Hindawi
Scientific Programming
Volume 2019, Article ID 7151574, 15 pages
https://doi.org/10.1155/2019/7151574

Research Article

An Efficient Hybrid Optimization Approach Using Adaptive Elitist Differential Evolution and Spherical Quadratic Steepest Descent and Its Application for Clustering

T. Nguyen-Trang,[1,2] **T. Nguyen-Thoi**,[1,3] **T. Truong-Khac**,[4] **A. T. Pham-Chau**,[1,2] **and HungLinh Ao**[5]

[1]*Division of Computational Mathematics and Engineering, Institute for Computational Science, Ton Duc Thang University, Ho Chi Minh City, Vietnam*
[2]*Faculty of Mathematics and Statistics, Ton Duc Thang University, Ho Chi Minh City, Vietnam*
[3]*Faculty of Civil Engineering, Ton Duc Thang University, Ho Chi Minh City, Vietnam*
[4]*Faculty of Information Technology, Industrial University of Ho Chi Minh City, Ho Chi Minh City, Vietnam*
[5]*Faculty of Mechanical Engineering, Industrial University of Ho Chi Minh City, Ho Chi Minh City, Vietnam*

Correspondence should be addressed to T. Nguyen-Trang; nguyentrangthao@tdtu.edu.vn

Received 16 April 2018; Revised 20 January 2019; Accepted 30 January 2019; Published 27 February 2019

Academic Editor: Manuel E. Acacio Sanchez

In this paper, a hybrid approach that combines a population-based method, adaptive elitist differential evolution (aeDE), with a powerful gradient-based method, spherical quadratic steepest descent (SQSD), is proposed and then applied for clustering analysis. This combination not only helps inherit the advantages of both the aeDE and SQSD but also helps reduce computational cost significantly. First, based on the aeDE's global explorative manner in the initial steps, the proposed approach can quickly reach to a region that contains the global optimal value. Next, based on the SQSD's locally effective exploitative manner in the later steps, the proposed approach can find the global optimal solution rapidly and accurately and hence helps reduce the computational cost. The proposed method is first tested over 32 benchmark functions to verify its robustness and effectiveness. Then, it is applied for clustering analysis which is one of the problems of interest in statistics, machine learning, and data mining. In this application, the proposed method is utilized to find the positions of the cluster centers, in which the internal validity measure is optimized. For both the benchmark functions and clustering problem, the numerical results show that the hybrid approach for aeDE (HaeDE) outperforms others in both accuracy and computational cost.

1. Introduction

Optimization has been widely applied in different fields such as economics, finance, engineering, etc. Although there are many optimization algorithms developed in various ways, they can be decomposed into two major techniques: population-based algorithms and gradient-based searching algorithms.

Population-based algorithms including evolutionary algorithms and swarm-based algorithms are types of global searching techniques. Evolutionary algorithms [1–8] are inspired by biological processes that allow population to adapt to their surroundings: genetic inheritance and survival of the best chromosomes; swarm-based algorithms [9–16] that focus on the social behaviors of insects and animals can solve the optimal problem as well. Among popular evolutionary algorithms, the differential evolution (DE) algorithm firstly introduced by Storn and Price [8] has been used in many practical problems and has demonstrated good convergence properties. In DE, individual solutions are selected from a population of solutions according to their fitness value to generate new offspring using some operators, such

as the crossover and the mutation operators. Nevertheless, similar to many other population-based optimization algorithms, the DE is still costly to approximate the global optimal solution. To overcome this drawback, the adaptive elitist differential evolution (aeDE) [17] in which two modifications are implemented was proposed. Firstly, an adaptive technique based on the variations between the best objective function and other objective functions in the current generation is proposed to choose a suitable mutation operator. The purpose of this modification is to preserve the balance between global and local searching abilities in the DE. Secondly, an elitist selection technique is utilized to speed up the convergence. The aeDE with those modifications is more efficient than the DE and is considered as one of the state-of-the-art methods in population-based algorithms, currently. However, the aeDE is a not-so-radical idea because of the characteristics of population-based techniques existing in it. In the later steps, when the current best solution is nearly obtained, the aeDE still utilizes the crossover and mutation operators, which leads to an unexpected additional number of function evaluations (FES) but cannot ensure the improvement of the objective function through iterations. Hence, it not only lacks locally effective exploitative behavior but also increases the computational cost.

On the other hand, gradient-based techniques only compute a single solution at any time and move the solution to a better one through iterations basing on gradient information. In comparison to population-based techniques, gradient-based techniques have more advantages in terms of locally exploitative behavior and computational cost, but they usually give the optimal solution which gets stuck at the local extreme values.

To avoid the disadvantages of both population-based and gradient-based algorithms, this paper proposes a hybrid approach that combines them together. Specifically, the state-of-the-art population-based algorithm, aeDE, is performed in the initial steps to explore the global searching space. In the later steps, when the aeDE nearly converges to a critical value, spherical quadratic steepest descent (SQSD) [18], a gradient-based method, is utilized to help obtain the fast and accurate optimal result. The reason for choosing SQSD instead of other gradient-based methods is its reliability and stability for solving extremely ill-conditioned problems [19]. The proposed algorithm is illustrated in detail through a bivariate function and compared with existing optimization algorithms through 32 benchmark numerical optimization functions. Finally, it is applied for clustering which is an interesting problem in statistics, machine learning, and data mining.

The remainder of this paper is organized as follows. The following section presents the aeDE, the SQSD, and the proposed algorithm. Section 3 evaluates the performance of the proposed method via 32 benchmark functions. In Section 4, the clustering method based on HaeDE and its performance are presented and evaluated. After discussing the advantages and disadvantages as well as the future research direction in Section 5, the conclusion is given in Section 6.

2. Materials and Methods

This section reviews some theories regarding the aeDE algorithm, SQSD algorithm, and proposes the hybrid HaeDE algorithm. The detail is presented as follows.

2.1. aeDE Algorithm. To clarify the notation used throughout this article, we refer to the minimization of the objective function $f(\mathbf{x})$, where $\mathbf{x}$ is a vector of N variables in the decision space $D = [\mathbf{x}_l, \mathbf{x}_u]$. The aeDE seeks for optimal solution through generations (iterations). In each generation, the aeDE evolves population, which is a set of NP feasible solutions, $P = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{\text{NP}}\}$. Each element in this set or each feasible solution $\mathbf{x}_i, i = \overline{1, \text{NP}}$ is called a chromosome, which is a vector of N variables, so-called genes. The four major phases of the aeDE algorithm, which include initialization, mutation, crossover, and selection, are briefly summarized.

2.1.1. Initialization. The initialization phase of the aeDE is similar to that of the original DE, in which an initial population, including NP individuals, is generated through a random sampling technique. Specifically, each individual is represented as a chromosome containing N genes and is created by

$$\mathbf{x}_{i,j} = \mathbf{x}_j^l + \text{rand}[0,1] \times \left(\mathbf{x}_j^u - \mathbf{x}_j^l\right), \quad i = 1, 2, ..., \text{NP}; \; j = 1, 2, ..., N, \tag{1}$$

where $\mathbf{x}_j^l$ and $\mathbf{x}_j^u$ are, respectively, the lower and upper bounds of $\mathbf{x}_j$, rand [0, 1] is the real number having the uniform distribution within [0, 1], and NP is the population size.

2.1.2. Mutation. In the case of the original DE, a mutant vector $\mathbf{v}_i$ is generated by individuals $\mathbf{x}_i$ in the population through mutation operations. Some mutation operations that are regularly used in the DE can be listed as follows.

(i) rand/1: $\mathbf{v}_i = \mathbf{x}_{r_1} + F \times (\mathbf{x}_{r_2} - \mathbf{x}_{r_3})$

(ii) rand/2: $\mathbf{v}_i = \mathbf{x}_{r_1} + F \times (\mathbf{x}_{r_2} - \mathbf{x}_{r_3}) + F \times (\mathbf{x}_{r_4} - \mathbf{x}_{r_5})$

(iii) best/1: $\mathbf{v}_i = \mathbf{x}_{\text{best}} + F \times (\mathbf{x}_{r_1} - \mathbf{x}_{r_2})$

(iv) best/2: $\mathbf{v}_i = \mathbf{x}_{\text{best}} + F \times (\mathbf{x}_{r_1} - \mathbf{x}_{r_2}) + F \times (\mathbf{x}_{r_3} - \mathbf{x}_{r_4})$

(v) current to best/1: $\mathbf{v}_i = \mathbf{x}_i + F \times (\mathbf{x}_{\text{best}} - \mathbf{x}_i) + F \times (\mathbf{x}_{r_1} - \mathbf{x}_{r_2})$

where integers r_1, r_2, r_3, r_4, r_5 are mutually exclusive integers randomly selected from {1, 2, . . ., NP}, F is the scale factor and randomly chosen within [0,2], and $\mathbf{x}_{\text{best}}$ is the best individual in the current population.

In the case of the aeDE, a new adaptive mutation scheme for the mutation phase of the DE is proposed. In this scheme, two mutation operators including "rand/1" and "current to best/1" are utilized. The "rand/1" aims to ensure diversity of the population and prohibits the population getting stuck in a local optimum and the "current to best/1" aims to accelerate convergence speed of the population by means of

guiding the population toward the best individual. These two mutation operators are adaptively chosen based on *delta*, the deviation modulus between the objective function of best individual and the objective functions of entire population in the previous generation. For more details, the new mutation scheme is described as follows.

IF *delta* > *threshold*

$$\mathbf{v}_i = \mathbf{x}_{r_1} + F \times \left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right), \tag{2}$$

ELSE

$$\mathbf{v}_i = \mathbf{x}_i + F \times \left(\mathbf{x}_{\text{best}} - \mathbf{x}_i\right) + F \times \left(\mathbf{x}_{r_1} - \mathbf{x}_{r_2}\right), \tag{3}$$

ENDIF

In above pseudocode, delta is defined by $delta = |f_{\text{mean}} - f_{\text{best}}|$, where f_{best} is the objective function value of the best individual and f_{mean} is the mean objective function value of the whole population; the choice of threshold is presented in Section 3.

2.1.3. Crossover. After completing mutation, each target vector $\mathbf{x}_i$ produces a trial vector $\mathbf{u}_i$ by substituting some components of the vector $\mathbf{x}_i$ by some components of the mutant vector $\mathbf{v}_i$ through the following binomial crossover operation.

$$\mathbf{u}_{ij} = \begin{cases} \mathbf{v}_{ij}, & \text{if rand } [0,1] \leq CR, \\ \mathbf{x}_{ij}, & \text{otherwise,} \end{cases} \tag{4}$$

where $i \in \{1, 2, \ldots, \text{NP}\}$; $j \in \{1,2,\ldots, N\}$, and CR is the crossover control parameter chosen within [0, 1].

2.1.4. Selection. In the selection process of the classical DE, each trial vector $\mathbf{u}_i$ created after crossover phase will be evaluated and compared with the target vector $\mathbf{x}_i$ to choose a better individual for the next generation. In the selection process of the aeDE algorithm, the elitist selection technique that was introduced by Padhye et al. [20] is utilized instead of the basic selection in the classical DE. In this new mechanism, NP best individuals are chosen from the set of NP trial vectors $\mathbf{u}_i$ and NP parent vectors $\mathbf{x}_i$. In this way, the current best individual of the whole population is always stored for the next generation, but with better convergence rate in comparison with the classical DE.

2.2. SQSD Algorithm. The SQSD algorithm is briefly summarized through the pseudocode as follows (Algorithm 1).

In above algorithm, the step limit d and the test of whether $\|x^k - x^{k-1}\| > d$ or not are used to control the step size between x^{k-1} and x^k in iterations. Generally, a small step size can avoid oscillations and guarantee the algorithm convergence but leads to slow convergence.

2.3. The Proposed Algorithm. As mentioned earlier, both aeDE and SQSD have their own advantages and disadvantages; therefore, we propose a hybrid approach for aeDE, called HaeDE, to create resonance between their advantages and avoid their disadvantages. The proposed algorithm is presented by following pseudocode (Algorithm 2) and Figure 1.

In above algorithm, NP is the population size. In case of low dimensions, according to [17], when NP > 15, the deviation of the optimal value is not significant but the run time is significantly proportional to NP. In case of very high dimensions, a popular method is to adapt NP according to the dimensions N. There already exists some proposals as $\text{NP} = 10N$ or $\text{NP} = 10^N$ [1, 21–23], but most of them have problems with premature convergence and high computational cost. Some studies deal with these problems by separating NP individuals into islands for reducing the computational cost [24, 25]. Another approach is to use a low NP with center-based initialization which is mathematically proved that it can increase the probability of finding the global optimum, especially when $N > 30$ [21, 26]. In this paper, because most of functions have $N < 30$, the approach of [17] is applied. Therefore, we choose NP = 20 to balance the computational cost and the quality of the solution. For the scale factor F, in general, a small value of F cannot explore the search space effectively and, as a result, cannot reach the optimal solution on the completion of the algorithm. In contrast, using a high value of F results in the occasional movement; hence, the algorithm has a weak exploitation behavior for reaching the global optimum in the later steps. For CR, the trial vector tends to be the same with the mutant vector when $CR \longrightarrow 1$ and the same with the target vector when $CR \longrightarrow 0$. Although the original version of DE uses fixed values of CR and F, it can be obviously claimed that proper choices of F and CR depend not only on the problem but also on the stage of the optimization process; hence, they must vary over time. The varied control parameters F and CR have demonstrated their successful performance in solving a variety of large-scale multivariate problems [17, 27–30]. Based on the above discussions and for the sake of comparison with the aeDE, in this paper, at each iteration, the values of scale factor F and crossover control parameter CR are recorded as specified in the aeDE [17]. In particular, we generate the F and CR values under uniform distributions on [0.4,1.0] and [0.7,1.0], respectively. The threshold, ε_m, is chosen based on the tolerance ε_a and must be greater than the tolerance. It has effects on the aeDE solutions as well as the initial solution of SQSD. The larger threshold is, the faster convergence and less number of function evaluations the aeDE has and vice versa. Normally, $\varepsilon_a = 10^{-6}$ is chosen; however, in the proposed method, the aeDE needs to stop when $\varepsilon_a > 10^{-6}$ for utilizing SQSD in later steps. Therefore, we set $\varepsilon_a = 10^{-5}$ and $\varepsilon_m = 10^{-2}$ in the numerical examples. For the SQSD, there are three parameters needing to be set up: the convergence tolerances ε_g, ε_x, and the step limit d. As mentioned earlier, the step limit d is used to keep the step size smaller than d. If d is too small, the algorithm will slowly converge, particularly for high dimensions. In contrast, a too large value of d may result in excessive oscillations occurring before convergence, particularly for

INPUT: $f(x)$ and its domain $[a,b]$, convergence criteria ε_g, ε_x and step limit $d>0$
Initialize a starting point x^0, randomly
Compute $\nabla f(x^0)$, $c_0 := \|\nabla f(x^0)/d\|$, $k:=1$
REPEAT
  $x^k := x^{k-1} - (\nabla f(x^{k-1})/c_{k-1})$
  IF $\|x^k - x^{k-1}\| > d$
    $x^k := x^{k-1} - d(\nabla f(x^{k-1})/\|\nabla f(x^{k-1})\|)$
  ENDIF
  IF $\|x^k - x^{k-1}\| < \varepsilon_x$
    $x^* \cong x^c = x^k$
    Stop the algorithm
  ENDIF
  Set $c^k := (2[f(x^{k-1}) - f(x^k) - \nabla^{\mathrm{T}} f(x^k)(x^{k-1} - x^k)]/\|x^{k-1} - x^k\|^2)$
  IF $c^k < 0$
    $c^k = 10^{-60}$
  ENDIF
  $k := k+1$
UNTIL $\|\nabla f(x^{k-1})\| < \varepsilon_g$
$x^* \cong x^c = x^{k-1}$
OUTPUT: x^* and $f(x^*)$

Algorithm 1: Given the function $f(x)$ and its domain $[a,b]$, this algorithm finds the $x* \in [a,b]$ such that $f(x*) = \min_{x\in[a,b]} f(x)$.

INPUT: $f(x)$ and its domain $[a,b]$; convergence criteria ε_m, ε_a $(\varepsilon_m > \varepsilon_a)$,$\varepsilon_g$, ε_x, population size NP, scale factor F, crossover control parameter CR, and step limit d
Use Formula (1) to generate an initial population of NP individuals
Compute $delta = |f_{\text{mean}} - f_{\text{best}}|$
WHILE $delta > \varepsilon_m$
  Run the aeDE using mutation operator "rand/1" (formula (2)), binomial crossover operator (formula (4)) and elitist selection technique
  Compute $delta = |f_{\text{mean}} - f_{\text{best}}|$
ENDWHILE
WHILE $delta > \varepsilon_a$
  Run the aeDE using mutation operator "current to best/1" (formula (3)), binomial crossover operator (Formula (4)) and elitist selection technique
  Compute $delta = |f_{\text{mean}} - f_{\text{best}}|$
ENDWHILE
Initialize a starting point $x^0 = x_{\text{best}}$
Run SQSD algorithm using convergences criteria ε_g and ε_x, step limit $d>0$
OUTPUT: x^* and $f(x^*)$

Algorithm 2: Given the function $f(x)$ and its domain $[a,b]$, this algorithm finds the $x* \in [a,b]$ such that $f(x^*) = \min_{x\in[a,b]} f(x)$.

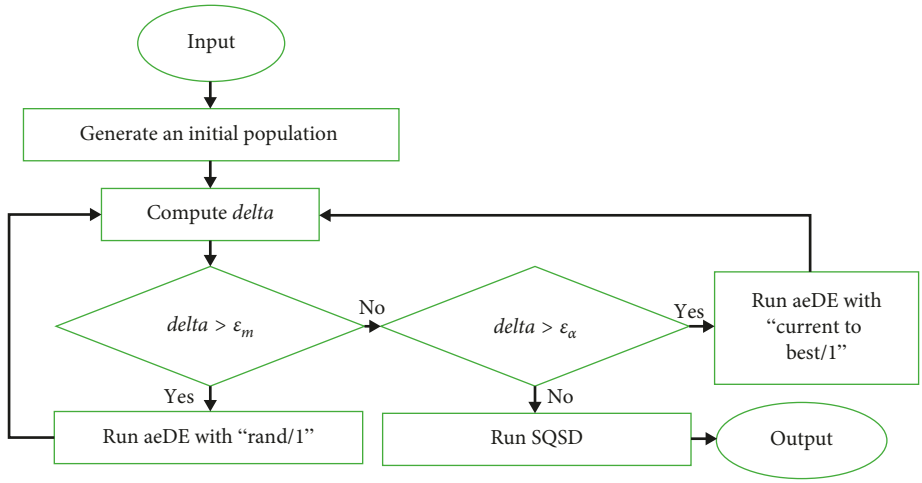

Figure 1: Flow chart of the HaeDE.

low dimensions. Therefore, in case of two dimensions, a relatively small value, for instance, $d = 0.3$, is required to be used. In case of high dimensions, the step limit d hinges on the number of function dimensions. Specifically, in case of quadratic functions, no step limit d is required because it is proved that the SQSD algorithm (without step size control) is always convergent when it is applied to the general quadratic function. This characteristic is very useful for optimizing a variety quadratic function problems, with very high dimensions, for instance, $N = 5000$, as presented in [18]. In this paper, because the number of dimensions is varied from 2 to 30, the step limit d should hinge on the number of dimensions. Furthermore, in the case of HaeDE, to ensure the exploitation in the later steps, the variation between x^{k-1} and x^k through iterations needs to be smaller than the aeDE variants. Hence, we choose the step limit $d = (\|x_{\text{best}} - x_{\text{worst}}\|/100)\sqrt{n}$, where x_{best} and x_{worst} are the best and the worst individual in the latest step of aeDE and n is the number of dimensions of function, respectively. For the convergence tolerances, $\varepsilon_g = 10^{-6}$ and $\varepsilon_x = 10^{-8}$ are applied.

3. Experiments on Benchmark Functions

This section presents two examples to illustrate and test the performance of the proposed method. Particularly, the first example describes the details of the proposed method when dealing with a well-known function, Bohachevsky1. The purpose of this experiment with a simple bivariate function is to analyze the proposed method behavior. As a result, we can illustrate how the new method works and why it is better than aeDE. In the second example, the performance of the proposed HaeDE algorithm is evaluated on 32 benchmark functions. Those functions are in 50 functions firstly performed by [31] and were often utilized later in order to compare the performance between optimization algorithms [1, 32]. Because gradient-based method can deal with unimodal functions that have only one optimal value, it is unnecessary to utilize the population-based algorithm to solve those functions. Therefore, in current research, the HaeDE performance is compared to those of the aeDE [17], particle swarm optimization (PSO) [11], Selfish Herd Optimizer (SHO) [9], Salp Swarm Algorithm (SSA) [14], and Dragonfly algorithm (DA) [13] using 32/50 functions that are multimodal. For each benchmark function, all methods are run 30 independent times with the same initial population in each time. The obtained results are then compared using Wilcoxon's paired tests. Finally, the computational cost, especially the number of FES of all methods, is examined.

3.1. Experiment on Function Bohachevsky1. In this subsection, we perform experiment on function Bohachevsky1, a simple bivariate function, to analyze the proposed method in detail. In the first phase, we run the aeDE algorithm with $delta = |f_{\text{mean}} - f_{\text{best}}| < 10^{-5}$ to determine the best individual in the current population x^0 which will be the initial point for the next phase. In the second phase, two cases are examined in which the usage of the aeDE is kept in the first case, and the SQSD is utilized in the second case to look for the final optimal solution. The first case is hence exactly the original aeDE, while the second case is named HaeDE which is the proposed algorithm in this study. The performance of the aeDE and HaeDE are then measured using the corresponding best fitness value (FBEST) and number of function evaluations (FES). In addition, the effect of step limit d on HaeDE performance is also examined. We need to look for the minimum of function Bohachevsky1 whose formulation, surface, and contour are presented as follows.

$$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7. \tag{5}$$

It can be seen from Figure 2 that the Bohachevsky1 is a multimodal function which has a large number of local solutions; as a result, gradient-based method that may be easily trapped to local minimum is unsuitable in this case. A more feasible strategy to solve this problem is to use evolution-based algorithm, like aeDE algorithm, for instance. As evidenced by Figure 3, aeDE (the blue line) has a good explorative manner when the FBEST rapidly decreases in the initial steps. However, in the later steps, when the number of iterations is about 30 or the number of function evaluations (FES) is about 600, the FBEST exhibits a slow decrease if the aeDE continues to be used (the red line). The primary reason is that aeDE still utilizes the crossover and mutation operators, which leads to unexpected additional FES but cannot ensure the improvement of fitness value in iterations. In contrast, if the SQSD algorithm is utilized in the later steps, FBEST value can quickly decrease through each step depending on the gradient information (the green line). As a result, it not only makes a better result but also saves the computational cost when taking only one FES for each iteration.

In addition to comparing with aeDE, the convergence behavior of HaeDE itself with different step limit parameters is examined. It can be seen from Figure 4 that when d is not small enough, for instance $d = 10^{-4}$, the algorithm converges after one or two iterations but FBEST is still far from the true optimal value of 0. On the contrary, the convergence speed of the HaeDE with a too small d, for instance $d = 10^{-7}$, falls behind others. The proposed step limit $d = (\|x_{\text{best}} - x_{\text{worst}}\|/100)\sqrt{n}$ in this paper takes more than ten iterations to converge, but it can help the algorithm reach the feasible FBEST which approximates the true optimal value of 0. Moreover, the proposed step limit in this paper can adapt to many cases of problems using different number of dimensions and can adapt to the quality of the last population in aeDE.

3.2. Experiment on 32 Benchmark Functions. In this subsection, we compare the performance of the proposed method with those of five well-known algorithms consisting of the aeDE, PSO, SHO, SSA, and DA over 32 benchmark functions. The control parameter settings for the HaeDE are

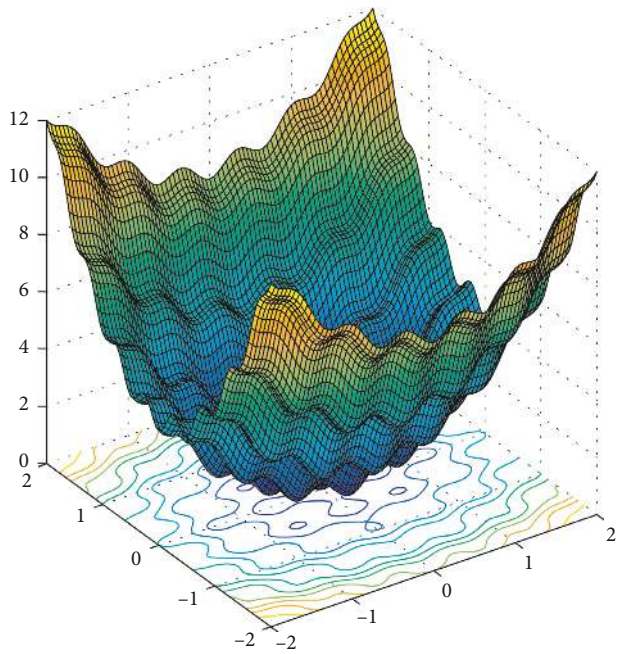

Figure 2: Surface and contour of function Bohachevsky1.

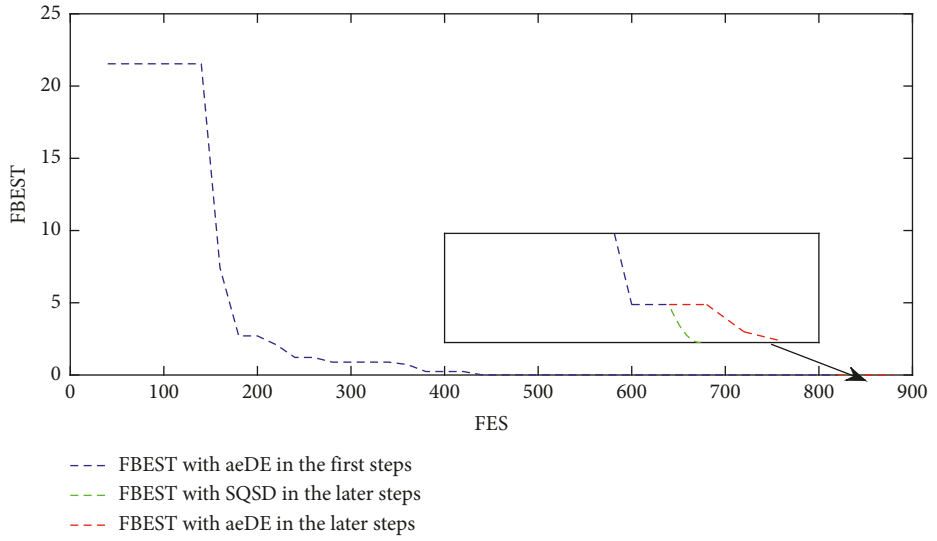

Figure 3: The convergence behavior of aeDE and HaeDE.

chosen as mentioned in Section 2. For other methods, we set the parameters as follows:

(i) aeDE: the population size NP = 20, the stop criterion $\Delta = 10^{-6}$, the maximum number of iterations max*iter* = 5000, the mutant factor $F \in [0.4, 1]$, the crossover control parameter CR $\in [0.7, 1]$, and the threshold $\varepsilon = 10^{-2}$.

(ii) PSO: the population size NP = 20, the stop criterion $\Delta = 10^{-6}$, the maximum number of iterations max*iter* = 5000, the initial velocity of particles is $v \in [0, (\text{Ub} - \text{Lb}/7)]$ where Ub and Lb are the upper and lower bounds of solutions; acceleration factors c_1 and c_2 are 0.5 and 1, respectively; the inertia weight $w = w_{\max} - (w_{\max} - w_{\min}/\max iter)iter$ where $w_{\max}$ and $w_{\min}$ denote the maximum and minimum values of the inertia weight; max*iter* denotes the maximum number of iterations and *iter* is the current number of iterations.

(iii) SHO, SSA, DA: the population size NP = 20, the stop criterion $\Delta = 10^{-6}$, the maximum number of iterations max*iter* = 5000; for detailed description of other parameter setting, please refer to [9, 13, 14].

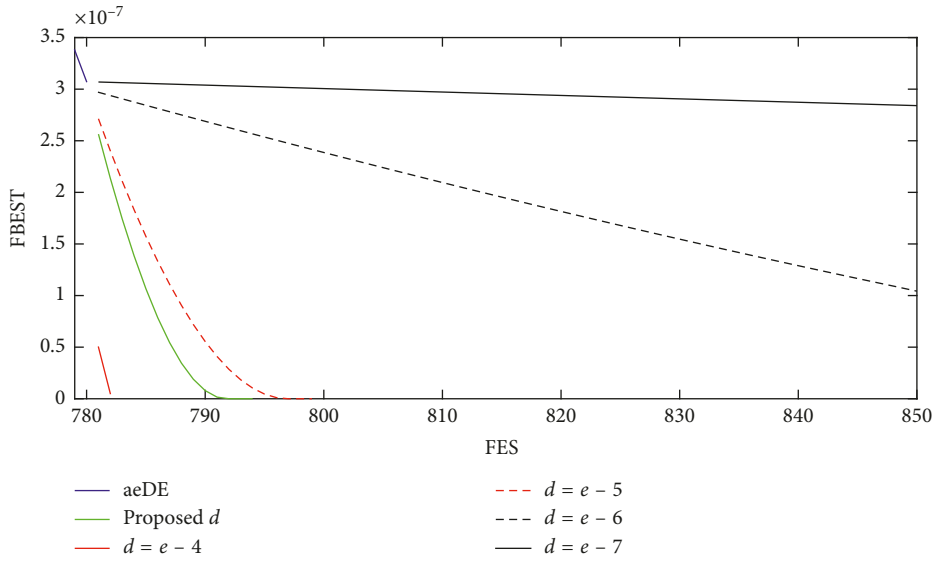

Figure 4: Convergence behavior of HaeDE with different step limit value.

For each benchmark function, we run HaeDE, aeDE, PSO, SHO, SSA, and DA 30 independent times. To ensure the fairness, initial population for comparative methods is chosen such that they are the same. To determine whether HaeDE reaches a statistically better solution than other methods or not, Wilcoxon's paired tests are examined. During the test, if the optimal value is below 10^{-9}, a very small positive number, it will be considered as zero. The benchmark functions and their characteristics are summarized in Table 1. The descriptive statistics of the comparative methods are presented in Table 2 where the first number is the mean of FBEST, and the number in parentheses is the rank of method in ascending order of FBEST. Through 32 benchmark functions, it can be seen that HaeDE is the best with the smallest total of ranks.

Although Table 2 provides a first insight into the performance of the algorithms, it is more reliable if we compare the performance by using statistical test. For this purpose, with the null hypothesis "there is no difference between two methods", the obtained results are tested using Wilcoxon signed-rank test with a statistical significance value $\alpha = 0.1$.

In Table 3, "+" indicates the case in which the null hypothesis is rejected and the HaeDE is better than the comparative method, "−" indicates the case in which the null hypothesis is rejected and the HaeDE is worse than the comparative method, and "=" indicates the case in which we cannot reject the null hypothesis. According to the total count of (+/=/−) presented in the last row of Table 3, it can be seen that the HaeDE outperforms aeDE, PSO, SHO, and DA, and it is competitive with SSA in terms of approximating the optimal value.

Finally, we examine the convergence behaviors of the comparative methods. Figure 5 illustrates some results for large problems with 30 dimensions. It can be seen that the PSO, SHO, and DA have slow convergence in general. The SSA is a competitive method with HaeDE as mentioned earlier but easily gets stuck in a specific point and takes a large number of FES for moving to a better individual. According to Figure 5, while the HaeDE quickly converges and stops, the convergence curve of SSA is nearly a straight line or FBEST is nearly constrained for a long time. As a result, it becomes the worst method in terms of the convergence speed (see more in Figure 6). The aeDE is an improved version of DE, which utilizes the mutation operator "current to best/1" and elitist selection technique to speed up the convergence. Therefore, it is not surprising that the aeDE outperforms PSO, SHO, SSA, and DA and ranks second (the red line). As explained in example 1, the HaeDE (the green line) is again computationally inexpensive compared to the aeDE. Figures 6 and 7 illustrate an overview of the number of FES and total CPU time consumed by all methods for all the benchmark functions over 30 independent runs. Clearly, in most cases, the HaeDE is the best in terms of computational cost. The results obtained from the above experiments demonstrate the promising performance of the proposed method when solving the optimization problems and especially when solving continuously differentiable functions. In addition, we can estimate the gradient numerically, for instance Euler approximation, if the function itself is nondifferentiable. Hence, the HaeDE can be utilized for further practical problems with discontinuous, nondifferentiable, and implicit objective functions. The above discussion and the research results have shown that the HaeDE is a competitive optimization algorithm and can be utilized in further application as the clustering problem.

4. An Application of HaeDE for Clustering Analysis

Clustering is a data mining technique that can partition unknown large data into groups so that elements in each group have the similar properties. It is the important first

Table 1: The benchmark functions and their characteristics.

No	N	Name	Formulation
1	2	Branin	$f(x) = (x_2 - (5.1/4\pi^2)x_1^2 + (5/\pi)x_1 - 6)^2 + 10(1 - (1/8\pi))\cos x_1 + 10$
2	2	Bohachevsky 1	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$
3	2	Booth	$f(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$
4	30	Rastrigin	$f(x) = \sum_{i=1}^{n}[x_i^2 - 10\cos 100(2\pi x_i) + 10]$
5	30	Schwefel	$f(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$
6	2	Michalewicz 2	$f(x) = -\sum_{i=1}^{n}\sin(x_i)(\sin(ix_i^2)/\pi)^{20}$
7	5	Michalewicz 5	$f(x) = -\sum_{i=1}^{n}\sin(x_i)(\sin(ix_i^2)/\pi)^{20}$
8	10	Michalewicz 10	$f(x) = -\sum_{i=1}^{n}\sin(x_i)(\sin(ix_i^2)/\pi)^{20}$
9	2	Schaffer	$f(x) = 0.5 + \sin^2(\sqrt{x_1^2 + x_2^2}) - 0.5/(1 + 0.001(x_1^2 + x_2^2))^2$
10	2	Six hump camel back	$f(x) = 4x_1^2 - 2.1x_1^4 + (1/3)x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$
11	2	Bohachevsky 2	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)(4\pi x_2) + 0.3$
12	2	Bohachevsky 3	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1 + 4\pi x_2) + 0.3$
13	2	Shubert	$f(x) = [\sum_{i=1}^{5} i\cos((i+1)x_1 + i)][\sum_{i=1}^{5} i\cos((i+1)x_2 + i)]$
14	2	GoldStein-Price	$f(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)].[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 + 36x_1x_2 + 27x_2^2)]$
15	4	Kowalik	$f(x) = \sum_{i=1}^{11} a_i - ((x_1(b_i^2 + b_ix_2))/(b_i^2 + b_ix_3 + x_4))$
16	4	Shekel 5	$f(x) = -\sum_{i=1}^{5}\sum_{j=1}^{4}[(x_j - a_{ij})(x_j - a_{ij})^T + c_i]$
17	4	Shekel 7	$f(x) = -\sum_{i=1}^{7}\sum_{j=1}^{4}[(x_j - a_{ij})(x_j - a_{ij})^T + c_i]$
18	4	Shekel 7	$f(x) = -\sum_{i=1}^{10}\sum_{j=1}^{4}[(x_j - a_{ij})(x_j - a_{ij})^T + c_i]$
19	4	Perm	$f(x) = \sum_{k=1}^{n}[\sum_{i=1}^{n}(i^k + \beta)((x_i/i)^k - 1)]^2$
20	4	PowerSum	$f(x) = \sum_{k=1}^{n}[(\sum_{i=1}^{n} x_i^k) - b_k]^2$
21	3	Hartman 3	$f(x) = -\sum_{i=1}^{4} c_i \exp[-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2]$
22	6	Hartman 6	$f(x) = -\sum_{i=1}^{4} c_i \exp[-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2]$
23	30	Griewank	$f(x) = (1/4000)\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(x_i/\sqrt{i}) + 1$
24	30	Ackley	$f(x) = -20\exp(-0.2\sqrt{(1/n)\sum_{i=1}^{n} x_i^2}) - \exp((1/n)\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$
25	30	Penalized	$f(x) = (\pi/n)\{10\sin^2(\pi y_1) + \sum_{i=1}^{n}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$, $y_i = 1 + (1/4)(x_i + 1)$, $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a, \\ 0, & -a \le x_i \le a, \\ k(-x_i - a)^m, & x_i < -a. \end{cases}$
26	30	Penalized 2	$f(x) = 0.1\{\sin^2(\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$
27	2	Langerman 2	$f(x) = -\sum_{i=1}^{m} c_i\{\exp[-1/\pi\sum_{j=1}^{n}(x_j - a_{ij})^2]\cos[\pi\sum_{j=1}^{n}(x_j - a_{ij})^2]\}$
28	5	Langerman 5	$f(x) = -\sum_{i=1}^{m}\sum c_i\{\exp[-(1/\pi)\sum_{j=1}^{n}(x_j - a_{ij})^2]\cos[\pi\sum_{j=1}^{n}(x_j - a_{ij})^2]\}$
29	10	Langerman 10	$f(x) = -\sum_{i=1}^{m} c_i\{\exp[-(1/\pi)\sum_{j=1}^{n}(x_j - a_{ij})^2]\cos[\pi\sum_{j=1}^{n}(x_j - a_{ij})^2]\}$
30	2	Fletcher Powell 2	$f(x) = \sum_{i=1}^{n}[\sum_{j=1}^{n} a_{ij}\sin\alpha_j + b_{ij}\cos\alpha_j - a_{ij}\sin x_j - b_{ij}\cos x_j]^2$
31	5	Fletcher Powell 5	$f(x) = \sum_{i=1}^{n}[\sum_{j=1}^{n} a_{ij}\sin\alpha_j + b_{ij}\cos\alpha_j - a_{ij}\sin x_j - b_{ij}\cos x_j]^2$
32	5	Fletcher Powell 5	$f(x) = \sum_{i=1}^{n}[\sum_{j=1}^{n} a_{ij}\sin\alpha_j + b_{ij}\cos\alpha_j - a_{ij}\sin x_j - b_{ij}\cos x_j]^2$

step to understand some basic information from data before implementing deeper analysis [33–37]. Therefore, the clustering problem has been extensively researched in many areas such as pattern recognition, bioengineering, image processing, renewable energy prediction, etc. [33, 38–41].

Mathematically, let $X = \{x_1, x_2, ..., x_N\}$ be a set of N elements given in R^n, we need to find an optimal way to partition them into k clusters $U = \{C_1, C_2, ..., C_k\}$, $C_i \cap C_j = \phi$ so that the elements belonging to the same cluster are as similar as possible, in terms of a given internal validity measure $m(U)$. It implies that the clustering problem can be transformed into an optimization problem where $m(U)$ needs to be maximized or minimized. The formulation of the optimization problem for clustering in this paper is present in Section 4.1.

4.1. The Formulation of the Optimization Problem for Clustering

4.1.1. Chromosome Representation. In optimization problem, each individual or each partition of clustering result is a

Table 2: The descriptive statistics.

No	Min	FBEST mean (rank)					
		HaeDE	aeDE	PSO	SHO	SSA	DA
1	0.398	0.3979 (4)	0.3979 (5)	0.3979 (1)	0.3979 (2)	0.3979 (3)	0.3979 (6)
2	0	0 (1)	0 (3)	0.0413 (6)	0.0138 (5)	0 (2)	0.0014 (4)
3	0	0 (4)	0 (5)	0 (1)	0 (3)	0 (2)	0 (6)
4	0	20.1717 (1)	20.6465 (2)	183.3748 (6)	55.2204 (3)	72.7645 (4)	149.0701 (5)
5	−12569.5	−12027.9391 (1)	−11886.7727 (2)	−6470.1728 (5)	−10658.3841 (3)	−7816.4553 (4)	−5825.5215 (6)
6	−1.8013	−1.8013 (2)	−1.8013 (4)	−1.8013 (1)	−1.7817 (6)	−1.8013 (3)	−1.8013 (5)
7	−4.6877	−4.6527 (1)	−4.6486 (2)	−4.5941 (3)	−4.1299 (5)	−4.3921 (4)	−3.854 (6)
8	−9.6602	−9.4678 (2)	−9.4813 (1)	−7.6207 (5)	−7.8455 (3)	−7.7399 (4)	−6.1953 (6)
9	0	0.0058 (3)	0.0068 (4)	0.0103 (6)	0.01 (5)	0 (1)	0.0016 (2)
10	−1.03163	−1.0316 (4)	−1.0316 (5)	−1.0316 (1)	−1.0316 (2)	−1.0316 (3)	−1.0316 (6)
11	0	0 (1)	0 (3)	0.0218 (6)	0.0146 (5)	0 (2)	0.002 (4)
12	0	0 (3)	0 (4)	0 (1)	0.0151 (6)	0 (2)	0.0018 (5)
13	−186.73	−186.7309 (2)	−186.7309 (4)	−186.7309 (1)	−184.6258 (6)	−186.7309 (3)	−186.7289 (5)
14	3	3 (3)	3 (4)	3 (1)	5.7 (6)	3 (2)	3 (5)
15	0.00031	0.0013 (4)	0.0006 (2)	0.0005 (1)	0.0042 (5)	0.0008 (3)	0.0042 (6)
16	−10.15	−8.3881 (4)	−8.8613 (3)	−7.4454 (5)	−6.6581 (6)	−9.4795 (1)	−9.296 (2)
17	−10.4	−9.5579 (2)	−9.4621 (3)	−7.8864 (4)	−5.8977 (6)	−10.0486 (1)	−7.7742 (5)
18	−10.53	−9.7993 (2)	−9.9541 (1)	−8.5676 (4)	−5.3894 (6)	−9.2793 (3)	−6.8984 (5)
19	0	0.1427 (4)	0.0989 (3)	0.8412 (6)	0.0497 (2)	0.0251 (1)	0.4327 (5)
20	0	0.0056 (3)	0.0066 (4)	0.0168 (6)	0.0004 (2)	0.0002 (1)	0.0107 (5)
21	−3.86	−3.8628 (5)	−3.8628 (4)	−3.8628 (1)	−3.8628 (3)	−3.8628 (2)	−3.8623 (6)
22	−3.32	−3.2747 (2)	−3.2866 (1)	−3.238 (6)	−3.2745 (3)	−3.2429 (5)	−3.2719 (4)
23	0	0.079 (3)	0.1389 (4)	33.4143 (6)	0.0135 (1)	0.0146 (2)	3.5681 (5)
24	0	1.1253 (1)	1.2893 (2)	12.1837 (6)	2.888 (4)	2.1901 (3)	5.6063 (5)
25	0	0.2291 (2)	0.0915 (1)	57.75 (6)	3.1098 (3)	21.3042 (4)	41.9515 (5)
26	0	0.0492 (3)	0.0591 (4)	163.2041 (6)	0.0003 (1)	0.0034 (2)	20.0337 (5)
27	−1.08	−1.0764 (3)	−1.0764 (4)	−1.0809 (1)	−1.0629 (5)	−1.0809 (2)	−1.0272 (6)
28	−1.5	−1.4175 (2)	−1.4212 (1)	−0.9667 (4)	−0.7674 (5)	−1.3188 (3)	−0.5915 (6)
29		−0.2872 (3)	−0.3065 (2)	−0.1523 (5)	−0.26 (4)	−0.3763 (1)	−0.0818 (6)
30	0	0 (2)	0 (4)	0 (1)	94.0393 (6)	0 (3)	0.0004 (5)
31	0	14.7267 (2)	83.2492 (3)	1.3738 (1)	395.239 (5)	189.9685 (4)	637.8159 (6)
32	0	356.7752 (1)	1168.6692 (2)	7531.9408 (5)	1495.7068 (3)	3859.5778 (4)	9598.2252 (6)
Sum of ranks		**80**	96	118	130	84	164

chromosome that represents the position of cluster centers. Because each center is a n-dimensional vector and we need to identify k clusters or k cluster's centers, each chromosome is a string with total length equals kn. In particular, the general form of each chromosome is represented by Figure 8.

4.2. Objective Function. As mentioned before, the clustering quality is usually evaluated via an internal validity measure $m(U)$ such as Intra index [42], Xie-Beni index [43], Dunn index [44], Davis–Bouldin index [45], Silhouette coefficient [46], etc. Here, we choose the Intra index, an internal validity measure used in the well-known k-means, to be the objective function.

Let $U = \{C_1, C_2, ..., C_k\}$, $C_i \cap C_j = \phi$ where C_i stands for the cluster i which is a partition needing to be evaluated. The Intra index $m(U)$ is defined as follows.

$$m(U) = \sum_{i=1}^{k} \sum_{x \in C_k} d^2(x, v_k), \tag{6}$$

where v_k is the center of cluster k and d is the Euclidean distance. From the above expression, it is seen that the value of $m(U)$ becomes smaller when the elements in cluster are more similar to those in cluster center. Hence, minimizing the Intra index is to optimize the compactness of established clusters, which leads to a suitable partition.

After identifying the objective function and chromosome representation, we can utilize the HaeDE to find the optimal partition for clustering problem.

4.3. Experiments and Results. In this subsection, the performance of the HaeDE in clustering is tested using the Iris flower dataset, a well-known benchmark dataset introduced by Fisher [47]. The dataset consists of 3 flower classes (clusters) named Iris setosa, Iris virginica, and Iris versicolor, with 150 samples and 4 independent variables representing for the length and width of the sepals and petals. To visualize the clustering results, only the petal length and width are used as predictors; therefore, the clustering problem is now the minimization problem with six variables in range [0, 1] and the objective function is the Intra index mentioned earlier. In this case, parameters setting is similar to Section 3 for all methods. Because we have the reference of the actual classes of data, the final clustering results of the HaeDE are compared with those of aeDE, PSO, SHO, SSA, and DA

Table 3: Wilcoxon signed-rank test results.

Function	HaeDE vs aeDE		HaeDE vs PSO		HaeDE vs SHO		HaeDE vs SSA		HaeDE vs DA	
	p value	Winner	p value	Winner	p value	Winner	p value	Winner	p value	Winner
1	0.000	+	0.000	−	0.000	−	0.000	−	0.000	+
2	1.000	=	0.083	+	0.317	=	1.000	=	0.008	+
3	1.000	=	1.000	=	1.000	=	1.000	=	0.180	+
4	0.877	=	0.000	+	0.000	+	0.000	+	0.000	+
5	1.000	=	1.000	=	1.000	=	1.000	=	1.000	=
6	1.000	=	1.000	=	1.000	=	1.000	=	1.000	=
7	1.000	=	1.000	=	1.000	=	1.000	=	1.000	=
8	1.000	=	1.000	=	1.000	=	1.000	=	1.000	=
9	0.949	=	0.296	=	0.246	=	0.000	−	0.001	−
10	1.000	=	1.000	=	1.000	=	1.000	=	1.000	=
11	1.000	=	0.083	+	0.180	=	1.000	=	0.018	+
12	1.000	=	1.000	=	0.180	=	1.000	=	0.012	+
13	1.000	=	1.000	=	1.000	=	1.000	=	1.000	=
14	0.000	+	0.000	−	0.018	−	0.959	=	0.000	+
15	0.102	=	0.003	−	0.004	+	0.478	=	0.000	+
16	1.000	=	1.000	=	1.000	=	1.000	=	1.000	=
17	1.000	=	1.000	=	1.000	=	1.000	=	1.000	=
18	1.000	=	1.000	=	1.000	=	1.000	=	1.000	=
19	0.237	=	0.015	+	0.006	−	0.000	−	0.289	=
20	0.810	=	0.040	+	0.001	−	0.005	−	0.047	+
21	1.000	=	1.000	=	1.000	=	1.000	=	1.000	=
22	1.000	=	1.000	=	1.000	=	1.000	=	1.000	=
23	0.159	=	0.000	+	0.000	−	0.000	−	0.000	+
24	0.629	=	0.000	+	0.000	+	0.000	+	0.000	+
25	0.318	=	0.000	+	0.581	=	0.000	+	0.000	+
26	0.000	+	0.000	+	0.073	+	0.225	=	0.000	+
27	1.000	=	1.000	=	1.000	=	1.000	=	1.000	=
28	1.000	=	1.000	=	1.000	=	1.000	=	1.000	=
29	1.000	=	1.000	=	1.000	=	1.000	=	1.000	=
30	1.000	=	1.000	=	0.046	+	1.000	=	0.001	+
31	0.198	=	0.701	=	0.001	+	0.002	+	0.000	+
32	0.057	+	0.000	+	0.001	+	0.000	+	0.000	+
+/=/−	**4/28/0**		**10/19/3**		**7/20/5**		**5/22/5**		**16/15/1**	

using the total accuracy and the Adjusted rand index (ARI) [48] where "0" indicates that the clustering results are not fitted with the actual data classes and "1" indicates that the clustering results and the actual classes are exactly the same. The distribution of Iris dataset, the actual classes, and the clustering results for the comparative methods are presented in Figure 9 and Table 4.

As can be seen from Figure 9 and Table 4, the actual Class 1 is sufficiently well separated from the other two clusters which are strongly and significantly overlapping. In this case, the SHO obtains a quite good solution when it can separate actual Class 1 to others but cannot separate the remaining two clusters. As a result, it ranks second in terms of ARI, with ARI = 0.5149. For the aeDE, it can well recognize Class 3, but incorrectly assigns most of the Class 1 elements to Class 2. This method ranks second in terms of accuracy, with 66.67%. The best clustering result is given by the HaeDE when actual Class 1 is properly grouped. Although there are some misclustering elements in case of actual Class 2 and actual Class 3 due to their high overlapped degree, the HaeDE is still the best with ARI = 0.8857 and accuracy is about 96%. The other methods make poor performance in this case with ARI < 0.5. In terms of computational cost, it can be observed that the HaeDE is also the best when it takes only 641 FES for convergence. The aeDE ranks second when taking 880 FES and the others are worse than both aeDE and HaeDE when taking from 30000 to 100000 FES for convergence. All of above experiments and analyses demonstrate the superiority of HaeDE over the comparative methods in solving the clustering problems.

5. The Drawbacks and Future Research Direction

Although the proposed method possesses some advantages in terms of finding the global optimum and reducing computational cost, some disadvantages can be indicated as follows:

(i) Using the proposed method for solving unimodal functions is not efficient but causes large computational costs in comparison with gradient-based methods.

(ii) A parallel version of the proposed method is out of scope of this article. Parallelism is a feasible method to reduce the high computational cost by dividing

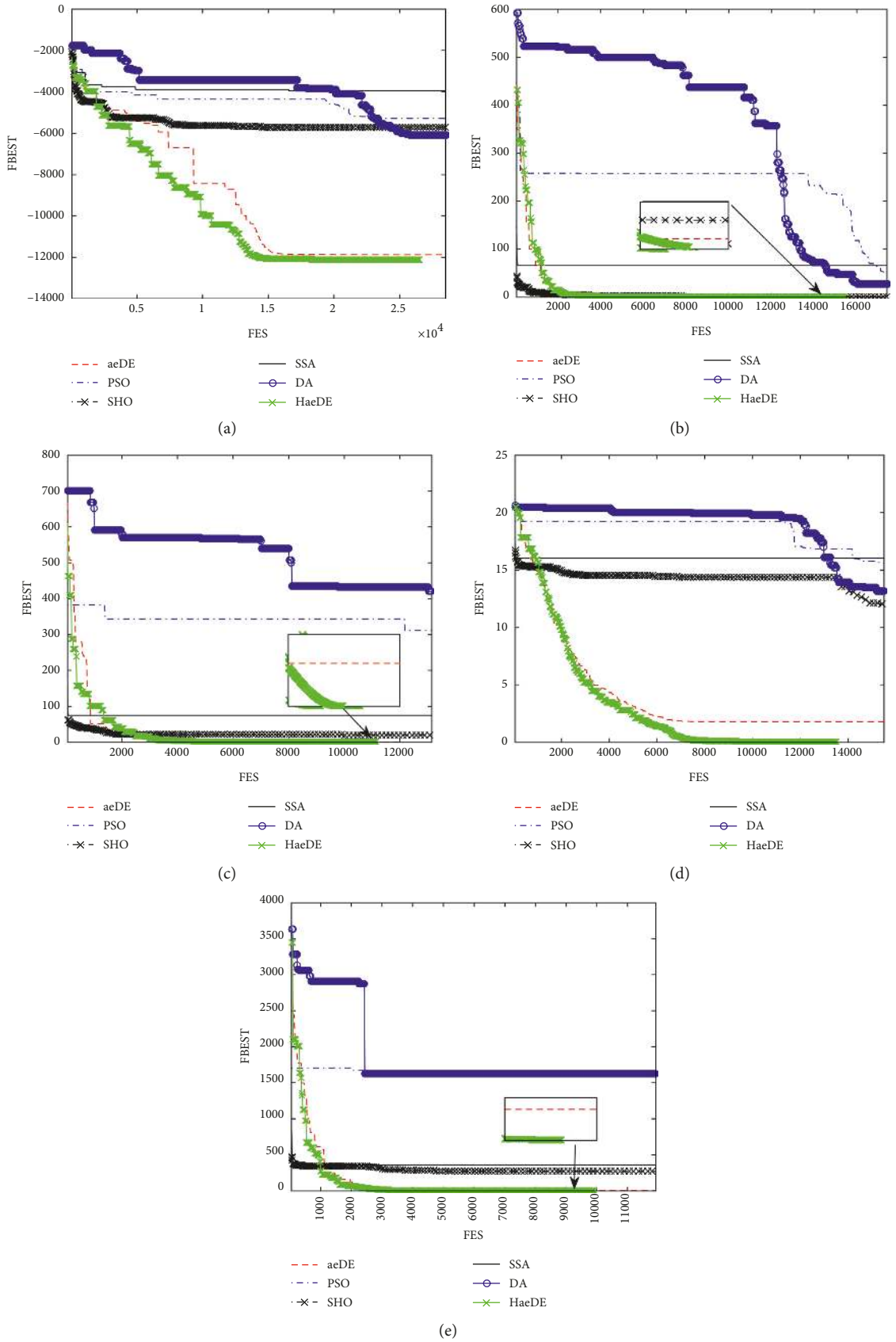

Figure 5: Convergence behaviors of comparative methods. (a) Schwefel, (b) Griewank, (c) Penalized, (d) Ackley, (e) Penalized 2.

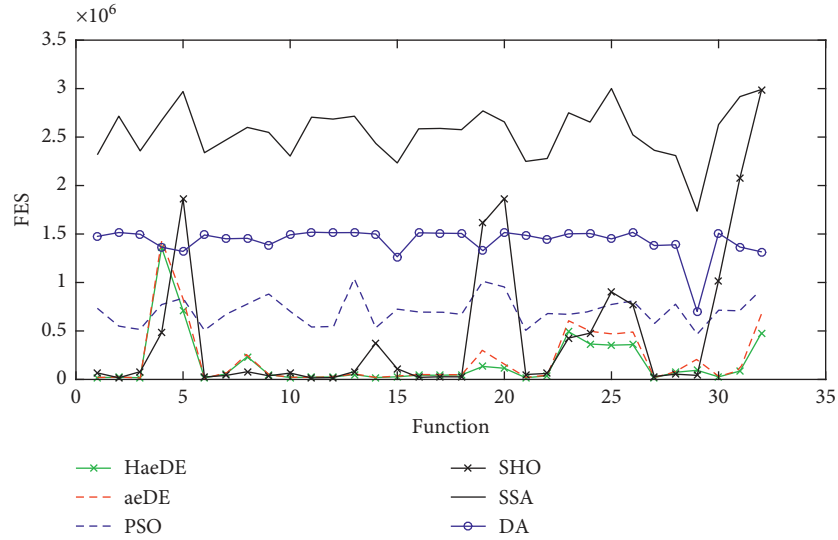

Figure 6: The number of FES of 30 runs for benchmark functions.

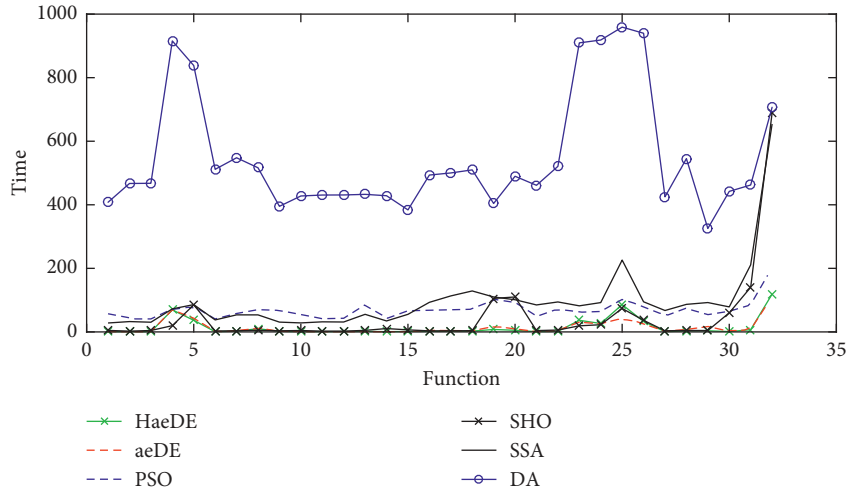

Figure 7: The computational time of 30 runs for benchmark functions.

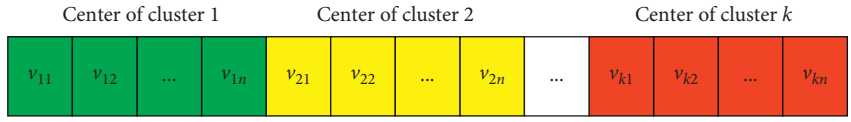

Figure 8: Chromosome representation.

the computational cost between multiple processors. Theoretically, in case that the network is homogeneous, we can predict the speedup by the number of processors (the efficiency = 1). However, the actual speedup obtained may be less than the number of available processors because the actual computer network is often a heterogeneous environment. In addition, for the proposed method, only the first stage, which is the aeDE, can typically be parallelized at all. After the first stage, the SQSD must be run sequentially. Certainly, enhancing the proposed method with parallel programming is an interesting future research direction for a wide range of researchers.

(iii) Another drawback of the paper is that the application of the proposed method to the clustering still requires a given number of clusters. In future, another encoding method can be proposed to apply the HaeDE to the clustering problem with unknown number of clusters.

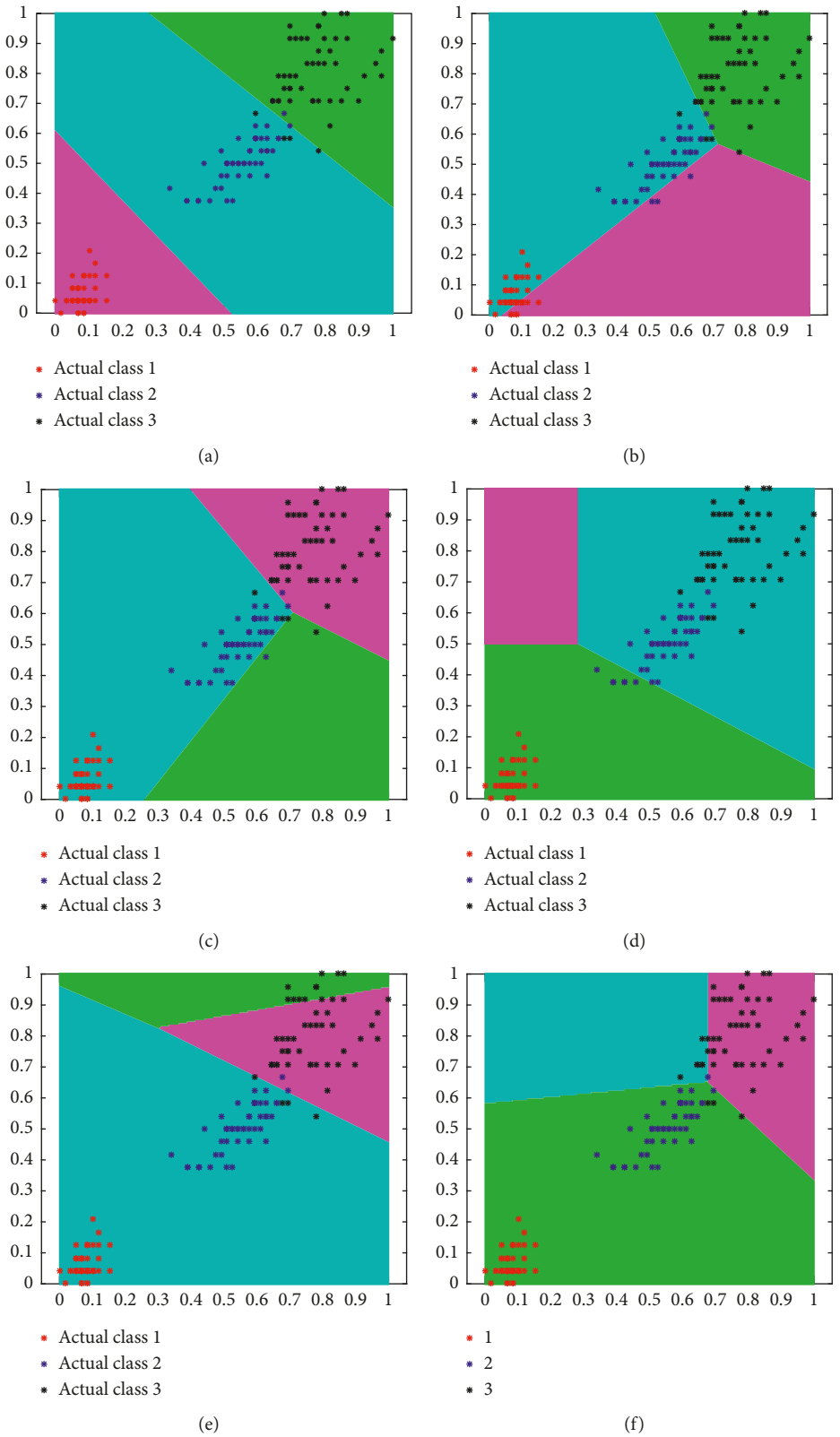

Figure 9: The clustering results. (a) HaeDE, FES = 641, ARI = 0.8857; (b) aeDE, FES = 880, ARI = 0.4124; (c) PSO, FES = 31420, ARI = 0.4730; (d) SHO, FES = 100000, ARI = 0.5149; (e) SSA, FES = 91220, ARI = 0.3775; (f) DA, FES = 50700, ARI = 0.4347.

Table 4: Summary of clustering results.

Method	FES	Accuracy	ARI
HaeDE	641	0.9600	0.8847
aeDE	880	0.6733	0.4124
PSO	31420	0.6467	0.4730
SHO	100000	0.6667	0.5149
SSA	91220	0.5733	0.3775
DA	50700	0.6133	0.4347

6. Conclusion

In this paper, an efficient hybrid optimization approach based on adaptive elitist Differential Evolution and Spherical Quadratic Steepest Descent was proposed and then applied for clustering problem. The new method benefits from the aeDE's global explorative manner in the initial steps and from the SQSD's locally effective exploitative manner in the later steps to improve the aeDE performance and reduce the computational cost significantly. The behavior of HaeDE is examined by a simple function, and its performance is evaluated on a set of 32 benchmark functions as well as an application in clustering problem. In summary, the HaeDE can be considered as a competitive optimization algorithm and can be utilized effectively in clustering and other applications in future. In addition to the mentioned advantages, the proposed method has a few disadvantages, e.g., it wastes the computational resource in case of unimodal functions, a parallel processing strategy for the proposed method is not considered, and the application of the proposed method to the clustering still requires a given number of clusters. They are also interesting future research directions for a wide range of researchers.

Data Availability

The data used to support the findings of this study are included within the article or are made publicly available to the research community at https://archive.ics.uci.edu/ml/datasets/iris.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[2] C. García-Martínez, M. Lozano, F. Herrera, D. Molina, and A. M. Molina, "Global and local real-coded genetic algorithms based on parent-centric crossover operators," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1088–1113, 2008.

[3] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[4] J. H. Holland, "Genetic algorithms and the optimal allocation of trials," *SIAM Journal on Computing*, vol. 2, no. 2, pp. 88–105, 1973.

[5] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing*, vol. 11, no. 2, pp. 1679–1696, 2011.

[6] N. E. Nawa, T. Hashiyama, T. Furuhashi, and Y. Uchikawa, "Fuzzy logic controllers generated by pseudo-bacterial genetic algorithm with adaptive operator," in *Proceedings of International Conference on Neural Networks*, pp. 2408–2413, IEEE, Houston, TX, USA, June 1997.

[7] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.

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

[9] F. Fausto, E. Cuevas, A. Valdivia, and A. González, "A global optimization algorithm inspired in the behavior of selfish herds," *Biosystems*, vol. 160, pp. 39–55, 2017.

[10] D. Karaboga, *An Idea Based on Honey Bee Swarm for Numerical Optimization*, Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, Kayseri, Turkey, 2005.

[11] J. Kenndy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks*, IEEE Press, Perth, Australia, December 1995.

[12] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

[13] S. Mirjalili, "Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," *Neural Computing and Applications*, vol. 27, no. 4, pp. 1053–1073, 2015.

[14] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili et al., "Salp Swarm Algorithm: a bio-inspired optimizer for engineering design problems," *Advances in Engineering Software*, vol. 114, pp. 163–191, 2017.

[15] X. S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proceedings of 2009 World Congress on Nature Biologically Inspired Computing (NaBIC)*, pp. 210–214, Coimbatore, India, December 2009.

[16] M. Zlochin, M. Birattari, N. Meuleau, and M. Dorigo, "Model-based search for combinatorial optimization: a critical survey," *Annals of Operations Research*, vol. 131, no. 1–4, pp. 373–395, 2004.

[17] V. Ho-Huu, T. Nguyen-Thoi, T. Vo-Duy, and T. Nguyen-Trang, "An adaptive elitist differential evolution for optimization of truss structures with discrete design variables," *Computers and Structures*, vol. 165, pp. 59–75, 2016.

[18] J. A. Snyman and A. M. Hay, "The spherical quadratic steepest descent (SQSD) method for unconstrained minimization with no explicit line searches," *Computers and Mathematics with Applications*, vol. 42, no. 1-2, pp. 169–178, 2001.

[19] J. Snyman, *Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms*, Springer Science and Business Media, Berlin, Germany, 2005.

[20] N. Padhye, P. Bhardawaj, and K. Deb, "Improving differential evolution through a unified approach," *Journal of Global Optimization*, vol. 55, no. 4, pp. 771–799, 2012.

[21] S. Rahnamayan and G. G. Wang, "Center-based initialization for large-scale black-box problems," in *Proceedings of 8th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases*, pp. 531–541, World Scientific and Engineering Academy and Society (WSEAS), Cambridge, UK, February 2009.

[22] M. S. Saad, H. Jamaluddin, and I. Z. M. Darus, "PID controller tuning using evolutionary algorithms," *WSEAS Transactions on Systems and Control*, vol. 7, pp. 139–149, 2012.

[23] R. Storn, "On the usage of differential evolution for function optimization," in *Proceedings of 1996 Biennial Conference of the North American Fuzzy Information Processing Society, NAFIPS*, pp. 519–523, Berkeley, California, USA, June 1996.

[24] J. Lampinen, "Differential evolution-new naturally parallel approach for engineering design optimization," in *Developments in Computational Mechanics with High Performance Computing*, , pp. 217–228, 1999.

[25] K. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, Springer Science and Business Media, Berlin, Germany, 2006.

[26] S. Mahdavi, S. Rahnamayan, and K. Deb, "Center-based initialization of cooperative co-evolutionary algorithm for large-scale optimization," in *Proceedings of 2016 IEEE Congress on Evolutionary Computation (CEC)*, pp. 3557–3565, Vancouver, BC, Canada, July 2016.

[27] F. Neri and V. Tirronen, "Scale factor local search in differential evolution," *Memetic Computing*, vol. 1, no. 2, pp. 153–171, 2009.

[28] M. Weber, F. Neri, and V. Tirronen, "A study on scale factor/crossover interaction in distributed differential evolution," *Artificial Intelligence Review*, vol. 39, no. 3, pp. 195–224, 2011.

[29] D. Zaharie, "Parameter adaptation in differential evolution by controlling the population diversity," in *Proceedings of the International Workshop on Symbolic and Numeric Algorithms for Scientific Computing*, pp. 385–397, Timisoara, Romania, October, 2002.

[30] J. Zhang and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Trans Evol Comput*, vol. 13, no. 5, pp. 945–958, 2009.

[31] D. Karaboga and B. Akay, "A comparative study of Artificial Bee Colony algorithm," *Applied Mathematics and Computation*, vol. 214, no. 1, pp. 108–132, 2009.

[32] B. Doğan and T. Ölmez, "A new metaheuristic for numerical function optimization: vortex Search algorithm," *Information Sciences*, vol. 293, pp. 125–145, 2015.

[33] L. E. Agustín-Blas, S. Salcedo-Sanz, S. Jiménez-Fernández et al., "A new grouping genetic algorithm for clustering problems," *Expert Systems with Applications*, vol. 39, pp. 9695–9703, 2012.

[34] P. Lingras and X. Huang, "Statistical, evolutionary, and neurocomputing clustering techniques: cluster-based vs object-based approaches," *Artificial Intelligence Review*, vol. 23, no. 1, pp. 3–29, 2005.

[35] T. Nguyentrang and T. Vovan, "Fuzzy clustering of probability density functions," *Journal of Applied Statistics*, vol. 44, no. 4, pp. 583–601, 2016.

[36] T. Warren Liao, "Clustering of time series data-a survey," *Pattern Recognition*, vol. 38, no. 11, pp. 1857–1874, 2005.

[37] R. Xu and D. WunschII, "Survey of clustering algorithms," *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 645–678, 2005.

[38] V. M. Gómez-Muñoz and M. A. Porta-Gándara, "Local wind patterns for modeling renewable energy systems by means of cluster analysis techniques," *Renewable Energy*, vol. 25, no. 2, pp. 171–182, 2002.

[39] S. Mitra and H. Banka, "Multi-objective evolutionary biclustering of gene expression data," *Pattern Recognition*, vol. 39, no. 12, pp. 2464–2477, 2006.

[40] P. Scheunders, "A genetic c-Means clustering algorithm applied to color image quantization," *Pattern Recognition*, vol. 30, no. 6, pp. 859–866, 1997.

[41] T. Vo-Van, T. Nguyen-Thoi, T. Vo-Duy, V. Ho-Huu, and T. Nguyen-Trang, "Modified genetic algorithm-based clustering for probability density functions," *Journal of Statistical Computation and Simulation*, vol. 87, no. 10, pp. 1964–1979, 2017.

[42] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281–297, Oakland, CA, USA, 1967.

[43] X. L. Xie and G. Beni, "A validity measure for fuzzy clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 8, pp. 841–847, 1991.

[44] J. C. Dunn, "Well-separated clusters and optimal fuzzy partitions," *Journal of Cybernetics*, vol. 4, no. 1, pp. 95–104, 1974.

[45] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 1, no. 2, pp. 224–227, 1979.

[46] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

[47] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 2012.

[48] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, vol. 66, no. 336, pp. 846–850, 1971.