Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 18(5): 6034–6065.
DOI: 10.3934/mbe.2021302
Received: 11 May 2021
Accepted: 27 June 2021
Published: 06 July 2021

Research article

Solving nonlinear equation systems via clustering-based adaptive speciation differential evolution

Qishuo Pang[1], **Xianyan Mi**[2,3,*], **Jixuan Sun**[4] **and Huayong Qin**[5]

[1] College of Mechanical, Naval Architecture and Ocean Engineering, Beibu Gulf University, Qinzhou 535011, China

[2] Beibu Gulf Ocean Development Research Center, Beibu Gulf University, Qinzhou 535000, China

[3] College of Economics and Management, Beibu Gulf University, Qinzhou 535000, China

[4] College of Ceramics and Design, Beibu Gulf University, Qinzhou 535000, China

[5] Center of Internet and Educational Technology, Beibu Gulf University, Qinzhou 535000, China

* **Correspondence:** Email: 563292753@qq.com.

Abstract: In numerical computation, locating multiple roots of nonlinear equations (NESs) in a single run is a challenging work. In order to solve the problem of population grouping and parameters settings during the evolutionary, a clustering-based adaptive speciation differential evolution, referred to as CASDE, is presented to deal with NESs. CASDE offers three advantages: 1) the clustering with dynamic clustering sizes is used to set clustering sizes for different problems; 2) adaptive parameter control at the niche level is proposed to enhance the search ability and efficiency; 3) re-initialization mechanism motivates the algorithm to search new roots and saves computing resources. To evaluate the performance of CASDE, we select 30 problems with different features as test suite. Experimental results indicate that the speciation clustering with dynamic clustering sizes, niche adaptive parameter control, and re-initialization mechanism when combined together in a synergistic manner can improve the ability to find multiple roots in a single run. Additionally, our method is also compared with other state-of-the-art methods, which is capable of obtaining better results in terms of peak ratio and success rate. Finally, two practical mechanical problems are used to verify the performance of CASDE, and it also demonstrates superior results.

Keywords: nonlinear equation systems; dynamic clustering sizes; niche adaptive parameter control; re-initialization mechanism; differential evolution

1. Introduction

Many real-world applications can be transformed into nonlinear equations (NESs), such as physics [1], engineerings [2, 3], economics [4], and so on. Generally, a NES contains multiple roots. Each root is equally important because it can provide multiple selections to the decision makers so that they can make a better decision [5]. In recent years, solving NESs have received widespread attention from researchers. However, it may cause the great challenge in mathematical field, especially to locate multiple roots in a single run.

Evolutionary algorithms (EAs) draw lessons from the evolution of biological operations in nature [6]. It includes the basic operations of population initialization, crossover mutation operator, retention mechanism, and so on. Among them, differential evolution (DE) [7] is a famous optimization technique and a versatile function optimizer. Due to its features of easy-to-implement and robust adaptability, DE has been widely applied for many optimization problems [8, 9]. More specifically, Li et al. presented an enhanced adaptive DE algorithm to extract the parameters of photovoltaic models [10]. Mohanmed et al. designed a novel mutation strategy to enhance SHADE and LSHADE algorithm to solve the global numerical optimization [11]. Pierezan et al. proposed a modified self-adaptive differential evolution to deal with the static force capability optimization of humanoids robots [12]. Santos Coelho et al. presented a self-adaptive chaotic differential evolution algorithm using gamma distribution to solve unconstrained global optimization problem [13]. Li et al. hybrid differential evolution algorithm with modified CoDE and JADE to improve the performance of solving the global optimization problem [14]. Civicioglu et al. adopted bezier search differential evolution algorithm to solve the numerical function optimization [15]. Zhao et al. proposed a collaborative LSHADE algorithm with comprehensive learning mechanism to slove the non-separable optimization problem and obtained the competitive results [16].

In recent years, DE is often used to solve NESs because it is insensitive to the characteristics of NESs, such as non-convexity and discontinuity, it has been applied to solve NESs [17]. However, DE encounters following two dilemma: 1) due to the lack of diversity preserving mechanism, it hardly locates multiple roots in a single run; 2) although some DE variants can obtain multiple roots, they have the problems of parameter settings, such as cluster size [18], repulsion radius [19].

To effectively solve NESs, it is important to maintain the population diversity during the evolutionary process. Clustering is perceived to be an effective methods, which can pratition the whole population into different species [20]. Thus, some clustering-based methods have been developed to find the roots of NESs [18, 21]. It requires to give the number of cluster in advance. Too small the number of clusters may lose some roots. In contrast, too large cluster number may obtain the roots with low accuracy since the algorithm cannot make full use of computational resources to exploit in each cluster. However, it is difficult to specify the appropriate number of clusters for different NESs.

Parameter control in evolutionary computation plays an important role to the robustness of the algorithm [22]. Recently, some adaptive or self-adaptive parameter control methods [23–25] have been proposed to dynamically control the parameters according to different fitness landscapes of optimization problems. These methods can significantly improve the search ability of DE, but most of them focus on global optimum. Thus, how to develop a self-adaptive strategy suited for NESs is still a problem to be solved.

After clustering, each subpopulation might gradually converge to a narrow range of space during the evolution process. If an individual fitness less than the threshold, it is considered as a candidate solution. By this stage, continuous optimization in this subpoulation prefers the exploitation, which is able to improve the quality of root. However, it is not conducive to the population diversity and wastes more computational resource to seek the same root. Therefore, it is an urgent issue to deal with the species contained the found root.

Based on the above considerations, we combine the species clustering with dynamic cluster sizes, niche adaptive parameter control, and re-initialization mechanism to locate multiple roots of NESs. The proposed method is referred as a clustering-based adaptive speciation differential evolution (CASDE). In CASDE, species clustering with dynamic cluster sizes can alleviate the trivial task to set the cluster number. Moreover, the adaptive parameter control is employed to dynamically adjust at the niche level, thereby improving the search efficiency of the algorithm. To verify the performance of our method, we select 30 NESs from the literature [26] as test suite. Experimental results demonstrate that our method obtains highly competitive results compared with other state-of-the-art methods.

The major contributions of this paper are summarized as follows:

- The species clustering with dynamic cluster sizes, niche adaptive parameter control, and re-initialization technique are combined together in a synergistic manner can greatly enhance the problem-solving capability. Among them, the species clustering with dynamic cluster sizes-maintaining the population diversity; niche adaptive parameter control-enhancing the exploitation performance in each species and avoiding trivial task to set parameters; and re-initiation technique-seeking the new roots with an reasonable using computational resource reasonably.
- The niche parameter adaptive method is proposed to ensure that the parameters can be changed according to different problem landscapes, so as to improve the efficiency of the algorithm.
- The effectiveness of the three different parts of components has been experimentally verified. The results show that these three parts combined together in a synergistic manner can greatly enhance the performance for solving NESs.
- To further evaluate the performance of CASDE, two cases of motor system are selected to measure the performance of the proposed approach. The results have been confirmed the effectiveness of CASDE on real-world problems.

The rest of this paper is summarized as follows. Section 2 introduces background knowledge about the transformed optimization problem, different evolution. Section 3 reviews the related work for solving NESs. In Section 4, the proposed CASDE is described in detail. The experimental results and discussion are respectively given in Section 5, follow by the discussion in Section 6. In Section 7, two practical problem are used to test the performance of the algorithm. Finally, Section 8 concludes this paper.

2. Background

2.1. Problem statement

Generally, a NES can be expressed as follows:

$$\mathbf{E}(\mathbf{x}) = \begin{cases} e_1(x_1, x_2, \ldots, x_D) & = & 0 \\ e_2(x_1, x_2, \ldots, x_D) & = & 0 \\ & \vdots & \\ e_n(x_1, x_2, \ldots, x_D) & = & 0 \end{cases} \tag{2.1}$$

where n is the number of equations, $\mathbf{x} = (x_1, x_2, \ldots, x_D)$ denotes a decision vector, $\mathbf{x} \in \mathbb{S}$, and $\mathbb{S} \subseteq \mathbb{R}^D$ is the search space. Generally,

$$\mathbb{S} = [\underline{x}_i, \overline{x}_i]^m$$

where $i = 1, \cdots, D$, $\underline{x}_i$ and $\overline{x}_i$ are the lower and upper bound of x_i, respectively.

Before solving a NES with optimization algorithm, it is commonly transformed into a single-objective optimization problem:

$$\text{minimize} \quad f(\mathbf{x}) = \sum_{i=1}^{n} e_i^2(\mathbf{x}) \tag{2.2}$$

Subsequently, solving NES is equivalent to find the global optimal of the transformed optimization problem in Eq (2.2).

2.2. *Differential evolution*

Differential evolution (DE) adopts three operators, including mutation, crossover, and selection, to deal with the population during the search process. Generally, a population contains NP real-valued vectors: $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{NP}\}$. NP is the population size.

2.2.1. Mutation

Mutation operator is used to generate a mutant vector $\mathbf{v}_i$ according to the parent population. Two well-known mutation strategies are shown below:

- ”DE/rand/1”

$$\mathbf{v}_i = \mathbf{x}_{r1} + F \cdot (\mathbf{x}_{r2} - \mathbf{x}_{r3}) \tag{2.3}$$

- ”DE/best/1”

$$\mathbf{v}_i = \mathbf{x}_{best} + F \cdot (\mathbf{x}_{r2} - \mathbf{x}_{r3}) \tag{2.4}$$

r_1, r_2 and r_3 respectively represent different random indices selected from the population, and they are distinct from the base index i. F is a scale factor that controls the size of difference vector. $\mathbf{x}_{best}$ is the best individual in current population.

2.2.2. Crossover

The crossover operator generates the trial vectors by recomposing the current vector and the mutant vector. The trial vector $\mathbf{x}'_i(j)$ is generated as follow:

$$\mathbf{x}'_i(j) = \begin{cases} \mathbf{v}_i(j), \text{if rand}_j(0,1) \leq CR_{i,j} \quad \text{or} \quad j = j_{rand} \\ \mathbf{x}_i(j), \text{otherwise} \end{cases} \tag{2.5}$$

where $CR_{i,j} \in (0,1)$ is the crossover rate; $\text{rand}_j(0,1)$ is a random value within $[0,1]$; $j = \{1,...,D\}$; $j_{rand} \in \{1,2,...,D\}$ represents a random index.

2.2.3. Selection

In original DE algorithm, greedy selection operator is employed to select the individual with better fitness value for a minimization problem. If the trial vector $\mathbf{x}'_i$ is better than $\mathbf{x}_i$, $\mathbf{x}_i$ is set to $\mathbf{x}'_i$; otherwise, $\mathbf{x}_i$ keeps unchanged.

$$\mathbf{x}_i = \begin{cases} \mathbf{x}'_i, \text{if } f(\mathbf{x}'_i) \leq f(\mathbf{x}_i) \\ \mathbf{x}_i, \text{otherwise} \end{cases} \tag{2.6}$$

where $f(\cdot)$ is the objective function to be minimized.

3. Algorithms for NES problems

Some stochastic approaches were proposed to solve NESs, which consist of three categories: multiobjective-optimization-based ones, single-objective-optimization-based ones, and constraint-optimization-based ones.

3.1. Multiobjective optimization based-methods

Multiobjective optimization can obtain a group of Pareto optimal solutions [27, 28], which is similar to locate different roots of NESs. Therefore, more attention has been paid to the use of multiobjective optimization to solve NESs. In [29], a NES was first transformed into a multiobjective optimization problem, and then solve the optimization problem via a evolution algorithm. This kind of transformation technique ensures that multiple roots can be found in a single run. However, it may cause the curse of dimensionality if the number of equation is too large. To solve this problem, In [30], a bi-objective transformation technique was proposed to transform a NES into multiobjective optimization problems with two objectives. However, since only one decision variable is utilized to design the location function, it may lose several roots. For this purpose, Gong et al. [31] presented a weighted bi-objective transformation technique (A-WeB) for NESs. In [32], Naidu and Ojha employed a hybrid cooperative multiobjective optimization IWO to solve NESs, where multiple populations are used to deal with multiple objectives.

3.2. Single-objective optimization based-methods

In general, a NES is usually transformed into a single-objective optimization problem, as shown in Eq (2.2). Two common methods are used to solve this problem: clustering-based methods and repulsion-based methods.

3.2.1. Clustering-based methods

Clustering [20] can divide the population into different subpopulations, thus enabling the algorithm to search for more than one promising area. Clustering can maintain the diversity of population, so several researchers put forward clustering-based methods to locate multiple roots of NESs. In [18], the clustering technique was employed to separate the estimated locations of solutions, and then invasive weed optimization was used to calculate the exact solution in each cluster. In [33], Fuzzy Clustering Means combines Luus-Jaakola random search and the Nelder-Mead simplex method to solve NESs. In [21], Multistart and Minfinder methods based on clustering were applied for locating multiple roots of NESs.

3.2.2. Repulsion-based methods

The repulsion techniques can generate a repulsive regions around the obtained roots, which can increase the variety of population. Based on the repulsion techniques, several methods have been developed to solve NESs. In [34], repulsion technique combined with simulated annealing (SA) to compute critical points in binary systems. Henderson et al. [35] designed a combination of continuous SA and repulsion technique to locate multiple roots of double retrograde vaporization. In [18], a two-phase root-finder was developed to find the roots of NESs, in which invasive weed optimization located the exact roots while repulsion technique was used to preserve the population diversity. In [36], a biased random-key genetic algorithm (BRKGA) was carried out many times to detect the roots. In [19], the improved harmony search algorithm combined with the repulsion methods to solve NESs. In [26], a new approach consisting of the repulsion technique, adaptive parameter control, and diversity preserving mechanism, named RADE, was designed. Further, In [37], a general framework based on the dynamic repulsion technique and evolutionary algorithms (DR-JADE) is presented.

3.3. Constraint-optimization-based methods

The third category is the constrained optimization transformation approach. It translates a NES into a constrained optimization problem:

$$\begin{cases} \min & \sum_{i=1}^{n} |e_i(\mathbf{x})| \\ \text{subject to} & e_i(\mathbf{x}) \geq 0, i = 1, 2, ..., n \end{cases} \tag{3.1}$$

Based on this translated optimization problem, Kuri-Morales [38] developed a penalty function to handle constraints and used a genetic algorithm (GA) to locate the root. Pourrajabian et al. [39] combined augmented lagrangian function with GA to find the optimal solution.

4. Our approach

4.1. Motivation

Generally, the number of clustering should be given before the algorithm runs. However, it is difficult to set appropriate clustering number for NESs. Besides, after clustering, there are different subpopulations in the search space. Different subpopulations represent multiple promising regions that may have roots. Thus, how to improve the search ability in each subpopulation also needs further research. Moreover, as the search proceeds, each subpopulation will converge to a narrow range as the

Algorithm 1: Speciation clustering with dynamic cluster sizes

```
Input: population $P$, cluster size set $C$
Output: a set of species
Sort $P$ in ascending order according to fitness value;
while $P$ is not empty do
    Random select a cluster size $M$ from $C$;
    Select the best individual in $P$ as a new seed;
    Find $M - 1$ individuals closest to the species seed and combine them as a species;
    Remove these $M$ individuals from $P$
end
```

number of iterations increases. If a root is found in the subpopulation, continuous optimization of the subpopulation will result in loss of diversity and waste of computing resources. In addition, several researchers have adopted improved DE algorithm to solve NESs, which has gained extensive attention and improved the performance of problem solving [26, 37]. For this reason, this paper still focuses on enhancing the DE algorithm by integrating other techniques in order to obtain satisfactory results.

Based on the above considerations, a re-initialization clustering-based adaptive differential evolution, named CASDE, is presented to solve NESs. In CASDE, a dynamical cluster sizing technique is employed to solve the problem that the number of clusters is difficult to set. Meanwhile, niche adaptive parameter setting is applied to improve the search ability in each subpopulation and avoid the trivial task of parameter settings. Moreover, the re-initialization mechanism will be triggered if a root has found in a subpopulation. For one thing, it can preserve the population diversity; for another, it is a benefit for exploration ability of the search algorithm and increases the probability to find new roots in other promising regions.

4.2. Dynamic clustering size

In [40], a speciation clustering was proposed to divide the population into different subpopulations. To reduce the sensitivity of the cluster size, dynamic cluster sizing technique [41] was introduced into speciation clustering. The effectiveness of this simple scheme was verified by experiments.

The process of dynamic clustering size (DCS) is outlined in Algorithm 1. First, the population is sorted according to fitness value in ascending order. Second, a random integer M is selected from the cluster size set C. Based on such integer, $M - 1$ individuals that is close to species seed are combined with the species seed to form a species. Finally, the algorithm divides the whole population into a number of species with M individuals. It is noted that M is a random integer selected from C.

4.3. Niche adaptive parameter control

After clustering, different subpopulations contain different promising regions, which have different landscape characteristics in the search space. To improve the search ability and avoid the trivial task of parameter settings in the subpopulations , a niche adaptive differential evolution is proposed in this section.

At each generation, the crossover rate $CR_{i,j}$ of each individual $\mathbf{x}_i$ in cluster j is independently generated according to a normal distribution of mean CR_j and standard deviation 0.1

$$CR_{i,j} = randn(\mu_{CR}, 0.1) \tag{4.1}$$

and truncated to [0,1], μ_{CR} is updated as follows:

$$\mu_{CR} = (1 - c) \cdot \mu_{CR} + c \cdot \text{mean}_L(S_{CR_j}) \tag{4.2}$$

where c is constant number between 0 and 1; S_{CR_j} is the set of the crossover rates $CR_{i,j}$ in each species. $\text{mean}_L(.)$ is the Lehmer mean

$$\text{mean}_L(S_{CR_j}) = \frac{\sum S_{CR_j}^2}{\sum S_{CR_j}} \tag{4.3}$$

Similarly, the mutation factor $F_{i,j}$ of each individual x_i in species j is independently generated according to a Cauchy distribution.

$$F_{i,j} = randc(\mu_F, 0.1) \tag{4.4}$$

It is regenerated if $F_{i,j} \le 0$ or truncated to be 1 if $F_{i,j} \ge 1$. The parameter μ_F updates as follow:

$$\mu_F = (1 - c) \cdot \mu_F + c \cdot \text{mean}_A(S_{F_j}) \tag{4.5}$$

where $\text{mean}_A(.)$ is the arithmetic mean; and S_{F_j} is the set of mutation factor $F_{i,j}$ at each species.

Comparison with JADE [24], we modify the parameter adaptive method to some extent. S_{F_j} does update via arithmetic mean whereas S_{CR_j} is used Lehmer mean to revise. The reasons are two-fold: i) the adaptation S_{F_j} put a greater emphasis on normal mutation factor by using the arithmetic mean instead of a Lehmer mean. The arithmetic mean is helpful to propagate average mutation factors, which improve the exploitation ability in each species; ii) to improve population diversity in the species and avoid trapping in local optima, Lehmer mean is used to update S_{CR_j}.

4.4. Reinitialization mechanism

As the algorithm proceeding, the candidate solution can be found in the subpopulation. Continuous optimization in such subpopulation may pays more attention to exploitation rather than exploration. It may lead to loss of population diversity and waste of computational resource. Therefore, the reinitialization mechanism is used to solve this problem.

If a subpopulation locates a root, it is considered as convergence during the run. The root is stored into a archive $\mathcal{A}$. Subsequently, all of the individuals in the subpopulation will reinitialize for maintaining population diversity. Additionally, F and CR of each individual are respectively set to 0.5 and 0.9.

It is worth mentioning that our approach may find the same root during the run. Thus, a method to update the archive is employed to avoid encountering this dilemma. Algorithm 2 outlines the process of updating the archive. In Algorithm 2, $\mathbf{x}^*$ is one of the root in $\mathcal{A}$, and ϵ is a small fixed value to avoid storing the same root in $\mathcal{A}$. In this paper, ϵ is set to 0.01.

In reinitialization mechanism, we initialize the entire subpopulation instead of the found root, which has two advantages. On the one hand, it can use computational resource efficiently and benefit population diversity. On the other hand, to some extent, this mechanism prevents the remaining individuals in the subpopulation from continuing to locate the same root.

Algorithm 2: Archive updating

```
Input: Solution x and $\epsilon > 0$
Output: The updated archive $\mathcal{A}$
if $s_{\mathcal{A}} = 0$ then                                   // The archive is empty
    if $f(\mathbf{x}) < \tau$ then
        $\mathcal{A} = \mathcal{A} \cup \mathbf{x}$;
        $s_{\mathcal{A}} = s_{\mathcal{A}} + 1$;
    end
end
else
    if $f(\mathbf{x}) < \tau$ then
        Find the closest root $\mathbf{x}^*$ to $\mathbf{x}$ in $\mathcal{A}$
        if $\| \mathbf{x} - \mathbf{x}^* \| < \epsilon$ and $f(\mathbf{x}^*) < f(\mathbf{x})$ then    // Update the found root
            $\mathbf{x} = \mathbf{x}^*$;
        end
        else if $\| \mathbf{x} - \mathbf{x}^* \| > \epsilon$ then       // A new root is found
            $\mathcal{A} = \mathcal{A} \cup \mathbf{x}$;
            $s_{\mathcal{A}} = s_{\mathcal{A}} + 1$;
        end
    end
end
```

4.5. The proposed framework: CASDE

The framework of CASDE is outlined in Algorithm 3. One iteration in CASDE includes following steps:

Step 1: Dividing population (in line 7). Algorithm 1 partitions the population into multiple subpopulations.

Step 2: Generating new individuals (in lines 9–20). Each individual generates new offspring based on three operations: mutation, crossover, and selection.

Step 3: Updating the information (in line 21). μ_F and μ_{CR} in j-th species are modified through Eqs (4.5) and (4.2).

Step 4: Updating the archive (in line 24). If a root is located, it will be stored in the archive.

Step 5: Reinitializating the subpopulation (in lines 25, 26). All individuals in such subpopulation are reinitialized; $F_{i,j}$ and $CR_{i,j}$ of each individual are also respectively set to 0.5 and 0.9.

The above iteration is repeated until the termination criterion was met.

It is worth mentioning that a hybrid mutation strategy (”DE/rand/1” and ”DE/best/1”) is used to generate the mutation vectors (in lines 12–15). The reason is that the hybrid strategy can balance the ability of exploration and exploitation to some extent and improve the search ability of the algorithm.

Algorithm 3: The framework of CASDE

```
Input: Control parameters: $NP$, $NFE$, $NFEs_{\max}$
Output: The final archive $\mathcal{A}$
Set $NFE = 0$ and the archive $\mathcal{A} = \emptyset$;
Randomly generate the population $P$;
$F$ and $CR$ of each individual in $P$ are set to 0.5 and 0.9 ;
Calculate the fitness value of $\mathbf{x}$ via Eq (2.2);
$NFE = NFE + NP$;
while $NFE < NFEs_{\max}$ do
    Partition the whole population into different species via Algorithm 1
    for $j$-th species do
        Implement Eqs (4.4) and (4.1) to produce $F_{i,j}$ and $CR_{i,j}$.
        for each individual in $j$-th species do
            Select $r1, r2, r3$ from the $j$-th species
            if $rand < 0.5$ then
                Generate mutation vector using Eq (2.3)
            else
                Generate mutation vector using Eq (2.4)
            Produce the trial vector $\mathbf{u}'_{i,j}$ via Eq (2.5).
            Evaluate offspring $\mathbf{u}'_{i,j}$ using Eq (2.2).
            if $f(\mathbf{u}'_{i,j}) < f(\mathbf{u}_{i,j})$ then
                $\mathbf{u}_{i,j} = \mathbf{u}'_{i,j}$
                Respectively update $F_{i,j}$ and $CR_{i,j}$ in $S_{F_j}$ and $S_{CR_j}$
            Update $\mu_F$ and $\mu_{CR}$ via Eqs (4.5) and (4.2);
        end
        Find the minimal fitness value $f(\mathbf{min})$ in $j$-th species
        if $f(\mathbf{min}) < \tau$ then
            Record the corresponding individual in the archive via Algorithm 2
            Re-initialize the individuals in $j$-th species
            Reset the $F_{i,j}$ and $CR_{i,j}$ to 0.5 and 0.9 in $j$-th species
    end
    $NFE = NFE + NP$;
end
```

5. Experiment results and analysis

In this section, we mainly focuses on experimental results and analysis, including the impact of different parts in CASDE, comparing CASDE with state-of-the-art methods.

5.1. Test problems

To evaluate the performance of different methods, frequently-used NESs (F01–F30) benchmark problems are selected as a test suite. They have different characteristics, and some derive from real-world applications, such as multiple steady states problem (F08) [42], robot kinematics problem (F17) [43], and molecular conformation (F23) [44]. The test problems are briefly introduced in Table 1.

Table 1. Brief information of the test problems, where D is the number of decision variables, LE is the number of linear equations, NE is the number of nonlinear equations, NoR is the number of the known roots, and $NFEs_{max}$ is the maximum number of fitness evaluations.

Prob.	D	LE	NE	NoR	$NFEs_{max}$
F01	20	0	2	2	50,000
F02	2	1	1	11	50,000
F03	2	0	2	15	50,000
F04	2	0	0	13	50,000
F05	10	0	10	1	50,000
F06	2	1	1	8	50,000
F07	2	0	2	2	50,000
F08	2	0	2	7	50,000
F09	5	4	1	3	100,000
F10	3	0	3	2	50,000
F11	2	0	2	4	50,000
F12	2	0	2	10	50,000
F13	3	0	3	12	50,000
F14	2	0	2	9	50,000
F15	2	0	2	2	50,000
F16	2	0	2	13	50,000
F17	8	1	7	16	100,000
F18	2	0	2	6	50,000
F19	20	19	1	2	200,000
F20	3	0	3	7	50,000
F21	2	0	2	4	50,000
F22	2	0	2	6	50,000
F23	3	0	3	16	500,000
F24	3	0	3	8	100,000
F25	3	0	3	2	50,000
F26	2	0	2	2	50,000
F27	2	0	2	3	50,000
F28	2	0	2	2	50,000
F29	3	0	3	5	50,000
F30	2	0	2	4	50,000

5.2. Performance metrics for NESs

To evaluate the effectiveness of algorithms, two performance Metrics in [26, 37] : (Root ratio (RR) and Success Rate (SR)), are employed in this paper.

- Root ratio (*RR*): It is the ratio of found roots number to total roots number over multiple runs within $NFEs_{max}$:

$$RR = \frac{\sum_{i=1}^{N_r} N_{f,i}}{NoR \cdot N_r} \tag{5.1}$$

where N_r is the number of runs; $N_{f,i}$ is the found roots number in the i-th run; NoR is the total roots number of a NES.

- Success rate (SR): It is the ratio of successful runs[*] to total runs :

$$SR = \frac{N_{r,s}}{N_r} \tag{5.2}$$

where $N_{r,s}$ is the number of successful runs.

5.3. Influence of different parts in CASDE

CASDE contains three parts, i.e., dynamic clustering sizes, niche adaptive parameter control, and the re-initialization mechanism. This subsection mainly dedicates to discuss the impact of different parts of CASDE on performance of solving NESs.

1) CASDE. The dynamic clustering sizes, niche adaptive parameter control, and the re-initialization mechanism were combined.

2) DCS-DE. It is the algorithm that DE combines with dynamic clustering sizes.

3) CASDE/DA. The re-initialization mechanism was removed from CASDE;

4) CASDE/DR. The niche adaptive parameter control was removed from CASDE. F and CR were respectively set to 0.5 and 0.9;

5) CASDE/AR. The dynamic clustering sizes were not used in CASDE. The clustering size were set to 5.

The detailed experiment result in terms of *RR* and SR is shown in Table A1. It is obvious that CASDE obtained the best average *RR* value, i.e., **0.9951** and the best average SR value, i.e., **0.9556**. Additionally, CASDE successfully solves 26 out of 30 NESs over 30 independent runs. In contrast, CASDE/AR, CASDE/DR, CASDE/DA and DCS-DE. successfully solve 23, 20, 15, 13 NESs over 30 independent runs, respectively.

The statistical test results acquired by the multiple-problem Wilcoxon's test are showed in Table 3. In addition, the ranking results obtained from the Friedman's test are shown in Table 2. From Table 3, CASDE consistently offers significantly better results than CASDE/DR, CASDE/DA, and DCS-DE due to the fact that p-values are less than 0.05 in all the cases. Additionally, CASDE also obtains the best ranking as shown in Table 2. In what follows, we try to analyze the influence of different parts of CASDE on the performance of solving NESs.

- DCS-DE: from Table A1, DCS-DE can successfully solve 13 out of 30 NESs, i.e., F01, F05, F06, F09-F11, F18, F20, F21, F26-F30. However, one feature of them is that they contain a small number of the known roots. For example, the known roots of this kind of NESs are no more than 8. Thus, DCS-DE is more suitable for solving these NESs.

[*]A successful run is considered as a run where all known optima of a NES are found.

- CASDE/DA: The combination of dynamic clustering size and niche adaptive parameter control can improve the performance of the algorithm to some extent. As shown in Table A1, *RR* and *SR* values obtained by CASDE/DA were higher than DCS-DE.
- CASDE/DR: From Table A1, the re-initialization method can improve the performance of the algorithm. For example, *RR* and *SR* values obtained by CASDE/DR were 0.9365 and 0.8156, respectively, significantly higher than those obtained by DCS-DE.
- CASDE/AR: Similarly, the combination of niche adaptive parameter control and re-initialization mechanism can also improve the performance of the algorithm.
- CASDE: It obtains the best *RR* and *SR* values. The reasons are as follows: 1) dynamic clustering size can reduce the sensitivity of clustering number and maintain population diversity; 2) niche adaptive parameter control reduces the tedious task of parameter settings and improves the search ability of the algorithm; 3) the re-initialization mechanism can improve the population diversity and detect the new roots.

Table 2. Average rankings of CASDE, CASDE/CA, CASDE/C, AND DCS-DE obtained by the Friedman test for both *RR* and *SR* criteria.

Algorithm	Ranking (*RR*)	Ranking (*SR*)
CASDE	**2.3667**	**2.3500**
CASDE/AR	2.5500	2.5333
CASDE/DR	3.0667	3.0333
CASDE/DA	3.3167	3.3667
DCS-DE	3.7000	3.7167

Table 3. Results obtained by the wilcoxon test for algorithm CASDE in terms of *RR* and *SR* compared with CASDE/AR, CASDE/DR, CASDE/DA and DCS-DE.

VS	*RR*			*SR*		
	R^+	R^-	*p*-value	R^+	R^-	*p*-value
CASDE/AR	265.5	199.5	*4.90E-01*	260.5	204.5	*5.57E-01*
CASDE/DR	337.0	128.0	**3.07E-02**	345.0	120.0	**1.42E-02**
CASDE/DA	378.5	86.5	**1.36E-03**	378.5	86.5	**1.28E-03**
DCS-DE	393.0	72.0	**4.99E-04**	394.5	70.5	**2.93E-04**

5.4. Comparison with other state-of-the-art methods

CASDE is compared with the following state-of-the-art algorithms:

- Decomposition-based differential evolution with reinitialization: DDE/R.
- A weighted biobjective transformation technique for NESs: A-WeB [45].
- Repulsion-based adaptive differential evolution: RADE [26].
- Dynamic repulsion-based evolutionary algorithms: DREA [37].
- Fuzzy neighborhood-based DE with orientation: FONDE [46].

- Evolutionary multiobjective optimization-based multimodal optimization: EMO-MMO [47].
- Niching technique integrated with CMA-ES [48]: N-CMA-ES.
- Niching integrated with JADE [24]: N-JADE.
- Niching integrated with coyote algorithm [49]: N-COA.

The parameter settings are given in Table 4. All experiments are run 30 times for fair comparison. In addition, the advantage of CMA-ES, JADE and COA are to find the global optimal. In order to improve the performance of solving NESs, we combine niche techniques with these algorithms.

Table 4. Parameter settings for different methods.

Method	Parameter settings
CASDE	$NP = 100, F = 0.5, CR = 0.9, C = \{5, 6, 7, 8, 9, 10\}$
DDE/R	$NP = 100, F = 0.5, CR = 0.9, t = 20, \ell = 20$
MONES	$NP = 100, H_m = NP$
A-WeB	$NP = 100, H_m = NP$
RADE	$NP = 100, H_m = 200$
DREA	$NP = 10, u_{CR} = 0.5, u_F = 0.5, c = 0.1$
MODFA	$NP = 100, \alpha = 0.23, \beta_0 = 1, \delta = 0.98, \gamma = 1$
FONDE	$NP = 100, F = 0.5, CR = 0.9, m = 11$
Self-CCDE	$NP = 100, CR_m = 0.5$
Self-CSDE	$NP = 100, CR_m = 0.5$
EMO-MMO	$NP = 100, \eta = 0.1, m = 20, \phi = 0$
ANDE	$NP = 100, F = 0.9, CR = 0.1$
CMA-ES	$\mu = 5, \lambda = 10$
JADE	$u_{CR} = 0.5, u_F = 0.5, c = 0.1$
COA	$N_p = 20, N_c = 5$

Tables A2 and A3 in 8 show the detailed results of RR and SR. Additionally, the statistical results obtained by the Friedman test and the Wilcoxon test give in Tables 5 and 6, respectively. It is obvious that CASDE obtains the highest average values in both RR (**0.9951**) and SR (**0.9556**). Moreover, compared with other methods, it also achieves the highest ranking in Table 5. Meanwhile, from Table 6 obtained by the Wilcoxon test, CASDE significantly exceeds other methods in terms of RR and SR except DDE/R and FONDE since all $p-$values are less than 0.05.

Moreover, the results[†] obtained by Nemenyi test are listed in Table 7. We can observe the results of the best classified algorithms in the competition, like CASDE, we see that there are significant differences with algorithms like A-WeB, RADE, EMO-MMO,N-CMA-ES, N-JADE and N-COA but this difference is not significant for DDE/R, RADE, DREA, FONDE, and ANDE. More specific, CASDE has better performance than DDE/R, RADE, DREA, FONDE, and ANDE according to the results achieved by the Friedman test and the Wilcoxon test.

[†]In order to save space, we will only list the comparison between CASDE and other methods

Table 5. Average rankings of different algorithms obtained by the Friedman test for both *RR* and *SR*.

Algorithm	Ranking (*RR*)	Ranking (*SR*)
CASDE	**4.1500**	**4.1167**
DDE/R	4.4667	4.6500
A-WeB	8.2333	8.1333
RADE	6.1500	6.3500
DREA	6.1167	6.5833
FONDE	4.6333	4.5833
EMO-MMO	9.9167	10.6167
N-CMA-ES	11.1333	10.6167
N-JADE	9.8667	9.7667
N-COA	9.8833	10.1833

Table 6. Results obtained by the Wilcoxon test for algorithm CASDE in terms of *RR* and *SR* compared with state-of-the-art algorithms.

VS	*RR*			*SR*		
	R^+	R^-	*p*-value	R^+	R^-	*p*-value
DDE/R	275.5	189.5	*3.46E-01*	242.0	193.0	*3.85E-01*
A-WeB	426.0	39.0	**6.30E-05**	411.0	54.0	**1.04E-04**
RADE	375.0	90.0	**3.19E-03**	377.0	88.0	**2.78E-03**
DREA	332.5	132.5	**3.87E-02**	345.0	120.0	**1.42E-02**
FONDE	321.0	144.0	*5.42E-02*	319.0	149.0	*5.92E-02*
EMO-MMO	436.5	28.5	**5.01E-06**	447.0	18.0	**1.02E-06**
N-CMA-ES	439.5	25.5	**7.01E-06**	439.5	25.5	**0.00E+00**
N-JADE	437.5	27.5	**1.90E-05**	437.5	27.5	**0.00E+00**
N-COA	437.5	27.5	**2.3E-04**	437.5	27.5	**0.00E+00**

Table 7. Nemenyi test for comparison between different algorithms in terms of *RR*.

	CASDE	DDE/R	A-WeB	RADE	DREA	FONDE	EMO-MMO	N-CMA-ES	N-JADE	N-COA
CASDE		7.65E-01	**2.67E-04**	**4.28E-02**	7.67E-02	5.91E-01	**0.00E+00**	**0.00E+00**	**3.00E-06**	**0.00E+00**
DDE/R	7.65E-01		**8.16E-04**	8.41E-02	1.42E-01	8.14E-01	**1.00E-06**	**0.00E+00**	**1.20E-06**	**2.01E-06**
A-WeB	**2.67E-04**	**8.16E-04**		1.05E-01	6.06E-02	**1.86E-03**	1.35E-01	**2.51E-02**	3.06E-01	1.65E-01
RADE	**4.28E-02**	8.41E-02	1.05E-01		7.98E-01	1.35E-01	**1.85E-03**	**1.14E-04**	**8.20E-03**	**2.64E-03**
DREA	7.67E-02	1.42E-01	6.06E-02	7.98E-01		2.16E-01	**7.56E-04**	**3.90E-05**	**3.73E-03**	**1.10E-03**
FONDE	5.91E-01	8.14E-01	1.86E-03	1.35E-01	2.16E-01		**4.00E-06**	**0.00E+00**	**3.50E-05**	**7.01E-06**
EMO-MMO	**0.00E+00**	**1.00E-06**	1.35E-01	**1.85E-03**	**7.56E-04**	**4.00E-06**		4.55E-01	6.31E-01	9.15E-01
N-CMA-ES	**0.00E+00**	**0.00E+00**	**2.51E-02**	**1.14E-04**	**3.90E-05**	**0.00E+00**	4.55E-01		2.24E-01	3.9E-01
N-JADE	**3.00E-06**	**1.20E-06**	3.06E-01	**8.20E-03**	**3.73E-03**	**3.50E-05**	6.31E-01	2.24E-01		7.17E-01
N-COA	**0.00E+00**	**2.01E-06**	1.65E-01	**2.64E-03**	**1.10E-03**	**7.01E-06**	9.15E-01	3.9E-01	7.17E-01	

6. Discussion

As shown in Section 5.2, the superior performance of CASDE has been evaluated. In this section, the influence of fixed clustering size, the impact of different parameter settings, and different mutation operators are studied.

6.1. Influence of fixed clustering size

An important aspect concerning CASDE is the clustering size. Therefore, this section mainly discusses the impact of different clustering sizes on CASDE.

To make it simple, we respectively set the fixed sizes as 5, 10, 20, which is for the precise division. They are employed to replace the dynamic clustering size previously used in CASDE. Therefore, three CASDE variants, i.e., CASDE-5, CASDE-10 and CASDE-20, are developed[‡].

The detailed experiment results are shown in Table A4 for both RR and SR in the supplementary file. From Table A4, compared with CASDE-5, CASDE-10, and CASDE-20, CASDE also obtains the best average RR and SR values. Furthermore, the case where the fixed clustering size is equal to 20 is worth discussing. This large clustering size leads to a small number of species. Therefore, it is not conductive to solving the NESs contained many roots. From Table A4 , it can be seen that CASDE-20 successfully solve 16 out of 30 NESs over 30 independent runs. One characteristic of these NESs is that they contain few roots. In contrast, if the NESs has many roots, such as F03, F04, F13, F16, F17 , F23, and F24, CASDE-20 is difficult in locating all the roots in a single run.

The statical results from the Friedman test and Wilcoxon test are shown in Tables 8 and 9. We can see clearly that CASDE achieves the best ranking value from Table 8. Meanwhile, from Table 9, CASDE significantly outperforms CASDE-20 at $\alpha = 0.05$ for RR and SR.

Based on the above analysis, comparison with the fixed clustering size, the dynamic clustering size is able to improve the performance of algorithm. Besides, dynamic clustering sizes alleviates the trivial task to give the proper clustering size for different NESs problems.

Table 8. Average ranking of CASDE, CASDE-5, CASDE-10, and CASDEE-20 obtained by the Friedman test for both RR and SR criteria.

Algorithm	Ranking (RR)	Ranking (SR)
CASDE	**2.1500**	**2.1500**
CASDE-5	2.4000	2.3833
CASDE-10	2.3300	2.3500
CASDE-20	3.1167	3.1167

[‡]The only difference between CASDE and CASDE-5 (CASDE-10, CASDE-20) is the clustering size, all other settings remain the same.

Table 9. Results obtained by the wilcoxon test for CASDE in terms of RR and SR compared with CASDE-5, CASDE-10, and CASDE-20.

VS	RR			SR		
	R^+	R^-	p-value	R^+	R^-	p-value
CASDE-5	265.5	199.5	*4.91E-01*	260.5	204.5	*5.57E-01*
CASDE-10	285.0	150.0	*1.41E-01*	285.0	150.0	*1.41E-01*
CASDE-20	397.0	68.0	**6.89E-04**	397.0	68.0	**4.95E-04**

Table 10. Average rankings of casde and different parameter setting obtained by the friedman test for both RR and SR criteria.

Algorithm	Ranking (RR)	Ranking (SR)
CASDE	**4.3333**	**4.2667**
F=0.1,CR=0.1	7.3000	6.4333
F=0.1,CR=0.5	4.8833	4.8833
F=0.1,CR=0.9	4.6833	4.6167
F=0.5,CR=0.1	6.4167	6.2833
F=0.5,CR=0.5	4.6500	4.8333
F=0.5,CR=0.9	4.3500	4.6000
F=0.9,CR=0.1	7.8500	7.6330
F=0.9,CR=0.5	5.4167	5.4000
F=0.9,CR=0.9	5.1167	5.2833

6.2. Study on different parameter settings

To improve the search ability and avoid the trivial tasks of parameter setting, niche adaptive parameter setting is used in CASDE. In this subsection, we verify the effect of the static parameter settings (F and CR) on the performance of CASDE. Therefore, the adaptive parameter setting is removed from the CASDE, and nine diverse sets of parameters are used in CASDE, i.e., ($F = 0.1, CR = 0.1$), ($F = 0.1, CR = 0.5$), ($F = 0.1, CR = 0.9$), ($F = 0.5, CR = 0.1$), ($F = 0.5, CR = 0.5$), ($F = 0.5, CR = 0.9$), ($F = 0.9, CR = 0.1$), ($F = 0.9, CR = 0.5$), and ($F = 0.9, CR = 0.9$). For fair comparison, all parameters are consistent with CASDE.

The detailed results based on RR and SR values are respectively shown in Tables A5 and A6 in the supplementary material. Additionally, the average ranking obtained by the Friedman test are reported in Table 10 and the results derived from the Wilcoxon test are shown in Tabel 11.

From Tables A5 and A6, we can see that ($F = 0.1, CR = 0.9$) get the higher average RR and SR values over 30 independent runs. Besides, different parameters can influence the performance of RCDE for solving NESs. For example, three sets of parameters, such as ($F = 0.1, CR = 0.1$), ($F = 0.5, CR = 0.1$), and ($F = 0.9, CR = 0.1$) obtain poor results due to a small CR value. Therefore, we can conclude that the parameter settings has significant impact on the CASDE. Its optimal value is difficult to give in advance and problem-dependent.

From Table 10, CASDE get the highest ranking for *RR* and *SR* via the Friedman test. Moreover, Table 11 shows that CASDE is better than the static parameter settings, because it can obtain higher R^+ than R^- values in all cases. In general, the proposed niche adaptive parameter setting is effective. More importantly, it can avoid setting the proper values of F and CR for different NESs.

Table 11. Results obtained by the willcoxon test for algorithm casde in terms of *RR* and *SR* compared with casde-5, casde-10, and casde-20.

VS	*RR*			*SR*		
	R^+	R^-	p-value	R^+	R^-	p-value
F=0.1,CR=0.1	396.0	39.0	**1.04E-04**	413.0	52.0	**1.13E-04**
F=0.1,CR=0.5	261.5	173.5	*3.35E-01*	282.5	152.5	*1.55E-01*
F=0.1,CR=0.9	260.0	175.0	*3.52E-01*	288.0	177.0	*2.49E-01*
F=0.5,CR=0.1	367.0	68.0	**1.14E-03**	367.0	68.0	**7.49E-04**
F=0.5,CR=0.5	260.0	175.0	*3.52E-01*	288.0	177.0	*2.49E-01*
F=0.5,CR=0.9	233.0	202.0	*7.29E-01*	263.0	202.0	*5.23E-01*
F=0.9,CR=0.1	407.5	27.5	**3.01E-05**	407.5	27.5	**1.13E-05**
F=0.9,CR=0.5	311.5	153.5	*1.01E-01*	317.0	148.0	*7.80E-02*
F=0.9,CR=0.9	283.0	152.0	*1.53E-01*	317.0	148.0	*7.80E-02*

6.3. On different mutation operators

In CASDE, ”DE/rand/1” and ”DE/best/1” as shown in Eqs (2.3) and (2.4) are selected randomly during the search process. The hybrid mutation strategy can contribute to the balance of exploration and exploitation. In this subsection, we replace the hybrid mutation strategy with single mutation operator to evaluate the effectiveness of CASDE . To this end, we focus on two CASDE variants: 1) CASDE-1, i.e., CASDE with ”DE/rand/1”; 2) CASDE-2, i.e., CASDE with ”DE/best/1”. The parameter settings are remained the same for two CASDE variants.

The detailed experiment results are shown in Table A7 in the supplement file. Compared with CASDE-1 and CASDE-2, CASDE obtains the highest average *RR* and *SR* values. In addition, several interesting phenomena appear in the experiment results. For F04, F12, and F23, CASDE obtains better *RR* and *SR* values than CASDE-1; for F13, CASDE-1 gets the best result; whereas CASDE-2 shows the best performance for F16. Therefore, different test instances require reasonable use of mutation operators to improve the effectiveness of algorithm. Hence, this hybrid strategy is used to remedy the drawback to some extent.

The statistical results derived from Friedman and Wilcoxon test are reported in Tables 12 and 13. We can observe that CASDE achieves the best ranking from Table 12. Meanwhile, from Table 13, CASDE is better than CASDE-1 and CASDE-2. settings, because it can obtain higher R^+ than R^- values in all cases.

Table 12. Average rankings of casde, casde-1, and casde-2 obtained by the friedman test for both RR and SR criteria.

Algorithm	Ranking (RR)	Ranking (SR)
CASDE	**1.9167**	**1.916**
CASDE-1	2.1333	2.1333
CASDE-2	1.9500	1.9500

Table 13. Results obtained by the willcoxon test for algorithm casde in terms of RR and SR compared with casde-1, and casde-2.

VS	RR			SR		
	R^+	R^-	p-value	R^+	R^-	p-value
CASDE-1	286.0	179.0	≥ 0.2	287.0	178.0	≥ 0.2
CASDE-2	232.5	202.5	≥ 0.2	232.5	202.5	≥ 0.2

7. Solving real-world NES problems

In the previous experimental results, the superiority of CASDE is verified. In this section, the effectiveness of CASDE on real-world problems is studied. Two cases of motor system [50] are selected to measure the performance of CASDE.

7.1. Test cases

1) Case-1. This case mainly focuses on the solution of internal variables in steady-state operation of synchronous generator with magnetic saturation. For example, saturation characteristics of a synchronous generator, $s = 0.047\ 6E_q^{12}$, synchronous reactance $x_d = x_q = 2.264$, reactance of armature reaction $x_{ad} = x_{aq} = 2.104$, field winding self-inductance $x_F = 2.209$, field winding resistance $r_F = 0.0008$, stator resistance $r = 0.02$, steady state operation active power $P = 0.8$, whereas reactive power $Q = 0.4$, terminal voltage $U = 1.0\angle 0$. It needs to calculate the internal variables of the unit.

Firstly, the following equation can be derived from the relationship between the internal variables during the steady-state operation of the synchronous generator and the saturation factor of the magnetic circuit:

$$\begin{cases} \operatorname{tg}\delta = \frac{P\cdot x_q - Q\cdot r}{U+P\cdot r+Q\cdot x_q} \\ U_q = U\cos\delta \\ i_d = I\sin(\delta+\varphi) \\ i_q = I\cos(\delta+\varphi) \\ i_f = \frac{U_q+i_q\cdot r+i_d\cdot x_d}{x_{ad}} \\ I\sum = \sqrt{(if-i_d)^2+i_q^2} \\ s = 0.047\ 6E_q^{12} \\ x_{ad(sa)}(1+s) = x_{ad(un)} \\ x_{ad(sa)}I\sum = E_q \end{cases} \tag{7.1}$$

where δ is power angle; φ denotes power factor angle; E_q is the air gap voltage; $x_{ad(sa)}$ and $x_{ad(un)}$ respectively illustrate saturated direct axis armature reactive reactance and Unsaturated direct axis armature reactive reactance; $I\sum$ is resultant current, and s is saturation factor.

Let: $x_1 = \mathrm{tg}\frac{\delta}{2}$, $x_2 = I\sum$, $x_3 = x_{ad(sa)}$, $x_4 = i_d$, $x_5 = U_q$, $x_6 = i_q$, $x_7 = i_f$, $x_8 = E_q$. Thus, the motor problem can be transformed into a NES:

$$\begin{cases} e_1(\mathbf{x}) = 0.8(x_1^2 + x_1 - 1)x_3 + 0.12x_1^2 + 2.16x_1 - 0.12 = 0 \\ e_2(\mathbf{x}) = (1 + x_1^2)x_4 + 0.4x_1^2 - 1.6x_1 - 0.4 = 0 \\ e_3(\mathbf{x}) = (1 + x_1^2)x_5 + x_1^2 - 1 = 0 \\ e_4\mathbf{x}) = (1 + x_1^2)x_6 + 0.8(x_1^2 + x_1 - 1) = 0 \\ e_5(\mathbf{x}) = x_3x_7 - 0.02x_6 - x_5 - x_3x_4 - 0.16x_4 = 0 \\ e_6(\mathbf{x}) = x_7^2 - 2x_4x_7 + x_6^2 + x_4^2 - x_2^2 = 0 \\ e_7(\mathbf{x}) = x_8 - x_2x_3 = 0 \\ e_8(\mathbf{x}) = 0.0476x_3x_8^{12} + x_3 - 2.104 = 0 \end{cases}$$

where $x_{i\_min} = [-3, -1, -2, -1, -1, -0.5, -1.5, -1.5]$; $x_{i\_max} = [1, 1, 2, 1, 1, 0.5, 1.5, 1.5]$; i denotes i-th dimension; x_{i_min} and x_{i_max} are the lower and upper bounds of x_i. It has four roots as shown in Table 14.

Table 14. Roots of Case 1.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
1	0.378136	0.583102	1.863559	0.829251	0.749801	0.335176	1.306394	1.086646
2	0.378136	-0.583102	1.863559	0.829251	0.749801	0.335176	1.306394	-1.086646
3	-2.644550	0.583102	1.863559	-0.829251	-0.749801	-0.335176	-1.306394	1.086646
4	-2.644550	-0.583102	1.863559	-0.829251	-0.749801	-0.335176	-1.306394	-1.086646

2) Case-2. It is used to solve the circuit model parameters of synchronous generator d-axis equivalent circuit. Several known parameters of a synchronous generator are described as follows: $x_d = 1.803$, $x_d' = 0.442$, $x_d^* = 0.328$, $T_d' = 1.497s$, $T_d^* = 0.035s$, $r_F = 0.000856$, $x_e = 0.232$. The work is to calculate the parameters of D - axis equivalent circuit model (Canay model). Follow this, the equations can be obtained according to equivalent circuit model:

$$\begin{cases} x_{de} = x_d - x_e \\ x_{de}^* = x_d^* - x_e \\ T_{de}'T_{de}^*x_{de} = x_{de}^*\left(T_{do}'T_{do}^*\right) \\ x_{de}'\left[x_{de}^*\left(T_{do}' + T_{do}^*\right) - T_{de}^*\left(x_{de} + x_{de}^*\right)\right] = \\ x_{de}x_{de}^*\left(T_{de}' - T_{de}^*\right) \\ bx_{ad} = x_d - x_e \\ r_Fb^2 = r_{Fce}' \\ x_{Foe}'\left(x_{de} - x_{de}'\right) = x_{de}x_{de}' \\ r_{Foe}'\left(T_{de}'\omega_0\right) = x_{Foe}' \\ \left(T_{de}' + T_{de}'\right)x_{de} = x_d\left(T_d' + T_d^*\right) - x_e\left(T_{do}' + T_{do}^*\right) \end{cases} \tag{7.2}$$

Let: $x_1 = x'_{de}$, $x_2 = T'_{de}$, $x_3 = T^*_{de}$, $x_4 = x_e$, $x_5 = x_{de}$, $x_6 = x^*de$, $x_7 = b$, $x_8 = r'_{Fce}$, $x_9 = x'_{Fce}$. Application Case-2 can be transformed into a NES as follow:

$$\begin{cases} e_1(\mathbf{x}) = x_5 + x_4 - 1.803 = 0 \\ e_2(\mathbf{x}) = (x_2 + x_3)x_5 + 6.19116x_4 - 1.803(1.497 + 0.035) = 0 \\ e_3(\mathbf{x}) = x_6 + x_4 - 0.328 = 0 \\ e_4(\mathbf{x}) = 0.28801x_6 - x_2x_3x_5 = 0 \\ e_5(\mathbf{x}) = (-6.19116x_1 + x_1x_3 + x_2x_5 - x_3x_5)x_6 + x_1x_3x_5 = 0 \\ e_6(\mathbf{x}) = 1.571x_7 + x_4 - 1.803 = 0 \\ e_7(\mathbf{x}) = x_8 - 0.000856x_7^2 = 0 \\ e_8(\mathbf{x}) = (x_5 - x_1)x_9 - x_1x_5 = 0 \\ e_9(\mathbf{x}) = x_9 - 377x_2x_8 = 0 \end{cases}$$

where $x_{i\_min} = [-0.5, -1, -1, -1, 1, -1, -1, 0, -1], x_{i\_max} = [0.5, 1, 1, 1, 2, 1, 1, 1, 1]$; i denotes i-th dimension; x_{i_min} and x_{i_max} are the lower and upper bounds of x_i. It has two roots as shown in Table 15.

Table 15. Roots of Case-2.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
1	0.128764	0.495929	0.000000	0.328000	1.475000	0.000000	0.938892	0.000755	0.141080
2	-0.100030	-0.427811	0.097650	0.514846	1.288154	-0.186846	0.819958	0.000576	-0.092822
3	0.138229	0.528951	0.003490	0.318482	1.484517	0.009517	0.944950	0.000764	0.152422
4	0.000000	0.000000	0.495929	0.328000	1.475000	0.000000	0.938892	0.000754	0.000000

7.2. Results

In this subsection, CASDE is compared with the above ten methods, the parameter settings shown in Table 4. To make a fair comparison, each method is executed over 30 runs. For the two problems, $NFEs_{max} = 200,000$.

The detailed results are reported in Tables 16 and 17, where the best results are listed in bold. It can be observed that CASDE obtains the highest average *RR* values, i.e., 0.9750 and the highest average SR values, i.e., 0.9000. Thus, comparison with other ten methods, our proposed approach, CASDE, is effective and efficient for solving the two motor systems problems.

Table 16. Comparison between DDE/R and other state-of-the-art methods with respect to root ratio.

Prob.	CASDE	DDE/R	A-WeB	RADE	DREA	MODFA	FONDE	Self-CCDE	Self-CSDE	EMO-MMO	ANDE
F01	0.9500	0.5333	0.5250	0.5333	0.5000	0.0000	0.5000	0.5625	0.5250	0.3000	0.5333
F02	1.0000	0.6667	0.3667	0.9583	0.3750	0.0000	0.9250	0.9125	0.7125	0.0750	0.2667
Avg.	**0.9750**	0.6000	0.4459	0.7458	0.4375	0.0000	0.7125	0.7375	0.6188	0.1875	0.4000

Table 17. Comparison between DDE/R and other state-of-the-art methods with respect to success rate.

Prob.	CASDE	DDE/R	A-WeB	RADE	DREA	MODFA	FONDE	Self-CCDE	Self-CSDE	EMO-MMO	ANDE
F01	0.8000	0.0000	0.0333	0.1000	0.0000	0.0000	0.0000	0.1000	0.0500	0.0667	0.0667
F02	1.0000	0.2667	0.0667	0.8667	0.0000	0.0000	0.7667	0.8500	0.3000	0.0000	0.0000
Avg.	**0.9000**	0.1333	0.0500	0.4833	0.0000	0.0000	0.3833	0.4750	0.1750	0.0333	0.0000

8. Conclusions and future work

Solving NESs is a very challenging task due to the fact that it needs to locate multiple roots of NESs in a single run. To address this issue, we propose a re-initialization clustering-based adaptive differential evolution, named CASDE, in which the dynamic clustering sizes, niche adaptive parameter control, and re-initialization mechanism were combined together to solve NESs effectively. The performance of CASDE was verified by 30 NESs selected from the literature. Experimental results demonstrated that CASDE is able to locate multiple roots in a single run. In addition, comparison with other state-of-the-art methods, our approach also obtains significant performance. Moreover, we execute extensive experiments to analyze the performance of our approach, as well as the effectiveness of different parts of CASDEand the influence of different parameter settings. From the experiments, we can testify that CASDE can be considered as an effective alternative to solve NESs.

In the near future, we plan to use the fusion of reinforcement learning and evolutionary algorithm to solve NESs. By evaluating the actions using different strategies and parameters in a certain environment, the algorithm will have the characteristic of autonomous decision. Moreover, we will employ CASDE to solve other complex re-world NESs.

Acknowledgments

This work was partly supported by the Natural Science Foundation of GuangXi Province under Grant No. 2020GXNSFAA297174 and the High-level Talents Research Project of Beibu Gulf under Grant No. 2020KYQD06.

Conflict of interest

The authors declare no conflict of interest.

References

1. M. Kastner, Phase transitions and configuration space topology, *Rev. Mod. Phys.*, **80** (2008), 167–187.
2. D. Guo, Z. Nie, L. Yan, The application of noise-tolerant ZD design formula to robots' kinematic control via time-varying nonlinear equations solving, *IEEE Trans. Syst., Man, Cybern.: Syst.*, **48** (2017), 2188–2197.
3. H. D. Chiang, T. Wang, Novel homotopy theory for nonlinear networks and systems and its applications to electrical grids, *IEEE Trans. Control Network Syst.*, **5** (2017), 1051–1060.

4. F. Facchinei, C. Kanzow, Generalized nash equilibrium problems, *Ann. Oper. Res.*, **175** (2010), 177–211.
5. Z. Sun, J. Wu, J. Pei, Z. Li, Y. Huang, J. Yang, Inclined geosynchronous spaceborne cairborne bistatic sar: Performance analysis and mission design, *IEEE Trans. Geosci. Remote Sens.*, **54** (2015), 343–357.
6. Y. Song, L. Xing, M. Wang, Y. Yi, W. Xiang, Z. Zhang, A knowledge-based evolutionary algorithm for relay satellite system mission scheduling problem, *Comput. Ind. Eng.*, **150** (2020), 106830.
7. R. Storn, K. Price, Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces, *J. Global Opt.*, **11** (1997), 341–359.
8. A. P. Piotrowski, J. J. Napiorkowski, Step-by-step improvement of jade and shade-based algorithms: Success or failure?, *Swarm Evol. Comput.*, **43** (2018), 88–108.
9. J. N. Bharothu, M. Sridhar, R. S. Rao, Modified adaptive differential evolution based optimal operation and security of ac-dc microgrid systems, *Int. J. Electr. Power Energy Syst.*, **103** (2018), 185–202.
10. S. Li, Q. Gu, W. Gong, B. Ning, An enhanced adaptive differential evolution algorithm for parameter extraction of photovoltaic models, *Energy Convers. Manage.*, **205** (2020), 112443.
11. A. W. Mohamed, A. A. Hadi, K. M. Jambi, Novel mutation strategy for enhancing shade and lshade algorithms for global numerical optimization,*Swarm Evol. Comput.*, **50** (2019), 100455.
12. J. Pierezan, R. Z. Freire, L. Weihmann, G. Reynoso-Meza, L. dos Santos Coelho, Static force capability optimization of humanoids robots based on modified self-adaptive differential evolution, *Comput. Oper. Res.*, **84** (2017), 205–215.
13. L. dos Santos Coelho, H. V. H. Ayala, V. C. Mariani, A self-adaptive chaotic differential evolution algorithm using gamma distribution for unconstrained global optimization, *Appl. Math. Comput.*, **234** (2014), 452–459.
14. G. Li, Q. Lin, L. Cui, Z. Du, Z. Liang, J. Chen, et al., A novel hybrid differential evolution algorithm with modified code and jade, *Appl. Soft Comput.*, **47** (2016), 577–599.
15. P. Civicioglu, E. Besdok, Bezier search differential evolution algorithm for numerical function optimization: A comparative study with crmlsp, mvo, wa, shade and lshade, *Expert Syst. Appl.*, **165** (2021), 113875.
16. F. Zhao, L. Zhao, L. Wang, H. Song, A collaborative lshade algorithm with comprehensive learning mechanism, *Appl. Soft Comput.*, **96** (2020), 106609.
17. W. Gong, Z. Liao, X. Mi, L. Wang, Y. Guo, Nonlinear equations solving with intelligent optimization algorithms: A survey, *Complex Syst. Model. Simul.*, **1** (2021), 15–32.
18. E. Pourjafari, H. Mojallali, Solving nonlinear equations systems with a new approach based on invasive weed optimization algorithm and clustering, *Swarm Evol. Comput.*, **4** (2012), 33–43.
19. G. C. Ramadas, E. M. Fernandes, A. A. Rocha, Multiple roots of systems of equations by repulsion merit functions, in *International Conference on Computational Science and Its Applications*, Springer, Cham, (2014), 126–139.
20. A. K. Jain, M. N. Murty, P. J. Flynn, Data clustering: A review, *ACM Comput. Surv.*, **31** (1999), 264–323.

21. I. Tsoulos, A. Stavrakoudis, On locating all roots of systems of nonlinear equations inside bounded domain using global optimization methods, *Nonlinear Anal.: Real World Appl.*, **11** (2010), 2465–2471.

22. G. Karafotias, M. Hoogendoorn, A. E. Eiben, Parameter control in evolutionary algorithms: Trends and challenges, *IEEE Trans. Evol. Comput.*, **19** (2015), 167–187.

23. A. K. Qin, V. L. Huang, P. N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, *IEEE Trans. Evol. Comput.*, **13** (2009), 398–417.

24. J. Zhang, A. C. Sanderson, JADE: Adaptive differential evolution with optional external archive, *IEEE Trans. Evol. Comput.*, **13** (2009), 945–958.

25. S. Yang, J. Wang, Y. Ma, Y. Tu, Multi-response online parameter design based on bayesian vector autoregression model, *Comput. Ind. Eng.*, **149** (2020), 106775.

26. W. Gong, Y. Wang, Z. Cai, L. Wang, Finding multiple roots of nonlinear equation systems via a repulsion-based adaptive differential evolution, *IEEE Trans. Syst. Man Cybern. Syst.*, **50** (2018), 1499–1513.

27. Y. Guo, H. Yang, M. Chen, J. Cheng, D. Gong, Ensemble prediction-based dynamic robust multi-objective optimization methods, *Swarm Evol. Comput.*, **48** (2019), 156–171.

28. Y. N. Guo, X. Zhang, D. W. Gong, Z. Zhang, J. J. Yang, Novel interactive preference-based multi-objective evolutionary optimization for bolt supporting networks, *IEEE Trans. Evol. Comput.*, **24** (2019), 750-764.

29. C. Grosan, A. Abraham, A new approach for solving nonlinear equations systems, *IEEE Trans. Syst., Man Cybern., Part A: Syst. Humans*, **38** (2008), 698–714.

30. W. Song, Y. Wang, H. X. Li, Z. Cai, Locating multiple optimal solutions of nonlinear equation systems based on multiobjective optimization, *IEEE Trans. Evol. Comput.*, **19** (2015), 414–431.

31. W. Gong, Y. Wang, Z. Cai, S. Yang, A weighted biobjective transformation technique for locating multiple optimal solutions of nonlinear equation systems, *IEEE Trans. Evol. Comput.*, **21** (2017), 697–713.

32. Y. R. Naidu, A. K. Ojha, Solving multiobjective optimization problems using hybrid cooperative invasive weed optimization with multiple populations, *IEEE Trans. Syst., Man, Cybern.: Syst.*, **48** (2016), 821–832.

33. W. Sacco, N. Henderson, Finding all solutions of nonlinear systems using a hybrid metaheuristic with fuzzy clustering means, *Appl. Soft Comput.* **11** (2011), 5424–5432.

34. L. Freitas, G. Platt, N. Henderson, Novel approach for the calculation of critical points in binary mixtures using global optimization, *Fluid Phase Equilib.*, **225** (2004), 29–37.

35. N. Henderson, W. F. Sacco, G. M. Platt, Finding more than one root of nonlinear equations via a polarization technique: An application to double retrograde vaporization, *Chem. Eng. Res. Des.*, **88** (2010), 551–561.

36. R. M. A. Silva, M. G. C. Resende, P. M. Pardalos, Finding multiple roots of a box-constrained system of nonlinear equations with a biased random-key genetic algorithm, *J. Global Opt.*, **60** (2014), 289–306.

37. Z. Liao, W. Gong, X. Yan, L. Wang, C. Hu, Solving nonlinear equations system with dynamic repulsion-based evolutionary algorithms, *IEEE Trans. Syst., Man, Cybern.: Syst.*, **50** (2018), 1590–1601.

38. A. F. Kuri-Morales, R. H. No, D. México, Solution of simultaneous non-linear equations using genetic algorithms, *WSEAS Trans. Syst.*, **2** (2003), 44–51.

39. A. Pourrajabian, R. Ebrahimi, M. Mirzaei, M. Shams, Applying genetic algorithms for solving nonlinear algebraic equations, *Appl. Math. Comput.*, **219** (2013), 11483–11494.

40. W. Gao, G. G. Yen, S. Liu, A cluster-based differential evolution with self-adaptive strategy for multimodal optimization, *IEEE Trans. Cybern.*, **44** (2014), 1314–1327.

41. Q. Yang, W. N. Chen, Y. Li, C. L. Chen, X. M. Xu, J. Zhang, Multimodal estimation of distribution algorithms, *IEEE Trans. Cybern.*, **47** (2016), 636–650.

42. C. A. Floudas, Recent advances in global optimization for process synthesis, design and control: Enclosure of all solutions, *Comput. Chem. Eng.*, **23** (1999), S963–S973.

43. C. Wang, R. Luo, K. Wu, B. Han, A new filled function method for an unconstrained nonlinear equation, *J. Comput. Appl. Math.*, **235** (2011), 1689–1699.

44. I. Z. Emiris, B. Mourrain, Computer algebra methods for studying and computing molecular conformations, *Algorithmica*, **25** (1999), 372–402.

45. W. Gong, Y. Wang, Z. Cai, S. Yang, A weighted biobjective transformation technique for locating multiple optimal solutions of nonlinear equation systems, *IEEE Trans. Evol. Comput.*, **21** (2017), 697–713.

46. W. He, W. Gong, L. Wang, X. Yan, C. Hu, Fuzzy neighborhood-based differential evolution with orientation for nonlinear equation systems, *Knowl.-Based Syst.*, **182** (2019), 104796.

47. R. Cheng, M. Li, K. Li, X. Yao, Evolutionary multiobjective optimization-based multimodal optimization: Fitness landscape approximation and peak detection, *IEEE Trans. Evol. Comput.*, **22** (2018), 692–706.

48. N. Hansen, S. D. Müller, P. Koumoutsakos, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES), *Evol. Comput.*, **11** (2003), 1–18.

49. J. Pierezan, L. Coelho, Coyote optimization algorithm: A new metaheuristic for global optimization problems, in *2018 IEEE Congress on Evolutionary Computation (CEC)*, 2018.

50. C. W. LUO Y, Newton chaos iterative method and its application in electric machine, in *Proceedings of the CSU-EPSA*, **1** (2006).

Appendix: Detailed results

Table A1. Influence of different parts in CASDE with respect to the peak ratio and success rate.

Prob.	RR					SR				
	CASDE	CASDE/AR	CASDE/DR	CASDE/DA	DCS-DE	CASDE	CASDE/AR	CASDE/DR	CASDE/DA	DCS-DE
F01	**1.0000**	0.8667	**1.0000**	0.9667	**1.0000**	**1.0000**	0.7333	**1.0000**	0.9333	**1.0000**
F02	**1.0000**	**1.0000**	**1.0000**	0.9515	0.9273	**1.0000**	**1.0000**	**1.0000**	0.6667	0.4667
F03	**1.0000**	**1.0000**	**1.0000**	0.9911	0.9733	**1.0000**	**1.0000**	**1.0000**	0.8667	0.6000
F04	**1.0000**	**1.0000**	0.7538	0.8103	0.7949	**1.0000**	**1.0000**	0.0000	0.0000	0.0667
F05	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F06	**1.0000**	**1.0000**	0.9667	0.9667	0.8833	**1.0000**	**1.0000**	0.7333	0.7333	0.3333
F07	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F08	**1.0000**	**1.0000**	**1.0000**	0.9905	0.9619	**1.0000**	**1.0000**	1.0000	0.9333	0.7333
F09	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F10	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F11	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F12	0.9867	**0.9933**	0.9467	0.8667	0.8667	0.9333	**0.9667**	0.7333	0.2000	0.2000
F13	0.9167	**0.9778**	0.9333	0.9500	0.9556	0.4667	**0.7667**	0.4667	0.6667	0.6000
F14	**1.0000**	**1.0000**	0.9926	0.9481	0.9259	**1.0000**	**1.0000**	0.9333	0.5333	0.4000
F15	**1.0000**	**1.0000**	1.0000	0.0000	0.0000	**1.0000**	**1.0000**	**1.0000**	0.0000	0.0000
F16	0.9744	**1.0000**	0.7026	0.9128	0.9641	0.6667	**1.0000**	0.0000	0.0667	0.6000
F17	**1.0000**	0.6125	0.9542	0.8750	0.7417	**1.0000**	0.0000	0.4667	0.0667	0.0000
F18	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F19	**1.0000**	0.8833	**1.0000**	**1.0000**	0.4333	**1.0000**	0.7667	**1.0000**	**1.0000**	0.3333
F20	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F21	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F22	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9889	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9333
F23	0.9750	0.8979	0.2583	0.4000	0.3875	0.6000	0.0333	0.0000	0.0000	0.0000
F24	**1.0000**	**1.0000**	**1.0000**	0.8917	0.9167	**1.0000**	**1.0000**	**1.0000**	0.1333	0.3333
F25	**1.0000**	0.9667	0.6000	**1.0000**	0.9333	**1.0000**	0.9333	0.2000	1.0000	0.8667
F26	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F27	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F28	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F29	**1.0000**	**1.0000**	0.9867	0.9867	0.9867	**1.0000**	**1.0000**	0.9333	0.9333	0.9333
F30	1.0000	**1.0000**	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Avg	**0.9951**	0.9733	0.9365	0.9169	0.8880	**0.9556**	0.9067	0.8156	0.7244	0.6800

Table A2. Comparison between DDE/R and other state-of-the-art methods with respect to root ratio.

Prob	CASDE	DDE/R	A-WeB	RADE	DREA	FONDE	EMO-MMO	N-CMA-ES	N-JADE	N-ACO
F01	**1.0000**	0.9333	0.6200	**1.0000**	0.0000	**1.0000**	0.5000	0.2667	**1.0000**	**1.0000**
F02	**1.0000**	**1.0000**	**1.0000**	0.9900	0.9970	**1.0000**	0.8455	**1.0000**	0.8545	0.8909
F03	**1.0000**	**1.0000**	0.9573	0.9960	0.9578	**1.0000**	0.9267	0.9511	0.9156	0.7067
F04	**1.0000**	**1.0000**	**1.0000**	0.9015	**1.0000**	0.9600	0.7500	0.1333	0.2103	0.7795
F05	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F06	**1.0000**	**1.0000**	0.9400	0.9900	0.9583	0.9975	0.3625	0.0083	0.2250	0.9750
F07	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F08	**1.0000**	**1.0000**	0.8371	0.9971	**1.0000**	**1.0000**	0.8500	0.5333	0.6095	0.7905
F09	**1.0000**	0.9778	0.8933	0.9700	**1.0000**	**1.0000**	0.1167	0.8667	0.9778	0.4222
F10	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.3250	**1.0000**	**1.0000**	**1.0000**
F11	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9750	0.4167	**1.0000**	**1.0000**
F12	0.9867	**1.0000**	0.8880	0.6310	0.8767	0.8640	0.8000	0.2267	0.2800	0.6067
F13	0.9167	0.7278	0.0933	0.8908	0.9167	0.7383	0.5167	0.3056	0.2778	0.5944
F14	**1.0000**	**1.0000**	0.9733	0.9867	**1.0000**	**1.0000**	0.4333	0.0296	0.2296	0.9185
F15	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.5000	**1.0000**	**1.0000**	0.9333	0.5000	0.5000
F16	0.9744	**1.0000**	**1.0000**	0.9954	**1.0000**	**1.0000**	**1.0000**	0.5846	0.6205	0.3744
F17	**1.0000**	**1.0000**	0.6688	0.9444	**1.0000**	0.9850	0.5406	0.0833	0.7458	0.0000
F18	**1.0000**	**1.0000**	0.9433	**1.0000**	**1.0000**	**1.0000**	0.9917	0.9778	0.9778	**1.0000**
F19	**1.0000**	0.9000	0.6200	0.7950	0.0000	0.9300	0.3250	0.4333	**1.0000**	0.6667
F20	**1.0000**	**1.0000**	0.9514	**1.0000**	**1.0000**	**1.0000**	0.9571	0.9905	0.7714	0.5714
F21	**1.0000**	**1.0000**	0.9950	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
F22	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9833	0.9889	0.9778	**1.0000**
F23	**0.9750**	0.9417	0.1563	0.5619	0.7979	0.9100	0.5219	0.1250	0.1250	0.1250
F24	**1.0000**	**1.0000**	0.8550	0.9988	0.9125	**1.0000**	0.8063	**1.0000**	0.9917	0.7583
F25	**1.0000**	**1.0000**	0.2360	0.8350	**1.0000**	**1.0000**	0.3500	0.0000	0.2667	0.0000
F26	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F27	**1.0000**	**1.0000**	**1.0000**	0.9967	**1.0000**	**1.0000**	0.8667	0.6667	0.7556	0.6889
F28	**1.0000**	**1.0000**	0.9400	**1.0000**	**1.0000**	**1.0000**	0.9250	0.0333	**1.0000**	**1.0000**
F29	**1.0000**	**1.0000**	0.9320	0.9940	0.9533	0.9960	0.2000	0.9067	0.9733	0.6000
F30	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9833	**1.0000**	0.6333
Avg.	**0.9951**	0.9827	0.8500	0.9491	0.8957	0.9794	0.7290	0.6148	0.7429	0.7201

Table A3. Comparison between DDE/R and other state-of-the-art methods with respect to success rate.

Prob.	casde	DDE/R	A-WeB	RADE	DREA	FONDE	EMO-MMO	N-CMA-ES	N-LSHADE	N-ACO
F01	**1.0000**	0.8667	0.3600	**1.0000**	0.0000	**1.0000**	0.0000	0.0000	**1.0000**	**1.0000**
F02	**1.0000**	**1.0000**	**1.0000**	0.9000	0.9300	**1.0000**	0.2000	**1.0000**	0.2000	0.2000
F03	**1.0000**	**1.0000**	0.5800	0.9500	0.4600	**1.0000**	0.3000	0.4000	0.2667	0.0000
F04	**1.0000**	**1.0000**	**1.0000**	0.3100	**1.0000**	0.5200	0.3500	0.0000	0.0000	0.0000
F05	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F06	**1.0000**	**1.0000**	0.6000	0.9300	0.6600	0.9800	0.1000	0.0000	0.0000	0.8000
F07	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F08	**1.0000**	**1.0000**	0.1200	0.9800	**1.0000**	**1.0000**	0.2000	0.0000	0.0000	0.0000
F09	**1.0000**	0.9333	0.6800	0.9100	**1.0000**	**1.0000**	0.0000	0.6000	0.9333	0.0000
F10	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.0000	**1.0000**	**1.0000**	**1.0000**
F11	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9500	0.0667	**1.0000**	**1.0000**
F12	0.9333	**1.0000**	0.2800	0.0000	0.0300	0.2800	0.2000	0.0000	0.0000	0.0000
F13	0.4667	0.0000	0.0000	0.1900	0.2600	0.0200	0.0000	0.0000	0.0000	0.0000
F14	**1.0000**	**1.0000**	0.7600	0.8900	**1.0000**	**1.0000**	0.2000	0.0000	0.0000	0.4000
F15	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.0000	**1.0000**	**1.0000**	0.8667	0.0000	0.0000
F16	0.6667	**1.0000**	**1.0000**	0.9400	**1.0000**	**1.0000**	**1.0000**	0.0000	0.0000	0.0000
F17	1.0000	1.0000	0.0000	0.4300	0.0300	0.7600	0.0000	0.0000	0.0000	0.0000
F18	**1.0000**	**1.0000**	0.6600	**1.0000**	**1.0000**	**1.0000**	0.9500	0.8667	0.8667	**1.0000**
F19	**1.0000**	0.8000	0.2400	0.6900	0.0000	0.8600	0.1500	0.2667	**1.0000**	0.5333
F20	**1.0000**	**1.0000**	0.7000	**1.0000**	**1.0000**	**1.0000**	0.7500	0.9333	0.0667	0.0000
F21	**1.0000**	**1.0000**	0.9800	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
F22	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9500	0.9333	0.8667	**1.0000**
F23	**0.6000**	0.3333	0.0000	0.0000	0.0000	0.2800	0.0000	0.0000	0.0000	0.0000
F24	**1.0000**	**1.0000**	0.1400	0.9900	0.4300	**1.0000**	0.0000	1.0000	0.9333	0.0000
F25	**1.0000**	**1.0000**	0.0200	0.6700	**1.0000**	**1.0000**	0.0000	0.0000	0.0000	0.0000
F26	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F27	**1.0000**	**1.0000**	**1.0000**	0.9900	**1.0000**	**1.0000**	0.7000	0.2667	0.2667	0.0667
F28	**1.0000**	**1.0000**	0.8800	**1.0000**	**1.0000**	**1.0000**	0.9000	0.0000	**1.0000**	**1.0000**
F29	**1.0000**	**1.0000**	0.6600	0.9700	0.7600	0.9800	0.0000	0.5333	0.8667	0.0000
F30	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9333	**1.0000**	0.0667
AVG.	**0.9556**	0.9311	0.6553	0.8247	0.7187	0.8893	0.4633	0.4556	0.5089	0.4022

Table A4. Influence of different fixed clustering size inCASDE with respect to the peak ratio and success rate.

Prob.	*RR*				*SR*			
	CASDE	CASDE-5	CASDE-10	CASDE-20	CASDE	CASDE-5	CASDE-10	CASDE-20
F01	**1.0000**	0.8667	**1.0000**	**1.0000**	**1.0000**	0.7333	**1.0000**	**1.0000**
F02	**1.0000**	**1.0000**	**1.0000**	0.9879	**1.0000**	**1.0000**	**1.0000**	0.8667
F03	**1.0000**	**1.0000**	**1.0000**	0.7533	**1.0000**	**1.0000**	**1.0000**	0.0000
F04	**1.0000**	**1.0000**	0.9974	0.7103	**1.0000**	**1.0000**	0.9667	0.0000
F05	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
F06	**1.0000**	**1.0000**	**1.0000**	0.8417	**1.0000**	**1.0000**	**1.0000**	0.0667
F07	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
F08	**1.0000**	**1.0000**	**1.0000**	0.9952	**1.0000**	**1.0000**	**1.0000**	0.9667
F09	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
F10	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
F11	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
F12	0.9867	**0.9933**	0.9733	0.8567	0.9333	**0.9667**	0.8667	0.2000
F13	0.9167	**0.9778**	0.8667	0.5472	0.4667	**0.7667**	0.1333	0.0000
F14	**1.0000**	**1.0000**	**1.0000**	0.8185	**1.0000**	**1.0000**	**1.0000**	0.0333
F15	**1.0000**	**1.0000**	**1.0000**	0.9167	**1.0000**	**1.0000**	**1.0000**	0.8333
F16	0.9744	**1.0000**	0.7487	0.4513	0.6667	**1.0000**	0.0000	0.0000
F17	**1.0000**	0.6125	**1.0000**	0.7750	**1.0000**	0.0000	**1.0000**	0.0000
F18	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
F19	**1.0000**	0.8833	**1.0000**	**1.0000**	**1.0000**	0.7667	**1.0000**	**1.0000**
F20	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
F21	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
F22	**1.0000**	**1.0000**	**1.0000**	0.9667	**1.0000**	**1.0000**	**1.0000**	0.8000
F23	**0.9750**	0.8979	0.9417	0.7875	**0.6000**	0.0333	0.1667	0.0000
F24	**1.0000**	**1.0000**	**1.0000**	0.1875	**1.0000**	**1.0000**	**1.0000**	0.0000
F25	**1.0000**	0.9667	**1.0000**	**1.0000**	**1.0000**	0.9333	**1.0000**	**1.0000**
F26	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
F27	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
F28	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
F29	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
F30	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Avg.	**0.9951**	0.9733	0.9843	0.8865	**0.9556**	0.9067	0.9044	0.6589

Table A5. Influence of different parameter setting in CASDE with respect to the peak ratio.

	$F = 0.1$			$F = 0.5$			$F = 0.9$		
Prob.	$CR = 0.1$	$CR = 0.5$	$CR = 0.9$	$CR = 0.1$	$CR = 0.5$	$CR = 0.9$	$CR = 0.1$	$CR = 0.5$	$CR = 0.9$
F01	0.4667	0.9333	**1.0000**	1.0000	1.0000	1.0000	**1.0000**	**1.0000**	0.0000
F02	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F03	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F04	0.7282	1.0000	1.0000	0.5692	**1.0000**	**1.0000**	0.4872	0.9897	**1.0000**
F05	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F06	0.9833	**1.0000**	**1.0000**	1.0000	1.0000	1.0000	0.9333	0.9833	**0.9917**
F07	0.7000	**1.0000**	**1.0000**	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F08	1.0000	1.0000	1.0000	0.9810	**1.0000**	**1.0000**	0.9810	**1.0000**	**1.0000**
F09	0.0000	0.9667	**1.0000**	0.0000	**1.0000**	**1.0000**	0.0000	**1.0000**	**1.0000**
F10	0.9667	**1.0000**	**1.0000**	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F11	0.5833	**1.0000**	**1.0000**	0.9167	1.0000	1.0000	0.8000	**1.0000**	**1.0000**
F12	0.9533	0.9933	0.9867	0.8800	0.9667	**1.0000**	0.8667	**1.0000**	**1.0000**
F13	0.4722	0.5333	0.5778	**0.8056**	0.8500	0.8444	0.8778	0.9500	**0.9944**
F14	1.0000	1.0000	1.0000	0.9852	**1.0000**	**1.0000**	0.9111	0.9926	**1.0000**
F15	0.8333	**1.0000**	**1.0000**	1.0000	1.0000	1.0000	0.9333	**1.0000**	0.9667
F16	0.9744	**1.0000**	**1.0000**	0.9026	0.9897	**1.0000**	0.7641	0.7897	0.7692
F17	0.9917	**1.0000**	**1.0000**	0.9917	**1.0000**	**1.0000**	0.8875	**0.9542**	**0.9583**
F18	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F19	0.0000	0.7000	**0.9667**	0.0000	**0.4667**	**0.4667**	0.0000	0.0000	0.0333
F20	0.9905	**1.0000**	**1.0000**	1.0000	1.0000	1.0000	0.9714	**1.0000**	**1.0000**
F21	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F22	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9889	**1.0000**	**1.0000**
F23	0.4458	0.7417	**0.8917**	0.2792	0.7458	0.7458	0.2167	0.5292	0.8583
F24	1.0000	1.0000	0.9250	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F25	0.2333	**1.0000**	**1.0000**	0.2667	**0.9667**	**0.9667**	0.1333	0.9667	**1.0000**
F26	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F27	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F28	0.5000	**1.0000**	**1.0000**	0.5000	**1.0000**	**1.0000**	0.5000	**1.0000**	**1.0000**
F29	0.9600	**1.0000**	**1.0000**	1.0000	1.0000	1.0000	0.9333	**1.0000**	**1.0000**
F30	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9833	**1.0000**	**1.0000**
Avg.	0.7928	0.9623	**0.9783**	0.8359	0.9662	**0.9675**	0.8056	**0.9385**	0.9191

Table A6. Influence of different parameter setting in CASDE with respect to the success rate.

	F=0.1			F=0.5			F=0.9		
Prob.	CR = 0.1	CR = 0.5	CR = 0.9	CR = 0.1	CR = 0.5	CR = 0.9	CR = 0.1	CR = 0.5	CR = 0.9
F01	0.2000	0.9333	**1.0000**	1.0000	1.0000	1.0000	**1.0000**	**1.0000**	0.0000
F02	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F03	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F04	0.0000	1.0000	1.0000	0.0000	**1.0000**	**1.0000**	0.0000	0.8667	**1.0000**
F05	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F06	0.8667	**1.0000**	**1.0000**	1.0000	1.0000	1.0000	0.6000	0.8667	**0.9333**
F07	0.4000	**1.0000**	**1.0000**	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F08	1.0000	1.0000	1.0000	0.8667	**1.0000**	**1.0000**	0.8667	**1.0000**	**1.0000**
F09	0.0000	0.9333	**1.0000**	0.0000	**1.0000**	**1.0000**	0.0000	**1.0000**	**1.0000**
F10	0.9333	**1.0000**	**1.0000**	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F11	0.0667	**1.0000**	**1.0000**	0.6667	1.0000	1.0000	0.3333	**1.0000**	**1.0000**
F12	0.6667	0.9333	0.9333	0.4000	0.8000	**1.0000**	0.4000	**1.0000**	**1.0000**
F13	0.0000	0.0000	0.0000	**0.1333**	0.0667	0.0667	0.2000	0.5333	**0.9333**
F14	1.0000	1.0000	1.0000	0.8667	**1.0000**	**1.0000**	0.4000	0.9333	**1.0000**
F15	0.7333	**1.0000**	**1.0000**	1.0000	1.0000	1.0000	0.8667	**1.0000**	0.9333
F16	0.6667	**1.0000**	**1.0000**	0.1333	0.8667	**1.0000**	0.0000	0.0000	0.0000
F17	0.8667	**1.0000**	**1.0000**	0.8667	**1.0000**	**1.0000**	0.0667	**0.4000**	**0.4000**
F18	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F19	0.0000	0.6000	**0.9333**	0.0000	**0.2000**	**0.2000**	0.0000	0.0000	0.0000
F20	0.9333	**1.0000**	**1.0000**	1.0000	1.0000	1.0000	0.8000	**1.0000**	**1.0000**
F21	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F22	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9333	**1.0000**	**1.0000**
F23	0.0000	0.0000	**0.1333**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
F24	1.0000	1.0000	0.4000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F25	0.0667	**1.0000**	**1.0000**	0.0667	**0.9333**	**0.9333**	0.0000	0.9333	**1.0000**
F26	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F27	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F28	0.0000	**1.0000**	**1.0000**	0.0000	**1.0000**	**1.0000**	0.0000	**1.0000**	**1.0000**
F29	0.8000	1.0000	1.0000	1.0000	1.0000	1.0000	0.6667	**1.0000**	**1.0000**
F30	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9333	**1.0000**	**1.0000**
Avg.	0.6400	**0.9133**	**0.9133**	0.7000	0.8956	**0.9067**	0.6022	**0.8511**	0.8400

Table A7. Influence of different mutation operators on CASDE in terms of the peak ratio and success rate.

Prob.	*RR*			*SR*		
	CASDE	CASDE-1	CASDE-2	CASDE	CASDE-1	CASDE-2
F01	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F02	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F03	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F04	**1.0000**	0.9897	**1.0000**	**1.0000**	0.8667	**1.0000**
F05	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F06	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F07	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F08	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F09	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F10	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F11	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F12	**0.9867**	0.9667	**0.9867**	**0.9333**	0.8000	**0.9333**
F13	0.9167	**0.9833**	0.8778	0.4667	**0.8000**	0.1333
F14	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F15	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F16	0.9744	0.9538	**0.9897**	0.6667	0.4000	**0.8667**
F17	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F18	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F19	1.0000	0.6000	1.0000	1.0000	0.6000	1.0000
F20	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F21	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F22	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F23	**0.9750**	0.9125	0.9292	**0.6000**	0.0667	0.2000
F24	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F25	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F26	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F27	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F28	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F29	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
F30	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Avg.	**0.9951**	0.9802	0.9928	**0.9556**	0.9178	0.9378