122×10^{2}	2.204×10^{2}	+
F4	1.328×10^{0}	1.980×10^{-1}	5.624×10^{0}	4.857×10^{-1}	+	5.269×10^{0}	3.961×10^{-1}	+	5.450×10^{0}	5.507×10^{-1}	+
F5	4.142×10^{0}	1.177×10^{0}	8.289×10^{0}	1.568×10^{0}	+	6.644×10^{0}	1.647×10^{0}	+	8.328×10^{0}	1.609×10^{0}	+
F6	8.695×10^{-9}	1.366×10^{-8}	8.917×10^{-15}	3.087×10^{-14}	−	0.000×10^{0}	0.000×10^{0}	−	1.337×10^{-14}	3.699×10^{-14}	−
F7	1.506×10^{1}	1.234×10^{0}	2.103×10^{1}	2.329×10^{0}	+	1.954×10^{1}	1.932×10^{0}	+	2.062×10^{1}	1.638×10^{0}	+
F8	4.988×10^{0}	1.227×10^{0}	8.519×10^{0}	1.745×10^{0}	+	7.126×10^{0}	2.081×10^{0}	+	8.822×10^{0}	1.846×10^{0}	+
F9	8.917×10^{-15}	3.834×10^{-14}	5.338×10^{-11}	2.878×10^{-10}	+	6.509×10^{-13}	1.257×10^{-12}	+	6.239×10^{-12}	4.019×10^{-11}	+
F10	1.282×10^{2}	7.429×10^{1}	4.277×10^{2}	9.345×10^{1}	+	2.846×10^{2}	1.233×10^{2}	+	4.198×10^{2}	1.061×10^{2}	+
F11	2.879×10^{0}	7.496×10^{-1}	3.959×10^{0}	8.380×10^{-1}	+	3.121×10^{0}	8.322×10^{-1}	=	3.965×10^{0}	8.899×10^{-1}	+
F12	1.519×10^{5}	1.763×10^{5}	4.230×10^{5}	2.670×10^{5}	+	2.257×10^{5}	1.336×10^{5}	+	2.952×10^{5}	1.629×10^{5}	+
F13	1.565×10^{1}	3.815×10^{0}	1.241×10^{3}	8.272×10^{2}	+	9.763×10^{2}	7.924×10^{2}	+	5.160×10^{2}	4.405×10^{2}	+
F14	5.089×10^{0}	2.055×10^{0}	2.248×10^{1}	1.264×10^{1}	+	1.757×10^{1}	9.547×10^{0}	+	1.663×10^{1}	6.420×10^{0}	+
F15	2.299×10^{0}	5.755×10^{-1}	9.037×10^{1}	1.066×10^{2}	+	7.376×10^{1}	1.015×10^{2}	+	2.645×10^{1}	2.058×10^{1}	+
F16	1.894×10^{0}	4.586×10^{-1}	1.738×10^{0}	2.947×10^{-1}	−	1.344×10^{0}	2.805×10^{-1}	−	1.698×10^{0}	3.131×10^{-1}	−
F17	2.534×10^{0}	1.131×10^{0}	3.822×10^{0}	1.438×10^{0}	+	2.636×10^{0}	1.350×10^{0}	=	3.739×10^{0}	1.198×10^{0}	+
F18	2.177×10^{1}	4.223×10^{0}	1.468×10^{3}	1.087×10^{3}	+	8.686×10^{2}	6.500×10^{2}	+	1.120×10^{3}	7.433×10^{2}	+
F19	1.233×10^{0}	3.791×10^{-1}	4.403×10^{1}	6.043×10^{1}	+	3.608×10^{1}	6.430×10^{1}	+	3.377×10^{0}	1.528×10^{0}	+
F20	8.325×10^{-2}	7.361×10^{-2}	1.111×10^{0}	6.078×10^{-1}	+	5.293×10^{-1}	4.013×10^{-1}	+	1.037×10^{0}	4.810×10^{-1}	+
F21	1.196×10^{2}	2.666×10^{1}	1.381×10^{2}	3.990×10^{1}	+	1.474×10^{2}	4.207×10^{1}	+	1.368×10^{2}	3.990×10^{1}	=
F22	9.785×10^{1}	1.180×10^{1}	1.003×10^{2}	2.786×10^{-1}	+	1.000×10^{2}	1.198×10^{-1}	−	1.003×10^{2}	2.870×10^{-1}	+
F23	3.013×10^{2}	3.556×10^{1}	3.069×10^{2}	2.222×10^{1}	+	3.068×10^{2}	2.054×10^{0}	=	3.091×10^{2}	1.528×10^{0}	+
F24	1.966×10^{2}	8.602×10^{1}	2.396×10^{2}	1.062×10^{2}	+	2.298×10^{2}	1.012×10^{2}	+	2.340×10^{2}	1.048×10^{2}	+
F25	3.991×10^{2}	6.383×10^{0}	4.068×10^{2}	1.470×10^{1}	+	4.098×10^{2}	1.861×10^{1}	+	4.073×10^{2}	1.564×10^{1}	+
F26	2.765×10^{2}	8.146×10^{1}	3.000×10^{2}	8.533×10^{-13}	−	3.000×10^{2}	0.000×10^{0}	−	3.000×10^{2}	1.548×10^{-12}	−
F27	3.901×10^{2}	1.717×10^{0}	3.916×10^{2}	2.174×10^{0}	+	3.926×10^{2}	2.081×10^{0}	+	3.911×10^{2}	2.133×10^{0}	+
F28	2.672×10^{2}	9.603×10^{1}	3.080×10^{2}	9.336×10^{1}	+	3.224×10^{2}	7.773×10^{1}	+	3.160×10^{2}	6.250×10^{1}	+
F29	2.503×10^{2}	1.522×10^{1}	2.667×10^{2}	9.150×10^{0}	+	2.636×10^{2}	7.153×10^{0}	+	2.691×10^{2}	7.645×10^{0}	+
F30	4.836×10^{3}	3.375×10^{3}	1.460×10^{5}	1.514×10^{5}	+	1.166×10^{5}	8.630×10^{3}	+	1.468×10^{5}	1.168×10^{5}	+
W/T/L	−/−/−		27/0/3			23/3/4			26/1/3		
Rank	1.30		3.40			2.30			3.00		
Dimension	**30**										
Fun	**DWSE**		**SE**			**LSE**			**CSE**		
	mean	**std**	**mean**	**std**		**mean**	**std**		**mean**	**std**	
F1	6.474×10^{-3}	9.240×10^{-3}	1.900×10^{3}	2.945×10^{3}	+	1.168×10^{3}	1.486×10^{3}	+	7.178×10^{2}	8.118×10^{2}	+
F2	3.535×10^{12}	1.058×10^{13}	3.164×10^{13}	8.376×10^{13}	+	1.748×10^{17}	2.197×10^{17}	+	2.182×10^{17}	3.212×10^{17}	+
F3	1.122×10^{5}	1.985×10^{5}	7.405×10^{3}	2.810×10^{3}	−	5.561×10^{5}	9.178×10^{3}	−	5.686×10^{5}	7.730×10^{3}	−
F4	9.259×10^{1}	1.530×10^{1}	8.083×10^{1}	2.095×10^{1}	−	1.056×10^{2}	8.527×10^{0}	+	1.051×10^{2}	8.288×10^{0}	+
F5	3.000×10^{1}	4.322×10^{0}	4.239×10^{1}	7.824×10^{0}	+	6.402×10^{1}	7.374×10^{0}	+	7.295×10^{1}	7.623×10^{0}	+
F6	1.137×10^{-13}	0.000×10^{0}	1.928×10^{-4}	1.349×10^{-3}	+	1.137×10^{-13}	0.000×10^{0}	=	1.605×10^{-13}	5.651×10^{-14}	+
F7	6.352×10^{1}	5.150×10^{0}	7.831×10^{1}	8.042×10^{0}	+	1.040×10^{2}	8.871×10^{0}	+	1.128×10^{2}	9.672×10^{0}	+
F8	3.588×10^{1}	4.393×10^{0}	4.642×10^{1}	7.819×10^{0}	+	6.973×10^{1}	8.356×10^{0}	+	7.829×10^{1}	7.559×10^{0}	+
F9	6.001×10^{-1}	4.285×10^{0}	3.698×10^{-1}	1.247×10^{0}	+	3.815×10^{-6}	2.627×10^{-5}	+	2.489×10^{0}	1.494×10^{0}	+
F10	1.880×10^{3}	2.390×10^{2}	2.343×10^{3}	3.183×10^{2}	+	3.505×10^{3}	2.874×10^{2}	+	3.792×10^{3}	2.180×10^{2}	+
F11	2.561×10^{1}	1.393×10^{1}	2.858×10^{1}	2.334×10^{1}	=	8.127×10^{1}	2.121×10^{1}	+	9.910×10^{1}	1.769×10^{1}	+
F12	4.119×10^{5}	4.166×10^{5}	8.621×10^{5}	5.536×10^{5}	+	1.520×10^{6}	5.005×10^{5}	+	1.901×10^{6}	5.339×10^{5}	+
F13	7.504×10^{3}	6.369×10^{3}	6.227×10^{3}	5.004×10^{3}	=	1.978×10^{5}	8.759×10^{3}	+	4.416×10^{5}	1.733×10^{5}	+
F14	1.200×10^{3}	3.074×10^{3}	5.763×10^{5}	4.374×10^{5}	+	3.899×10^{5}	2.654×10^{5}	+	5.690×10^{3}	3.106×10^{3}	+
F15	1.923×10^{2}	5.878×10^{2}	1.483×10^{3}	1.364×10^{3}	+	5.930×10^{3}	3.842×10^{3}	+	5.373×10^{3}	3.075×10^{3}	+
F16	3.989×10^{2}	1.499×10^{2}	5.214×10^{2}	1.438×10^{2}	+	5.093×10^{2}	1.202×10^{2}	+	6.154×10^{2}	1.294×10^{2}	+
F17	7.037×10^{1}	2.503×10^{1}	1.020×10^{2}	7.339×10^{1}	=	8.328×10^{1}	2.472×10^{1}	+	1.177×10^{2}	3.591×10^{1}	+
F18	2.120×10^{5}	1.399×10^{5}	1.705×10^{5}	9.223×10^{5}	−	2.162×10^{5}	8.494×10^{5}	=	1.434×10^{5}	6.177×10^{5}	−
F19	5.133×10^{2}	1.448×10^{3}	1.713×10^{3}	1.793×10^{3}	+	8.625×10^{3}	4.787×10^{3}	+	8.048×10^{3}	6.022×10^{3}	+
F20	1.113×10^{2}	5.584×10^{1}	1.570×10^{2}	7.121×10^{1}	+	9.321×10^{1}	5.164×10^{1}	−	1.534×10^{2}	5.285×10^{1}	+
F21	2.359×10^{2}	5.198×10^{0}	2.440×10^{2}	1.762×10^{1}	+	2.694×10^{2}	8.641×10^{0}	+	2.757×10^{2}	1.564×10^{1}	+
F22	1.023×10^{2}	6.160×10^{0}	1.827×10^{2}	4.172×10^{2}	+	1.143×10^{2}	2.116×10^{0}	+	1.364×10^{2}	1.092×10^{1}	+
F23	3.828×10^{2}	5.986×10^{0}	3.938×10^{2}	8.117×10^{0}	+	4.149×10^{2}	8.296×10^{0}	+	4.244×10^{2}	7.944×10^{0}	+
F24	4.557×10^{2}	6.443×10^{0}	4.836×10^{2}	1.233×10^{1}	+	5.008×10^{2}	1.464×10^{1}	+	5.154×10^{2}	1.053×10^{1}	+
F25	3.874×10^{2}	2.734×10^{-1}	3.871×10^{2}	7.208×10^{-1}	−	3.872×10^{2}	2.053×10^{-1}	−	3.873×10^{2}	1.480×10^{-1}	−
F26	1.312×10^{3}	1.204×10^{2}	1.079×10^{3}	5.078×10^{2}	=	1.216×10^{3}	3.594×10^{2}	=	1.348×10^{3}	2.833×10^{2}	=
F27	5.091×10^{2}	3.647×10^{0}	5.114×10^{2}	4.347×10^{0}	+	5.157×10^{2}	2.969×10^{0}	+	5.169×10^{2}	3.708×10^{0}	+
F28	3.299×10^{2}	4.899×10^{1}	4.100×10^{2}	1.711×10^{1}	+	4.133×10^{2}	3.866×10^{0}	+	4.261×10^{2}	5.351×10^{0}	+
F29	4.968×10^{2}	3.597×10^{1}	5.399×10^{2}	6.938×10^{1}	+	5.529×10^{2}	4.198×10^{1}	+	5.889×10^{2}	4.974×10^{1}	+
F30	4.934×10^{3}	2.072×10^{3}	6.114×10^{3}	2.233×10^{3}	+	1.672×10^{5}	7.040×10^{3}	+	2.589×10^{5}	1.075×10^{5}	+
W/T/L	−/−/−		22/4/4			24/3/3			26/1/3		
Rank	1.52		2.23			2.75			3.50		

Table 2. Experimental results in CEC2017.

Dimension						**50**					
Fun	**DWSE**		**SE**			**LSE**			**CSE**		
	Mean	**std**	**Mean**	**std**		**Mean**	**std**		**Mean**	**std**	
F1	4.043×10^{2}	1.135×10^{3}	2.086×10^{3}	2.360×10^{3}	+	6.505×10^{3}	6.210×10^{3}	+	9.163×10^{3}	6.214×10^{3}	+
F2	1.674×10^{25}	6.113×10^{25}	2.050×10^{26}	5.857×10^{26}	+	9.826×10^{29}	1.241×10^{29}	+	1.000×10^{30}	1.421×10^{14}	+
F3	2.363×10^{5}	2.433×10^{5}	4.302×10^{5}	8.548×10^{3}	−	1.497×10^{5}	1.849×10^{5}	−	1.469×10^{5}	1.706×10^{5}	−
F4	1.047×10^{2}	6.048×10^{1}	1.069×10^{2}	3.115×10^{1}	=	1.308×10^{2}	1.987×10^{1}	+	1.444×10^{2}	1.385×10^{1}	+
F5	6.855×10^{1}	8.209×10^{0}	1.031×10^{2}	1.484×10^{1}	+	1.437×10^{2}	1.424×10^{1}	+	1.717×10^{2}	1.372×10^{1}	+
F6	1.142×10^{-2}	2.440×10^{-2}	4.484×10^{-4}	2.372×10^{-3}	−	2.809×10^{-13}	5.731×10^{-14}	−	3.299×10^{-13}	3.414×10^{-14}	−
F7	1.265×10^{2}	8.621×10^{0}	1.622×10^{2}	1.405×10^{1}	+	2.069×10^{2}	1.411×10^{1}	+	2.350×10^{2}	1.498×10^{1}	+
F8	7.400×10^{1}	7.443×10^{0}	1.066×10^{2}	1.549×10^{1}	+	1.442×10^{2}	1.324×10^{1}	+	1.683×10^{2}	1.089×10^{1}	+
F9	1.373×10^{1}	4.203×10^{1}	4.606×10^{1}	4.398×10^{1}	+	2.644×10^{1}	1.613×10^{1}	+	2.496×10^{2}	8.526×10^{1}	+
F10	3.672×10^{3}	3.021×10^{2}	4.539×10^{3}	3.894×10^{2}	+	6.293×10^{3}	4.575×10^{2}	+	6.903×10^{3}	3.461×10^{2}	+
F11	6.151×10^{1}	1.789×10^{1}	6.612×10^{1}	1.544×10^{1}	+	1.881×10^{2}	5.531×10^{1}	+	1.641×10^{2}	3.906×10^{1}	+
F12	7.415×10^{5}	1.014×10^{6}	4.938×10^{6}	1.997×10^{6}	+	1.005×10^{7}	2.592×10^{6}	+	1.081×10^{7}	3.009×10^{6}	+
F13	1.963×10^{3}	2.158×10^{3}	2.936×10^{3}	1.858×10^{3}	+	1.607×10^{5}	9.187×10^{3}	+	5.314×10^{5}	2.681×10^{5}	+
F14	1.311×10^{5}	1.258×10^{5}	7.213×10^{5}	4.287×10^{5}	+	4.336×10^{5}	2.330×10^{5}	+	5.014×10^{5}	2.858×10^{5}	−
F15	3.721×10^{3}	3.647×10^{3}	2.554×10^{3}	2.329×10^{3}	=	8.368×10^{3}	3.464×10^{3}	+	9.348×10^{3}	4.252×10^{3}	+
F16	9.398×10^{2}	2.287×10^{2}	1.173×10^{3}	2.219×10^{2}	+	1.079×10^{3}	1.690×10^{2}	+	1.168×10^{3}	2.006×10^{2}	+
F17	6.126×10^{2}	1.463×10^{2}	7.072×10^{2}	1.923×10^{2}	+	6.460×10^{2}	1.495×10^{2}	=	8.101×10^{2}	1.309×10^{2}	+
F18	1.172×10^{6}	4.891×10^{5}	1.556×10^{6}	1.075×10^{6}	=	1.198×10^{6}	6.425×10^{5}	=	4.650×10^{5}	2.113×10^{5}	−
F19	1.362×10^{5}	6.780×10^{3}	5.225×10^{3}	3.837×10^{3}	−	1.600×10^{5}	4.939×10^{3}	=	1.055×10^{5}	3.853×10^{3}	−
F20	4.854×10^{2}	2.130×10^{2}	5.919×10^{2}	1.515×10^{2}	+	4.689×10^{2}	1.356×10^{2}	=	5.971×10^{2}	1.422×10^{2}	+
F21	2.761×10^{2}	7.131×10^{0}	3.090×10^{2}	1.427×10^{1}	+	3.504×10^{2}	1.448×10^{1}	+	3.773×10^{2}	1.180×10^{1}	+
F22	3.216×10^{3}	1.915×10^{3}	4.610×10^{3}	1.536×10^{3}	+	4.941×10^{3}	3.040×10^{3}	+	5.150×10^{3}	3.242×10^{3}	+
F23	5.067×10^{2}	1.393×10^{1}	5.468×10^{2}	1.651×10^{1}	+	5.824×10^{2}	1.519×10^{1}	+	6.038×10^{2}	1.313×10^{1}	+
F24	5.857×10^{2}	1.423×10^{1}	6.808×10^{2}	2.514×10^{1}	+	7.045×10^{2}	1.783×10^{1}	+	7.315×10^{2}	2.098×10^{1}	+
F25	5.192×10^{2}	3.821×10^{1}	5.311×10^{2}	1.655×10^{1}	=	5.417×10^{2}	1.116×10^{1}	+	5.539×10^{2}	9.237×10^{0}	+
F26	1.897×10^{3}	1.196×10^{2}	2.225×10^{3}	3.254×10^{2}	+	2.555×10^{3}	1.602×10^{2}	+	2.746×10^{3}	1.968×10^{2}	+
F27	5.771×10^{2}	1.741×10^{1}	5.874×10^{2}	2.774×10^{1}	+	6.055×10^{2}	1.489×10^{1}	+	6.142×10^{2}	2.016×10^{1}	+
F28	5.070×10^{2}	2.239×10^{0}	5.021×10^{2}	1.613×10^{1}	+	5.014×10^{2}	1.309×10^{1}	=	5.106×10^{2}	1.054×10^{1}	+
F29	4.960×10^{2}	7.496×10^{1}	6.543×10^{2}	1.264×10^{2}	+	6.314×10^{2}	8.857×10^{1}	+	7.129×10^{2}	9.568×10^{1}	+
F30	8.703×10^{5}	1.105×10^{5}	7.755×10^{5}	7.404×10^{5}	−	8.976×10^{5}	7.345×10^{5}	+	8.459×10^{5}	6.818×10^{5}	=
W/T/L	−/−/−		22/4/4			23/5/2			24/1/5		
Rank	1.53		2.23			2.77			3.47		
Dimension						**100**					
Fun	**DWSE**		**SE**			**LSE**			**CSE**		
	mean	**std**	**mean**	**std**		**mean**	**std**		**mean**	**std**	
F1	1.386×10^{3}	2.465×10^{3}	5.056×10^{3}	5.204×10^{3}	+	6.802×10^{3}	6.344×10^{3}	+	5.597×10^{3}	3.139×10^{3}	+
F2	1.000×10^{30}	1.421×10^{14}	1.000×10^{30}	2.843×10^{14}	=	1.000×10^{30}	1.421×10^{14}	=	1.000×10^{30}	1.421×10^{14}	=
F3	6.105×10^{5}	5.046×10^{5}	2.386×10^{5}	2.512×10^{5}	−	4.810×10^{5}	3.912×10^{5}	−	4.659×10^{5}	3.803×10^{5}	−
F4	2.305×10^{2}	2.780×10^{1}	2.346×10^{2}	2.039×10^{1}	=	2.341×10^{2}	1.719×10^{1}	=	2.590×10^{2}	2.268×10^{1}	+
F5	1.945×10^{2}	1.950×10^{1}	3.265×10^{2}	2.495×10^{1}	+	4.276×10^{2}	3.626×10^{1}	+	5.025×10^{2}	2.725×10^{1}	+
F6	1.398×10^{0}	4.804×10^{-1}	6.040×10^{-5}	2.399×10^{-4}	−	6.687×10^{-13}	4.342×10^{-14}	−	6.197×10^{-13}	6.556×10^{-14}	−
F7	3.525×10^{2}	2.983×10^{1}	4.592×10^{2}	3.011×10^{1}	+	5.630×10^{2}	3.066×10^{1}	+	6.408×10^{2}	2.109×10^{1}	+
F8	1.970×10^{2}	1.899×10^{1}	3.296×10^{2}	3.230×10^{1}	+	4.223×10^{2}	3.588×10^{1}	+	4.994×10^{2}	2.330×10^{1}	+
F9	2.902×10^{2}	2.578×10^{2}	7.582×10^{3}	1.691×10^{3}	+	8.399×10^{3}	2.005×10^{3}	+	1.115×10^{5}	2.118×10^{3}	+
F10	1.015×10^{5}	4.486×10^{2}	1.182×10^{5}	6.424×10^{2}	+	1.530×10^{5}	8.450×10^{2}	+	1.699×10^{5}	4.869×10^{2}	+
F11	5.027×10^{5}	1.448×10^{5}	4.288×10^{3}	1.886×10^{3}	−	1.915×10^{5}	4.900×10^{3}	−	2.438×10^{3}	7.156×10^{2}	−
F12	1.604×10^{6}	1.285×10^{6}	2.915×10^{7}	8.604×10^{6}	+	4.308×10^{7}	1.073×10^{7}	+	3.072×10^{7}	8.416×10^{6}	+
F13	3.248×10^{3}	2.960×10^{3}	3.671×10^{3}	2.807×10^{3}	=	1.211×10^{5}	5.742×10^{3}	+	3.241×10^{5}	1.199×10^{5}	+
F14	5.564×10^{6}	1.526×10^{6}	4.153×10^{6}	1.250×10^{6}	−	5.717×10^{6}	1.935×10^{6}	=	6.036×10^{5}	2.796×10^{5}	−
F15	6.978×10^{2}	6.967×10^{2}	1.450×10^{3}	9.566×10^{2}	+	5.603×10^{3}	2.670×10^{3}	+	9.098×10^{3}	4.256×10^{3}	+
F16	2.661×10^{3}	3.263×10^{2}	3.224×10^{3}	3.807×10^{2}	+	3.015×10^{3}	3.355×10^{2}	+	3.322×10^{3}	3.274×10^{2}	+
F17	2.059×10^{3}	2.154×10^{2}	2.140×10^{3}	3.034×10^{2}	=	2.171×10^{3}	2.246×10^{2}	+	2.364×10^{3}	2.647×10^{2}	+
F18	4.183×10^{6}	1.094×10^{6}	4.124×10^{6}	1.380×10^{6}	=	4.767×10^{6}	1.301×10^{6}	+	1.322×10^{6}	4.702×10^{5}	−
F19	9.863×10^{2}	1.149×10^{3}	2.516×10^{3}	1.624×10^{3}	+	1.568×10^{5}	6.910×10^{3}	+	1.811×10^{5}	8.100×10^{3}	+
F20	1.946×10^{3}	2.066×10^{2}	2.171×10^{3}	2.898×10^{2}	+	2.030×10^{3}	2.527×10^{2}	+	2.229×10^{3}	2.336×10^{2}	+
F21	4.445×10^{2}	2.347×10^{1}	5.747×10^{2}	2.702×10^{1}	+	6.585×10^{2}	3.247×10^{1}	+	7.260×10^{2}	2.767×10^{1}	+
F22	1.112×10^{5}	1.629×10^{3}	1.300×10^{5}	6.822×10^{2}	+	1.653×10^{5}	8.865×10^{2}	+	1.781×10^{5}	2.392×10^{3}	+
F23	6.474×10^{2}	1.381×10^{1}	7.010×10^{2}	1.903×10^{1}	+	7.287×10^{2}	1.973×10^{1}	+	7.590×10^{2}	1.703×10^{1}	+
F24	1.094×10^{3}	2.323×10^{1}	1.195×10^{3}	2.698×10^{1}	+	1.250×10^{3}	2.505×10^{1}	+	1.307×10^{3}	2.748×10^{1}	+
F25	7.755×10^{2}	5.605×10^{1}	7.588×10^{2}	4.400×10^{1}	−	8.239×10^{2}	2.015×10^{1}	+	8.490×10^{2}	2.085×10^{1}	+
F26	5.594×10^{3}	2.638×10^{2}	6.590×10^{3}	3.466×10^{2}	+	7.256×10^{3}	2.867×10^{2}	+	7.891×10^{3}	2.720×10^{2}	+
F27	7.098×10^{2}	1.558×10^{1}	7.285×10^{2}	2.278×10^{1}	+	7.524×10^{2}	2.004×10^{1}	+	7.601×10^{2}	2.112×10^{1}	+
F28	5.697×10^{2}	2.632×10^{1}	5.763×10^{2}	3.910×10^{1}	=	6.170×10^{2}	2.876×10^{1}	+	6.358×10^{2}	1.905×10^{1}	+
F29	2.120×10^{3}	2.505×10^{2}	2.662×10^{3}	2.788×10^{2}	+	2.494×10^{3}	2.282×10^{2}	+	2.792×10^{3}	2.145×10^{2}	+
F30	7.590×10^{3}	4.762×10^{3}	8.103×10^{3}	2.754×10^{3}	+	1.551×10^{5}	3.269×10^{3}	+	1.010×10^{5}	2.087×10^{3}	+
W/T/L	−/−/−		19/6/5			24/3/3			24/1/5		
Rank	1.50		2.17			2.97			3.37		

Figure 3 displays a box and convergence graph of four SEs on problem 1, problem 15, and problem 30 in CEC2017, where $dim = 30$. Problem 1 represents a single-peak unimodal function, wherein the local optimum coincides with the global optimum, demanding a greater focus on algorithmic exploitation rather than exploration. The figure clearly illustrates that DWSE not only exhibits superior local search capabilities but also demonstrates notably better stability when dealing with the single-peak problem, outperforming the other three algorithms.

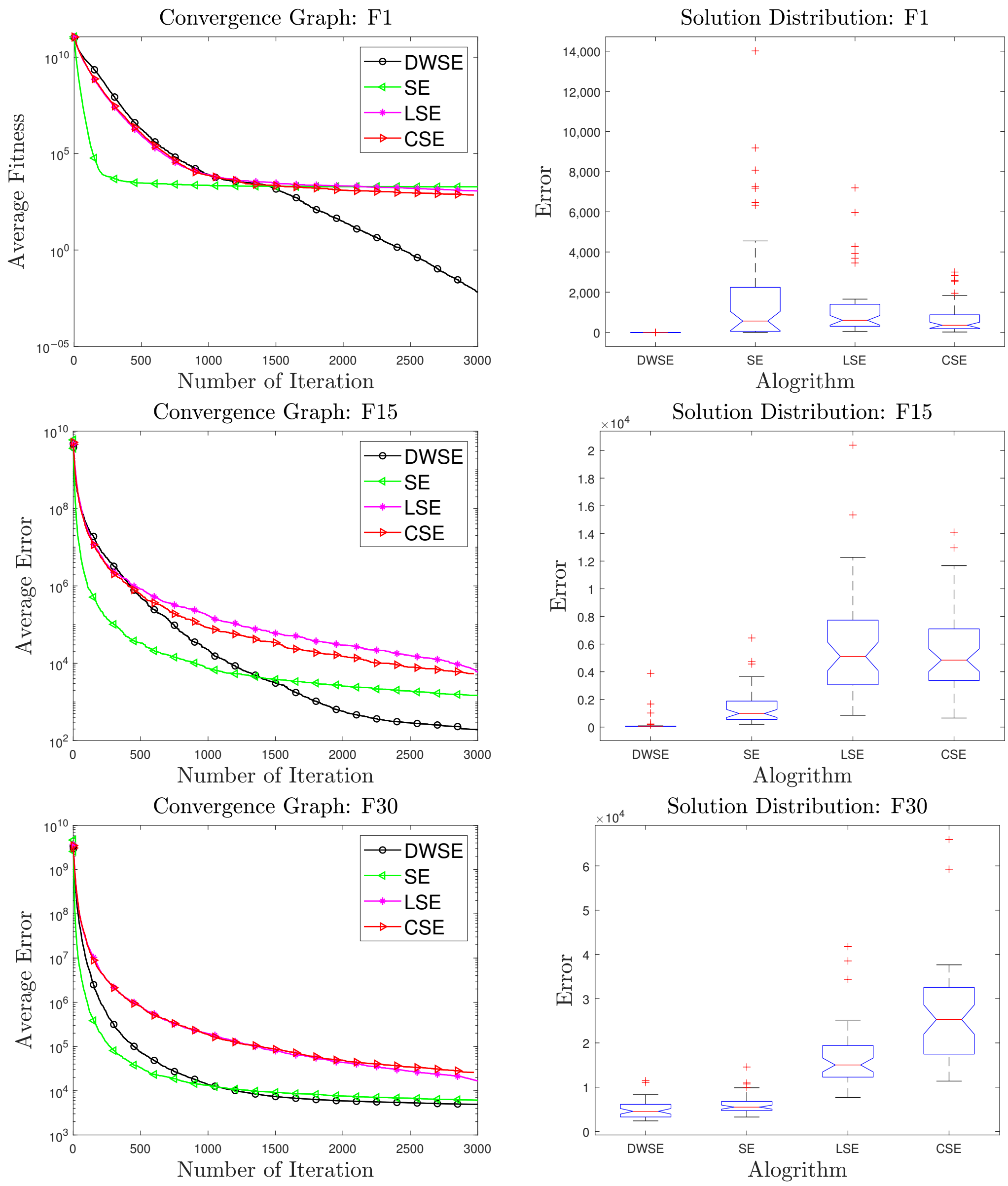

Figure 3. Box and convergence graph of SEs in CEC2017.

Problem 15 represents a hybrid function optimization problem with multiple peaks, indicating inherent unrelatedness among its dimensions. In the convergence graph, DWSE exhibits robust convergence, with a slight slowdown observed in the later stages when focusing on local search, while the other SEs show significantly slower convergence. As evidenced by the box plots, DWSE maintains similar stability performance as in problem 1 and significantly outperforms the other algorithms. The red + represent the location of the extremes in the repeat run. The presence of individual extreme values indicates a tendency to converge to a local optimum, suggesting that DWSE may have the capability to find a globally optimal solution for this problem when extreme values are not involved.

In contrast, problem 30 is a multipeak composition function problem, requiring a higher level of exploratory capability than problem 1. Even in this scenario, DWSE emerges as the top performer, as evidenced by the convergence chart. Notably, CSE, which employs chaotic differential operators, fares the worst on this problem, while LSE exhibits weaker exploration capabilities compared to SE, mainly due to its emphasis on strengthening exploitation without an increase in the initial population size ($18 \times dim$ in L-SHADE).

Considering the combined analysis of both problems, it is evident that DWSE not only enhances the local search ability of SE but also improves its capacity to explore global optima. Furthermore, the convergence process of DWSE is smooth and stable, contributing to its impressive performance.

3.2. Experimental Results and Analysis in CEC2011

Table 3 shows the experimental results in CEC2011, a test set commonly used to test the ability of algorithms to solve real-world problems. Since the global optimal solution of a real-world problem may not be a knowable value, we use the mean of the final fitness for statistics on that problem. Among these 22 real-world problems for solving the minimum, DWSE is still substantially ahead of the other algorithms.

Table 3. Experimental results of 22 real-world problems in CEC2011.

Fun	DWSE		SE			LSE			CSE		
	Mean	**std**	**Mean**	**std**		**Mean**	**std**		**Mean**	**std**	
F1	5.843×10^{0}	2.953×10^{0}	1.031×10^{1}	3.667×10^{0}	+	9.370×10^{0}	3.447×10^{0}	+	1.065×10^{1}	3.129×10^{0}	+
F2	-2.616×10^{1}	1.277×10^{0}	-2.061×10^{1}	1.015×10^{0}	+	-2.119×10^{1}	9.709×10^{-1}	+	-2.039×10^{1}	9.685×10^{-1}	+
F3	1.151×10^{-5}	3.277×10^{-19}	1.151×10^{-5}	2.673×10^{-19}	−	1.151×10^{-5}	2.331×10^{-19}	−	1.151×10^{-5}	2.600×10^{-19}	−
F4	1.632×10^{1}	3.002×10^{0}	1.632×10^{1}	3.098×10^{0}	=	1.591×10^{1}	2.988×10^{0}	−	1.600×10^{1}	2.942×10^{0}	−
F5	-3.669×10^{1}	2.448×10^{-1}	-3.388×10^{1}	4.240×10^{-1}	+	-3.419×10^{1}	5.927×10^{-1}	+	-3.388×10^{1}	3.930×10^{-1}	+
F6	-2.913×10^{1}	2.430×10^{-1}	-2.812×10^{1}	5.762×10^{-1}	+	-2.845×10^{1}	4.703×10^{-1}	+	-2.805×10^{1}	5.357×10^{-1}	+
F7	1.205×10^{0}	9.278×10^{-2}	1.325×10^{0}	7.553×10^{-2}	+	1.298×10^{0}	1.015×10^{-1}	+	1.336×10^{0}	9.118×10^{-2}	+
F8	2.200×10^{2}	0.000×10^{0}	2.200×10^{2}	0.000×10^{0}	=	2.200×10^{2}	0.000×10^{0}	=	2.200×10^{2}	0.000×10^{0}	=
F9	1.351×10^{3}	5.020×10^{2}	2.767×10^{3}	3.369×10^{2}	+	2.376×10^{3}	3.406×10^{2}	+	2.872×10^{3}	3.665×10^{2}	+
F10	-1.915×10^{1}	1.434×10^{0}	-1.768×10^{1}	8.702×10^{-1}	+	-1.842×10^{1}	9.357×10^{-1}	+	-1.769×10^{1}	9.081×10^{-1}	+
F11	5.164×10^{5}	5.421×10^{2}	5.807×10^{5}	7.384×10^{2}	+	5.657×10^{5}	7.366×10^{2}	+	5.819×10^{5}	7.997×10^{2}	+
F12	1.768×10^{7}	1.051×10^{5}	1.732×10^{7}	4.691×10^{3}	−	1.732×10^{7}	4.902×10^{3}	−	1.732×10^{7}	2.350×10^{3}	−
F13	1.545×10^{5}	2.440×10^{0}	1.546×10^{5}	4.942×10^{0}	+	1.546×10^{5}	6.291×10^{0}	+	1.546×10^{5}	4.940×10^{0}	+
F14	1.906×10^{5}	1.084×10^{2}	1.933×10^{5}	1.855×10^{2}	+	1.933×10^{5}	1.503×10^{2}	+	1.936×10^{5}	1.808×10^{2}	+
F15	3.294×10^{5}	2.627×10^{1}	3.297×10^{5}	2.421×10^{1}	+	3.297×10^{5}	2.789×10^{1}	+	3.296×10^{5}	2.711×10^{1}	+
F16	1.339×10^{5}	1.174×10^{3}	1.348×10^{5}	1.765×10^{3}	+	1.350×10^{5}	1.280×10^{3}	+	1.349×10^{5}	1.944×10^{3}	+
F17	1.916×10^{6}	1.431×10^{5}	1.937×10^{6}	1.401×10^{5}	+	1.934×10^{6}	1.484×10^{5}	+	1.934×10^{6}	1.492×10^{5}	+
F18	9.386×10^{5}	1.486×10^{3}	9.575×10^{5}	6.445×10^{3}	+	9.542×10^{5}	5.250×10^{3}	+	9.570×10^{5}	5.591×10^{3}	+
F19	1.181×10^{6}	7.244×10^{5}	1.207×10^{6}	6.304×10^{5}	+	1.186×10^{6}	7.117×10^{5}	=	1.206×10^{6}	6.742×10^{5}	+
F20	9.379×10^{5}	1.687×10^{3}	9.570×10^{5}	6.184×10^{3}	+	9.548×10^{5}	4.625×10^{3}	+	9.561×10^{5}	5.929×10^{3}	+
F21	1.561×10^{1}	1.748×10^{0}	1.712×10^{1}	1.802×10^{0}	+	1.688×10^{1}	2.091×10^{0}	+	1.696×10^{1}	1.986×10^{0}	+
F22	1.660×10^{1}	2.335×10^{0}	1.883×10^{1}	2.241×10^{0}	+	1.841×10^{1}	2.069×10^{0}	+	1.907×10^{1}	1.978×10^{0}	+
W/T/L	−/−/−		18/2/2			17/2/3			18/1/3		
Rank	1.48		3.16			2.16			3.20		

Figure 4 shows the box and convergence graph of SEs in CEC2011. The red + represent the location of the extremes in the repeat run. Problem 1 is a parameter estimation for a frequency-modulated sound wave problem; DWSE shows strong convergence in this problem. The algorithm converges progressively faster on simple single-peak-type problems due to the adaptive weights, gaining an advantage over other algorithms. The DWSE continues to be the most consistent in the box plot, with two extremes, but the values in its middle segment are much closer together. Problem 2 is a spacecraft trajectory optimization problem; DWSE continues to outperform other algorithms on this problem. However, the stability of DWSE decreases slightly for relatively complex problems, so repeated optimizations may become increasingly necessary. The advantages of DWSE in terms of the speed of convergence of this problem is still evident.

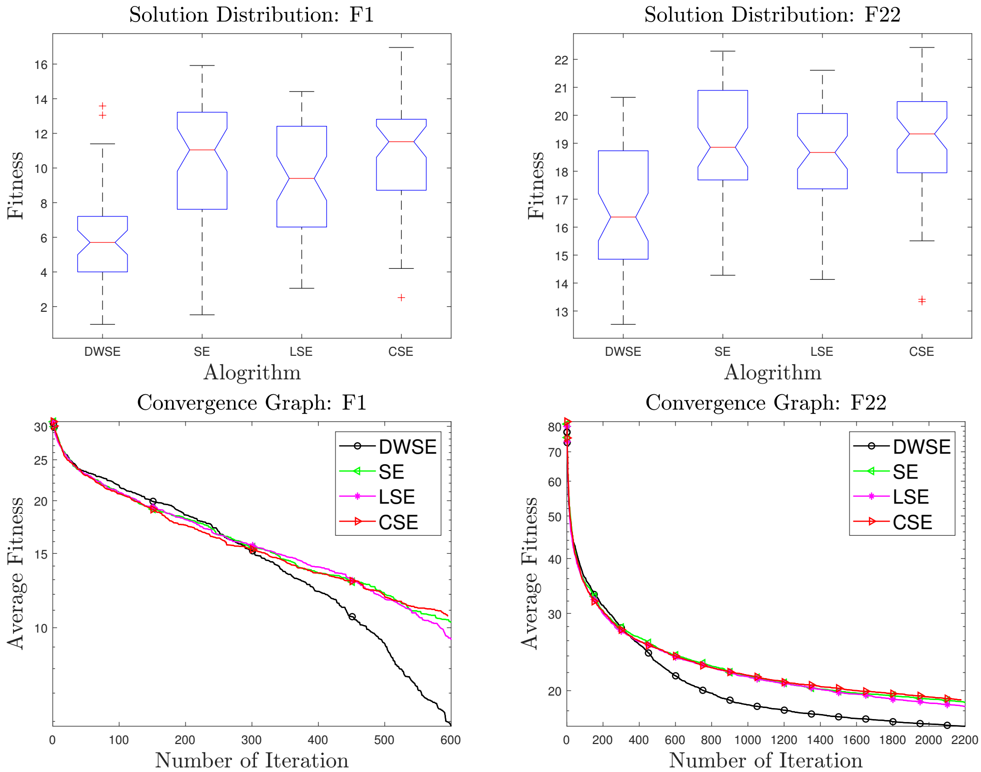

Figure 4. Box and convergence graph of SEs in CEC2011.

3.3. Experimental Results and Analysis in CEC2013 LSGO

Previous experimental results and analyses show that DWSE is much stronger than other SEs on low-dimensional problems, but its performance decreases as the problem dimensions increase. Therefore, in this paper, we adopted the large-scale global optimization problem set of CEC2013 as a test of the performance of DWSE under ultra-high dimensional problems.

Table 4 shows the experimental results of large-scale global optimization in CEC2013. However, in terms of results, DWSE is only better than SE and equal to LSE and CSE in terms of wins and losses. However, considering the performance decay of DWSE with increased dimensions, while it can still achieve a performance not weaker than that of other SEs on problems with dimensions of 1000 or less, the improvement of DWSE can be considered a success. Large-scale global optimization problems have consistently posed significant challenges for evolutionary algorithms, and often, only algorithms designed for large-scale problems can effectively optimize such cases. We utilized these problems to test whether DWSE would be less effective than other SEs when confronted with performance decay arising from increased problem dimensionality. However, the experimental results were satisfactory, demonstrating DWSE's capability to adeptly handle such scenarios.

Table 4. Experimental results of large-scale global optimization in CEC2013.

Dimension	1000							
Fun	DWSE		SE		LSE		CSE	
	Mean	std	Mean	std	Mean	std	Mean	std
F1	2.487×10^{7}	5.540×10^{6}	2.584×10^{5}	1.733×10^{3}	2.678×10^{3}	1.596×10^{2}	1.844×10^{5}	3.365×10^{3}
F2	6.953×10^{3}	3.796×10^{2}	4.620×10^{-1}	2.810×10^{-1}	1.135×10^{0}	9.313×10^{-1}	3.787×10^{-1}	2.601×10^{-1}
F3	1.524×10^{1}	9.607×10^{-1}	1.113×10^{-5}	6.815×10^{-7}	6.779×10^{-7}	9.492×10^{-8}	8.754×10^{-6}	9.384×10^{-7}
F4	5.267×10^{10}	1.350×10^{11}	3.812×10^{11}	8.957×10^{10}	3.304×10^{11}	9.063×10^{10}	2.227×10^{10}	7.358×10^{9}
F5	4.035×10^{6}	3.780×10^{5}	1.068×10^{7}	7.643×10^{5}	1.010×10^{7}	8.718×10^{5}	9.721×10^{6}	1.020×10^{6}
F6	4.796×10^{5}	4.253×10^{5}	1.016×10^{6}	2.706×10^{3}	1.004×10^{6}	2.923×10^{3}	1.014×10^{6}	7.361×10^{3}
F7	2.498×10^{9}	1.134×10^{9}	1.653×10^{9}	2.979×10^{8}	1.726×10^{9}	3.136×10^{8}	1.678×10^{9}	2.760×10^{8}
F8	9.051×10^{14}	2.977×10^{15}	1.689×10^{16}	4.084×10^{15}	1.429×10^{16}	4.062×10^{15}	3.894×10^{14}	1.791×10^{14}
F9	2.147×10^{8}	3.247×10^{7}	7.994×10^{8}	5.636×10^{7}	7.470×10^{8}	7.674×10^{7}	7.060×10^{8}	9.144×10^{7}
F10	1.822×10^{7}	4.272×10^{6}	1.824×10^{7}	3.103×10^{6}	1.508×10^{7}	4.458×10^{6}	1.432×10^{7}	3.205×10^{6}
F11	1.979×10^{11}	3.061×10^{11}	1.612×10^{11}	5.442×10^{10}	1.520×10^{11}	4.154×10^{10}	1.256×10^{11}	4.443×10^{10}
F12	2.453×10^{6}	9.784×10^{5}	5.024×10^{3}	1.587×10^{2}	4.076×10^{3}	1.482×10^{2}	4.819×10^{3}	1.586×10^{2}
F13	2.973×10^{10}	6.118×10^{9}	2.258×10^{10}	2.142×10^{9}	2.060×10^{10}	2.406×10^{9}	2.205×10^{10}	2.943×10^{9}
F14	3.340×10^{11}	8.037×10^{10}	3.300×10^{11}	5.522×10^{10}	3.184×10^{11}	5.341×10^{10}	3.046×10^{11}	6.401×10^{10}
F15	2.600×10^{7}	2.600×10^{6}	1.459×10^{8}	2.647×10^{7}	1.298×10^{8}	2.205×10^{7}	1.198×10^{8}	1.890×10^{7}
W/T/L	−/−/−		7/2/6		7/1/7		7/1/7	

Figure 5 shows the box and convergence graph of SEs in CEC2013 LSGO. The red + represent the location of the extremes in the repeat run. The search efficiency of DWSE on problem 1 is much lower than that of other SEs, which may be due to the fact that the adaptive range of the weights is so small in the ultra-high dimensionality problem that the number of dimensions being changed each time is too large. Changing too many dimensions on that problem may lead to slower convergence and therefore insufficient *val* values, eventually leading to insufficient change of weights comprising a vicious circle. On problem 15, at the beginning of convergence, DWSE is still inferior to the other algorithms, again due to the weighting calculation. However, since all algorithms fall into a close local optimum on this problem, while the other SEs lack local search capability and stagnate altogether, DWSE's excellent local search capability allows it to obtain the most accurate set of solutions on the local optimum, defeating the other SEs.

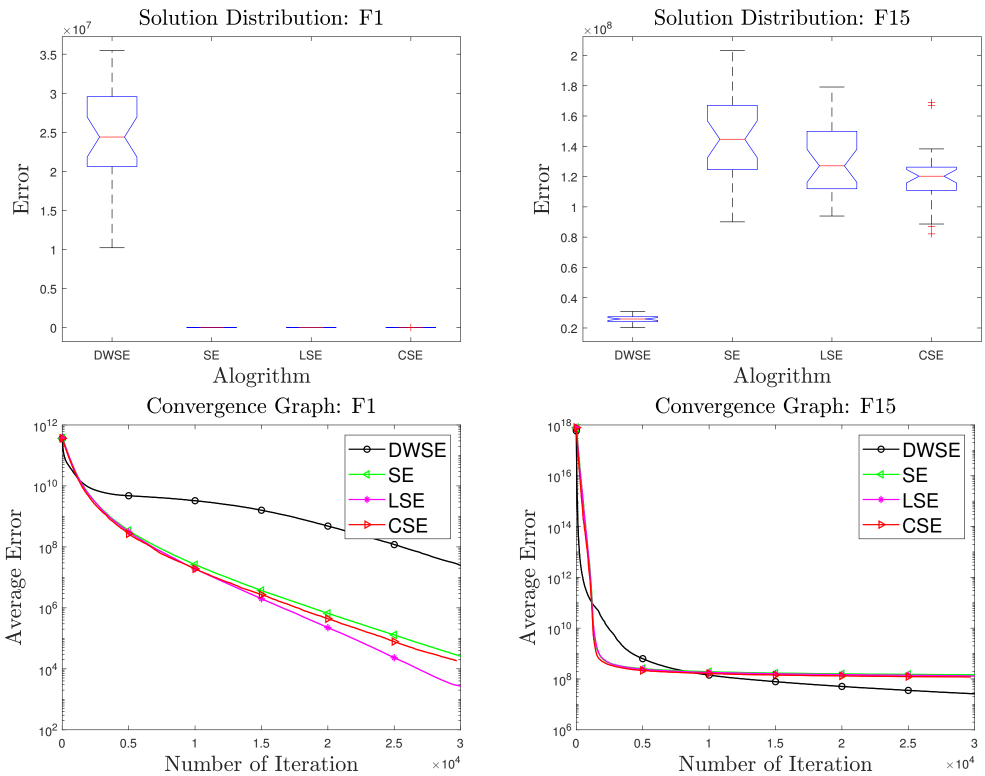

Figure 5. Box and convergence graph of SEs in CEC2013 LSGO.

3.4. Comparison with Other MHAs in CEC2017 and CEC2011

In order to further demonstrate the improvement in the performance of DWSE, it was compared with a selection of classic, strong, and recent algorithms on a low to medium dimensional test set [41–46]. Table 5 shows the comparison results of DWSE with other MHAs in CEC2017 and CEC2011. DWSE took the lead in the overall comparison results and gained the lead in every set of problems, except for CEC2017, where it was slightly behind DE under the 30-dimension problem. We believe this is a testament to DWSE's strength in performance. It should be noted that IPA, due to its design mechanism, cannot be run on problems with a low dimension count; therefore its data are not available for CEC2011.

Table 5. Comparison results of DWSE with other MHAs in CEC2017 and CEC2011.

	DWSE VS.	**DE**	**GLPSO**	**DNLGSA**	**GWO**	**IPA**	**DEPSO**	**CSO**
CEC2017	$dim = 30$	13/1/16	28/1/1	28/1/1	29/0/1	24/3/3	27/1/2	26/1/3
	$dim = 50$	18/2/10	27/1/2	27/2/1	29/0/1	22/2/6	27/2/1	25/2/3
	$dim = 100$	18/4/8	24/1/5	25/1/4	26/0/4	25/2/3	26/2/2	27/2/1
CEC2011		13/3/6	19/3/0	20/1/1	19/1/2	−/−/−	17/4/1	20/2/0
	total	62/10/40	98/6/8	100/5/7	103/1/8	71/7/12	97/9/6	98/7/7

3.5. Discussion

In the time complexity analysis, the initialization complexity of DWSE is $O(popusize)$. The upper bound of the DSF operation is dependent on the problem dimension, resulting in a time complexity of $O(dim)$ for the computation of DSF and W. The operator utilized in DWSE consumes a time complexity of $O(dim)$. Additionally, the time complexity associated with the dimension selection process is $O(\log(dim))$. In summary, the overall time complexity of DWSE can be expressed as $O(popusize \cdot dim \cdot dim \cdot \log(dim))$. In comparison to SE, the complexity of DWSE is higher. This higher complexity is attributed to DWSE's requirement of dimension weighting, which increases its complexity by a factor of dim due to the additional operations conducted in the loop involving $popusize \cdot dim$ computations. However, considering the performance enhancement achieved by DWSE, the slight increase in computation time is deemed acceptable.

Table 6 shows the experimental results of all SE operators in dimension 30 in CEC2017. Among these, DWSE utilizes SE/rand/1 and SE/rand/2, which exhibit nearly identical performance, displaying negligible distinctions, with notable proximity to SE/current/1 and SE/current/2. Given that SE/rand/1 is the recommended operator within the SE framework and is widely employed for real-world problem optimization, we concur with its selection.

Table 6. Experimental results of operator discussion in CEC2017.

Fun	DWSE (Rand/1) VS.	Current-to-Best/1	Best/1	Best/2	Rand/2	Current/1	Current/2
	Mean	Mean	Mean	Mean	Mean	Mean	Mean
F1	6.474×10^{-3}	2.953×10^{-1}	1.404×10^{-3}	2.754×10^{-3}	6.613×10^{-3}	7.918×10^{-1}	1.735×10^{0}
F2	3.535×10^{12}	1.486×10^{1}	1.723×10^{-4}	2.179×10^{-6}	2.208×10^{12}	4.359×10^{2}	4.695×10^{1}
F3	1.122×10^{5}	1.023×10^{5}	1.069×10^{5}	1.112×10^{5}	7.933×10^{5}	1.207×10^{5}	8.139×10^{5}
F4	9.259×10^{1}	6.834×10^{1}	8.338×10^{1}	6.888×10^{1}	9.417×10^{1}	8.355×10^{1}	9.880×10^{1}
F5	3.000×10^{1}	5.492×10^{1}	7.263×10^{1}	8.557×10^{1}	3.673×10^{1}	6.729×10^{1}	7.603×10^{1}
F6	1.137×10^{-13}	2.841×10^{-3}	6.338×10^{0}	1.049×10^{1}	1.137×10^{-13}	3.833×10^{-3}	2.982×10^{-3}
F7	6.352×10^{1}	1.065×10^{2}	2.502×10^{2}	1.438×10^{2}	5.980×10^{1}	8.694×10^{1}	9.616×10^{1}
F8	3.588×10^{1}	7.401×10^{1}	5.572×10^{1}	8.855×10^{1}	3.809×10^{1}	5.557×10^{1}	6.127×10^{1}
F9	6.001×10^{-1}	3.373×10^{2}	2.220×10^{3}	3.466×10^{2}	0.000×10^{0}	7.615×10^{2}	4.045×10^{2}
F10	1.880×10^{3}	2.434×10^{3}	2.523×10^{3}	1.776×10^{3}	1.852×10^{3}	2.124×10^{3}	1.769×10^{3}
F11	2.561×10^{1}	3.323×10^{1}	1.142×10^{2}	1.406×10^{2}	2.137×10^{1}	3.742×10^{1}	2.063×10^{1}
F12	4.119×10^{5}	6.249×10^{5}	8.152×10^{5}	5.854×10^{5}	1.233×10^{5}	3.953×10^{5}	4.153×10^{5}
F13	7.504×10^{3}	6.463×10^{2}	5.960×10^{5}	2.959×10^{5}	8.429×10^{3}	4.945×10^{2}	6.064×10^{2}
F14	1.200×10^{3}	7.186×10^{1}	7.406×10^{2}	7.455×10^{1}	7.491×10^{1}	7.148×10^{1}	8.118×10^{1}
F15	1.923×10^{2}	7.142×10^{1}	2.915×10^{5}	6.659×10^{1}	1.024×10^{2}	6.731×10^{1}	1.091×10^{2}
F16	3.989×10^{2}	6.332×10^{2}	1.195×10^{3}	1.142×10^{3}	2.001×10^{2}	5.178×10^{2}	4.305×10^{2}
F17	7.037×10^{1}	8.409×10^{1}	6.279×10^{2}	8.532×10^{1}	7.794×10^{1}	6.963×10^{1}	1.576×10^{2}
F18	2.120×10^{5}	7.492×10^{5}	5.255×10^{5}	1.155×10^{5}	6.630×10^{5}	1.180×10^{5}	9.448×10^{3}
F19	5.133×10^{2}	3.663×10^{1}	1.795×10^{2}	6.031×10^{3}	4.879×10^{2}	3.674×10^{1}	4.695×10^{1}
F20	1.113×10^{2}	2.050×10^{2}	3.010×10^{1}	1.608×10^{2}	5.790×10^{1}	8.585×10^{1}	7.037×10^{1}
F21	2.359×10^{2}	2.670×10^{2}	2.945×10^{2}	2.267×10^{2}	2.364×10^{2}	1.352×10^{2}	2.641×10^{2}
F22	1.023×10^{2}	1.058×10^{2}	1.038×10^{2}	1.000×10^{2}	1.169×10^{2}	1.108×10^{2}	1.000×10^{2}
F23	3.828×10^{2}	3.844×10^{2}	4.164×10^{2}	4.437×10^{2}	3.877×10^{2}	3.819×10^{2}	3.808×10^{2}
F24	4.557×10^{2}	5.308×10^{2}	4.994×10^{2}	4.812×10^{2}	4.564×10^{2}	2.106×10^{2}	5.319×10^{2}
F25	3.874×10^{2}	3.870×10^{2}	4.294×10^{2}	3.894×10^{2}	3.871×10^{2}	3.869×10^{2}	3.869×10^{2}
F26	1.312×10^{3}	3.077×10^{2}	1.703×10^{3}	1.574×10^{3}	1.366×10^{3}	2.458×10^{2}	2.461×10^{2}
F27	5.091×10^{2}	5.111×10^{2}	5.022×10^{2}	5.246×10^{2}	5.105×10^{2}	5.182×10^{2}	5.104×10^{2}
F28	3.299×10^{2}	3.796×10^{2}	4.401×10^{2}	4.067×10^{2}	4.070×10^{2}	3.000×10^{2}	3.030×10^{2}
F29	4.968×10^{2}	5.091×10^{2}	1.138×10^{3}	9.890×10^{2}	5.034×10^{2}	5.666×10^{2}	6.217×10^{2}
F30	4.934×10^{3}	9.960×10^{3}	3.846×10^{3}	9.506×10^{3}	6.082×10^{3}	8.189×10^{3}	3.797×10^{3}
W/T/L	−/−/−	17/4/9	20/2/8	19/3/8	10/10/10	14/3/13	13/5/12

4. Conclusions

The development of optimization algorithms has a long and rich history, which has given rise to a series of intriguing and diverse algorithms in its early stages. As problems and algorithms have progressed, driven by the pursuit of better performance, algorithmic improvements have gradually converged, resulting in a phenomenon of becoming trapped in local optima. Although the developers of SE were the first to observe and propose this unique algorithm, it has also presented challenges in implementing many successful improvement experiences in this context.

The improvement method proposed in this paper involves a specialized operation based on the dimension selection of SE. Despite its non-complex nature, the effectiveness of this improvement is evident. The substantial enhancements in low- and medium-dimensional problems, coupled with the sustained performance in ultra-high-dimensional problems, bring SEs closer to approximating the global optimum. As one of the few SE improvement algorithms, we also hope that DWSE can contribute to the diversification of algorithms.

SE is an algorithm commonly employed to optimize parameter estimation problems for photovoltaic models and has proven its effectiveness in solving such problems. Our next step involves selecting this problem and utilizing the current best-performing SE, i.e., DWSE, as the foundation for further enhancements, aiming to attain even better results for real-world applications.

Author Contributions: Conceptualization, Y.Y. and S.T.; methodology, Y.Y.; software, Y.Y. and S.T.; validation, Y.Y.; formal analysis, Y.Y. and S.T.; investigation, Y.Y. and S.D.; resources, Z.T.; data curation, S.D.; M.N.; writing—original draft preparation, Y.Y.; writing—review and editing, Y.Y. and Z.T.; visualization, S.T.; supervision, Z.T.; project administration, Z.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data can be made available upon request by contacting the corresponding author's email address. The source code is publicly available at https://github.com/louiseklocky (accessed on 15 August 2023).

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

References

1. Abualigah, L.; Elaziz, M.A.; Khasawneh, A.M.; Alshinwan, M.; Ibrahim, R.A.; Al-Qaness, M.A.; Mirjalili, S.; Sumari, P.; Gandomi, A.H. Meta-heuristic optimization algorithms for solving real-world mechanical engineering design problems: A comprehensive survey, applications, comparative analysis, and results. *Neural Comput. Appl.* **2022**, *34*, 4081–4110. [CrossRef]
2. Desale, S.; Rasool, A.; Andhale, S.; Rane, P. Heuristic and meta-heuristic algorithms and their relevance to the real world: A survey. *Int. J. Comput. Eng. Res. Trends* **2015**, *351*, 2349–7084.
3. Katoch, S.; Chauhan, S.S.; Kumar, V. A review on genetic algorithm: past, present, and future. *Multimed. Tools Appl.* **2021**, *80*, 8091–8126. [CrossRef] [PubMed]
4. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
5. Rutenbar, R.A. Simulated annealing algorithms: An overview. *IEEE Circuits Devices Mag.* **1989**, *5*, 19–26. [CrossRef]
6. Das, S.; Suganthan, P.N. Differential Evolution: A Survey of the State-of-the-Art. *IEEE Trans. Evol. Comput.* **2011**, *15*, 4–31. [CrossRef]
7. Storn, R.; Price, K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
8. Bansal, J.C.; Sharma, H.; Jadon, S.S. Artificial bee colony algorithm: A survey. *Int. J. Adv. Intell. Paradig.* **2013**, *5*, 123–159. [CrossRef]
9. Dorigo, M.; Birattari, M.; Stutzle, T. Ant colony optimization. *IEEE Comput. Intell. Mag.* **2006**, *1*, 28–39. [CrossRef]
10. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
11. Mirjalili, S. SCA: A sine cosine algorithm for solving optimization problems. *Knowl.-Based Syst.* **2016**, *96*, 120–133. [CrossRef]
12. Hansen, N.; Müller, S.D.; Koumoutsakos, P. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evol. Comput.* **2003**, *11*, 106365603321828970. [CrossRef] [PubMed]
13. Kramer, O. Evolutionary self-adaptation: A survey of operators and strategy parameters. *Evol. Intell.* **2010**, *3*, 51–65. [CrossRef]
14. Umbarkar, A.J.; Sheth, P.D. Crossover operators in genetic algorithms: A review. *ICTACT J. Soft Comput.* **2015**, *6*, 34–36.
15. Ramos-Figueroa, O.; Quiroz-Castellanos, M.; Mezura-Montes, E.; Kharel, R. Variation operators for grouping genetic algorithms: A review. *Swarm Evol. Comput.* **2021**, *60*, 100796. [CrossRef]
16. Zhang, J.; Sanderson, A.C. JADE: Adaptive Differential Evolution with Optional External Archive. *IEEE Trans. Evol. Comput.* **2009**, *13*, 945–958. [CrossRef]
17. Tanabe, R.; Fukunaga, A. Success-history based parameter adaptation for Differential Evolution. In Proceedings of the 2013 IEEE Congress on Evolutionary Computation, Cancun, Mexico, 20–23 June 2013; pp. 71–78. [CrossRef]
18. Yang, H.; Yu, Y.; Cheng, J.; Lei, Z.; Cai, Z.; Zhang, Z.; Gao, S. An intelligent metaphor-free spatial information sampling algorithm for balancing exploitation and exploration. *Knowl.-Based Syst.* **2022**, *250*, 109081. [CrossRef]
19. Chakraborty, A.; Kar, A.K. Swarm intelligence: A review of algorithms. In *Nature-Inspired Computing and Optimization: Theory and Applications*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 475–494.
20. Schmitt, L.M. Theory of genetic algorithms. *Theor. Comput. Sci.* **2001**, *259*, 69–85. [CrossRef]
21. Bang-Jensen, J.; Gutin, G.; Yeo, A. When the greedy algorithm fails. *Discret. Optim.* **2004**, *1*, 121–127. [CrossRef]
22. Črepinšek, M.; Liu, S.H.; Mernik, M. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Comput. Surv.* **2013**, *45*, 1–33. [CrossRef]
23. Yu, Y.; Gao, S.; Wang, Y.; Todo, Y. Global optimum-based search differential evolution. *IEEE/CAA J. Autom. Sin.* **2019**, *6*, 379–394. [CrossRef]
24. Gershman, S.J. Uncertainty and exploration. *Decision* **2019**, *6*, 277. [CrossRef] [PubMed]
25. Li, W.; Ding, Y.; Yang, Y.; Sherratt, R.S.; Park, J.H.; Wang, J. Parameterized algorithms of fundamental NP-hard problems: A survey. *Hum.-Centric Comput. Inf. Sci.* **2020**, *10*, 29. [CrossRef]
26. Gao, S.; Yu, Y.; Wang, Y.; Wang, J.; Cheng, J.; Zhou, M. Chaotic Local Search-Based Differential Evolution Algorithms for Optimization. *IEEE Trans. Syst. Man Cybern. Syst.* **2021**, *51*, 3954–3967. [CrossRef]
27. Xu, Z.; Gao, S.; Yang, H.; Lei, Z. SCJADE: Yet Another State-of-the-Art Differential Evolution Algorithm. *IEEJ Trans. Electr. Electron. Eng.* **2021**, *16*, 644–646. [CrossRef]
28. Zheng, Y.J.; Zhang, B. A simplified water wave optimization algorithm. In Proceedings of the 2015 IEEE Congress on Evolutionary Computation (CEC), Sendai, Japan, 25–28 May 2015; pp. 807–813.
29. Li, H.; Zhang, B.; Li, J.; Zheng, T.; Yang, H. Using sparrow search hunting mechanism to improve water wave algorithm. In Proceedings of the 2021 IEEE International Conference on Progress in Informatics and Computing (PIC), Shanghai, China, 17–19 December 2021; pp. 19–23.
30. Tang, D. Spherical evolution for solving continuous optimization problems. *Appl. Soft Comput.* **2019**, *81*, 105499. [CrossRef]

31. Weng, X.; Heidari, A.A.; Liang, G.; Chen, H.; Ma, X.; Mafarja, M.; Turabieh, H. Laplacian Nelder-Mead spherical evolution for parameter estimation of photovoltaic models. *Energy Convers. Manag.* **2021**, *243*, 114223. [CrossRef]
32. Zhou, W.; Wang, P.; Heidari, A.A.; Zhao, X.; Turabieh, H.; Mafarja, M.; Chen, H. Metaphor-free dynamic spherical evolution for parameter estimation of photovoltaic modules. *Energy Rep.* **2021**, *7*, 5175–5202. [CrossRef]
33. Yang, H.; Gao, S.; Lei, Z.; Li, J.; Yu, Y.; Wang, Y. An improved spherical evolution with enhanced exploration capabilities to address wind farm layout optimization problem. *Eng. Appl. Artif. Intell.* **2023**, *123*, 106198. [CrossRef]
34. Zhao, J.; Zhang, B.; Guo, X.; Qi, L.; Li, Z. Self-Adapting Spherical Search Algorithm with Differential Evolution for Global Optimization. *Mathematics* **2022**, *10*, 4519. [CrossRef]
35. Yang, H.; Gao, S.; Wang, R.L.; Todo, Y. A ladder spherical evolution search algorithm. *IEICE Trans. Inf. Syst.* **2021**, *104*, 461–464. [CrossRef]
36. Tanabe, R.; Fukunaga, A.S. Improving the search performance of SHADE using linear population size reduction. In Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, China, 6–11 July 2014; pp. 1658–1665. [CrossRef]
37. Yang, J.; Zhang, Y.; Wang, Z.; Todo, Y.; Lu, B.; Gao, S. A cooperative coevolution wingsuit flying search algorithm with spherical evolution. *Int. J. Comput. Intell. Syst.* **2021**, *14*, 178. [CrossRef]
38. Yang, L.; Gao, S.; Yang, H.; Cai, Z.; Lei, Z.; Todo, Y. Adaptive chaotic spherical evolution algorithm. *Memetic Comput.* **2021**, *13*, 383–411. [CrossRef]
39. Zhou, H.; Cheung, W.; Leung, L.C. Minimizing weighted tardiness of job-shop scheduling using a hybrid genetic algorithm. *Eur. J. Oper. Res.* **2009**, *194*, 637–649. [CrossRef]
40. Bai, L.; Liang, J.; Dang, C.; Cao, F. A novel attribute weighting algorithm for clustering high-dimensional categorical data. *Pattern Recognit.* **2011**, *44*, 2843–2861. [CrossRef]
41. Gong, Y.J.; Li, J.J.; Zhou, Y.; Li, Y.; Chung, H.S.H.; Shi, Y.H.; Zhang, J. Genetic Learning Particle Swarm Optimization. *IEEE Trans. Cybern.* **2016**, *46*, 2277–2290. [CrossRef] [PubMed]
42. Zhang, A.; Sun, G.; Ren, J.; Li, X.; Wang, Z.; Jia, X. A Dynamic Neighborhood Learning-Based Gravitational Search Algorithm. *IEEE Trans. Cybern.* **2018**, *48*, 436–447. [CrossRef]
43. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
44. Aslan, S.; Demirci, S. Immune Plasma Algorithm: A Novel Meta-Heuristic for Optimization Problems. *IEEE Access* **2020**, *8*, 220227–220245. [CrossRef]
45. Zhang, W.J.; Xie, X.F. DEPSO: hybrid particle swarm with differential evolution operator. In Proceedings of the SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme—System Security and Assurance (Cat. No.03CH37483), Washington, DC, USA, 8 October 2003; Volume 4, pp. 3816–3821. [CrossRef]
46. Yang, X.S.; Deb, S. Cuckoo Search via Lévy flights. In Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore, India, 9–11 December 2009; pp. 210–214. [CrossRef]