COMPLEX SYSTEM MODELING AND SIMULATION
ISSN 2096-9929 05/06 pp 78−95
Volume 2, Number 1, March 2022
DOI: 10.23919/CSMS.2022.0003

Solving Nonlinear Equations Systems with an Enhanced Reinforcement Learning Based Differential Evolution

Zuowen Liao* and Shuijia Li

Abstract: Nonlinear equations systems (NESs) arise in a wide range of domains. Solving NESs requires the algorithm to locate multiple roots simultaneously. To deal with NESs efficiently, this study presents an enhanced reinforcement learning based differential evolution with the following major characteristics: (1) the design of state function uses the information on the fitness alternation action; (2) different neighborhood sizes and mutation strategies are combined as optional actions; and (3) the unbalanced assignment method is adopted to change the reward value to select the optimal actions. To evaluate the performance of our approach, 30 NESs test problems and 18 test instances with different features are selected as the test suite. The experimental results indicate that the proposed approach can improve the performance in solving NESs, and outperform several state-of-the-art methods.

Key words: nonlinear equations systems; reinforcement learning; differential evolution; multiple roots location

1 Introduction

Nonlinear equations systems (NESs) arise in a wide range of domains[1], including economics[2], power systems[3], and complex systems[4]. Generally, an NES can be expressed as follows:

$$F(X) = 0 \tag{1}$$

where $F(X) = (f_1(X), \ldots, f_m(X))^{\mathrm{T}}$. $f_1(X), \ldots, f_m(X)$ represen m equations, at least one of which is a nonlinear equation; $X = (x_1, x_2, \ldots, x_n)$ is an n-dimensional vector; and S is the search space, which can be expressed as follows:

$$S = [\underline{x}_i, \overline{x}_i]^n \tag{2}$$

where $\underline{x}_i$ and $\overline{x}_i$ are the lower and upper bounds of x_i, $i \in [1, 2, \ldots, n]$, respectively.

In real-world applications, an NES contains multiple roots, and the decision-maker can select diverse solutions in complex environments[4]. For example, the synthesis problem of the motion mechanism can be expressed as nonlinear equations. By obtaining multiple solutions to nonlinear equations, the motion characteristics and transmission principles of complex machines, which lay the foundation for the further selection of materials and the design of load-bearing capacity design[5], can be expressed concisely.

Thus, the goal of solving NESs is to find multiple roots in a single run. However, it is a bottleneck challenge in numerical computation.

Differential evolution (DE) is a kind of evolutionary algorithm (EA), which was presented by Storn and Price in 1997[6]. DE is a population-based optimization method, which finds the optima through the mutation, crossover, and selection operations. Because DE can be easily implemented, it is insensitive to functional characteristics, and can achieve rapid convergence, it has been successfully applied to many problems[7, 8]. Recently, several improved DE variants have been designed to effectively solve NESs[9]. These algorithms mainly focus on balancing exploration and exploitation. On one hand, diversity preservation

• Zuowen Liao is with Beibu Gulf Ocean Development Research Center, Beibu Gulf University, Qinzhou 535000, China. E-mail: liaozuowen@bbgu.edu.cn.
• Shuijia Li is with the School of Computer Science, China University of Geosciences, Wuhan 430074, China. E-mail: shuijiali@cug.edu.cn.
* To whom correspondence should be addressed.
Manuscript received: 2021-12-25; revised: 2022-01-19; accepted: 2022-02-17

techniques[10−13] can ensure that the algorithm adequately searches the feasible region, and finds more promising areas. On the other hand, several techniques, such as neighborhood size[11], mutation strategy selection[14], and local search[15], can facilitate the convergence of the algorithm.

Although the algorithms for solving NESs have made considerable progress, there are still some weaknesses. First, fewer efforts pay attention to individual states during the run. Second, parameter settings or mutation strategies need to be given in advance and easily fall into the local optima. Third, locating all of the roots of NESs within computational costs faces difficulties. Finally, how to efficiently balance exploration and exploitation is still a challenge. Reinforcement learning (RL) merged from research on machine learning and learned from interactions with the environment[16]. Environments, states, actions, and rewards constitute the elements of RL. In RL, the agent matches the actions based on the current state to obtain the maximum reward function[17]. In recent years, RL has been used to facilitate EAs to improve their problem-solving capabilities[18]. Hu and Gong proposed RL-CORCO to solve constrained optimization problems[19]. Hu et al. utilized RL to assist DE for parameter extraction of photovoltaic models[20]. Emary et al. combined RL, neural networks, and gray wolf optimization to deal with feature selection problems[21]. Liu et al. utilized RL to train the parameters in the particle swarm optimization algorithm[22]. Li et al. developed a new DE based on RL with fitness ranking to solve multimodal multiobjective optimization problems, and obtained competitive results[23]. More details can be found in Ref. [24]. Despite the successes in combining RL and EAs to solve the optimization problems, only a few researches on utilizing RL to assist EAs to solve NESs have been conducted. Commonly, combining RL and EAs to deal with NESs faces the following challenges: (1) Defining the states and characteristics of EAs is difficult; thus, agents need to perceive the corresponding environmental information and its dynamic changes. (2) How to design reasonable actions to improve the search efficiency of the algorithm is also a content worthy of research. (3) How to develop the reward function for actions is also a difficulty in RL. Based on the aforementioned considerations, we propose an enhanced RL based DE for solving NESs, where RL is used to control the operator of DE during the run. In this manner, we can alleviate trivial tasks to provide the appropriate actions for different individual states by trial-and-error. Our proposed approach is referred to as enhanced reinforcement learning based differential evolution (ERLDE). In ERLDE, the crowding technique is used to preserve population diversity. The combination between mutation strategies and neighborhood size is also used as the candidate actions that individuals can choose. Moreover, the unbalanced assignment method is beneficial to action selection. To evaluate the performance of ERLDE, 30 NESs problems and 18 test instances with different features are selected as the test suite. Furthermore, the numerical experiments are conducted in this study, including the comparison of several advanced algorithms, the impact of different actions, and the influence of reward functions in ERLDE.

The proposed ERLDE algorithm has the following contributions:

- The core contribution is the fusion of RL and DE to solve NESs. Through the interaction between agent and environment, the algorithm can select the appropriate actions autonomously, thus facilitating problem-solving.
- The fitness-based states, actions of the combination between mutation strategies and neighborhood size, and unbalanced assignment method, when combined synergistically, can significantly improve the search efficiency of the algorithm.
- Experiments on 30 NESs problems and 18 test sets with different features show that ERLDE achieves better root ratio (RR) and success rate (SR) results than state-of-the-art methods.

The remainder of this paper is organized as follows: Section 2 presents the transformation technique of NESs, RL, DE, and EAs for NESs. Section 3 introduces the proposed approach in detail. Section 4 states the test functions, evaluation criteria, and experimental results. Section 5 presents the discussions. Section 6 provides the results of solving the new test set. Finally, Section 7 summarizes this study.

2 Related Work

In this section, first, the transformation technique of NESs is presented. Then, RL, DE, and the methods for solving NESs are introduced.

2.1 Transformation technique

To solve NESs with EAs, it should be transformed into

optimization problems. Generally, single-objective transformation (SOT)[25] and multi-objective transformation[26] are two typical methods. Because of its simplicity and effectiveness, the SOT method is chosen to transform NESs into single-objective optimization problems:

$$\min \ F(X) = \sum_{i=1}^{n} f_i^2(X) \tag{3}$$

The optima of Eq. (3) represents the root of Eq. (1). To locate multiple roots of NESs, the diversity preservation technique requires the assistance of EAs when solving NESs.

2.2 Reinforcement learning

RL is a branch of machine learning, in which agents select actions according to the environment and obtain feedback[27]. In RL, the agents do not know the problem and learn through " trial and error". This process can be called the Markov decision processing (MDP). First, the agent randomly selects an action based on the current state. The original state may change after the action is performed, and the agent receives a reward or penalty value. Then, the value in the Q-table is changed according to Eq. (4) to provide a basis for the agent to select the appropriate action next time.

$$\begin{aligned} Q_{t+1}(s_t, a_t) = Q(s_t, a_t) + \alpha[R_{t+1} + \\ \gamma \max Q(s_{t+1}, 1) - Q(s_t, a_t)] \end{aligned} \tag{4}$$

where α is the learning rate and γ is the discount factor; $\alpha, \gamma \in [0, 1]$; R_{t+1} is the immediate reward; and $Q(s_t, a_t)$ is the cumulative reward obtained by the agent at time t.

In this study, Q-learning with ϵ-greedy policy is used to drive the agent to select the best action under the current state. Q-learning has a Q-table, which is composed of the value function $Q(s_t, a_t)$ corresponding to the state and action. Through $Q(s_t, a_t)$, the agent can obtain the optimal action. The process of RL is shown in Fig. 1.

2.3 Differential evolution

DE is a well-known heuristic algorithm consisting of three operations: mutation, crossover, and selection[6].

(1) **Mutation** : This operation is mainly used to generate a mutation vector. "DE/rand/1" and "DE/best/1" are two classic mutation strategies:

DE/rand/1:

$$V_i = X_{r1} + F \cdot (X_{r2} - X_{r3}) \tag{5}$$

DE/best/1:

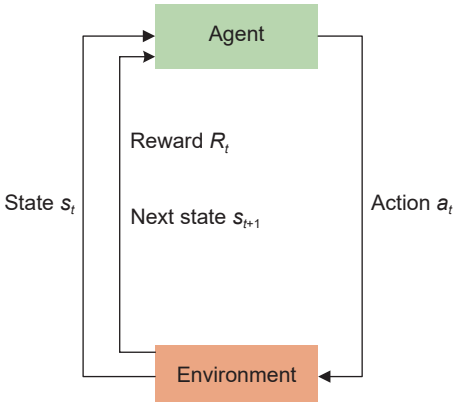

Fig. 1 Framework of RL.

$$V_i = X_{best} + F \cdot (X_{r1} - X_{r2}) \tag{6}$$

where $r1, r2,$ and $r3$ are the subscripts of randomly selected individuals, X_{best} is the best individual in the population, and F is the scale factor.

(2) **Crossover**: The target vector is obtained by the crossover operation of target vector X_i and mutation vector V_i.

$$U_{i,j} = \begin{cases} V_{i,j}, \text{ if } rand_j \leqslant CR \text{ or } j = j_{rand}; \\ X_{i,j}, \text{ otherwise} \end{cases} \tag{7}$$

where $rand_j$ is a uniform random number on the interval [0, 1]; CR is the crossover rate; and j_{rand} is a random integer from 1 to D.

(3) **Selection**:

$$X_i = \begin{cases} X_i, \text{ if } f(X_i) \leqslant f(U_i); \\ U_i, \text{ otherwise} \end{cases} \tag{8}$$

where $f(\cdot)$ is the individual fitness, and U_i is the trial vector. Through the comparison of fitness, a better individual is selected to enter the next evolution.

2.4 EAs for NESs

In past decades, solving NESs with EAs has attracted considerable attention from researchers[1]. According to the characteristics of transformation technology, the methods for solving NESs can be divided into single-objective based and multi-objective based methods. We will briefly review these methods in the subsequent subsections.

2.4.1 Single-objective based method

- **Repulsion-based method**: The repulsion-based method can generate a repulsion region near the found root to drive the algorithm to search other promising areas. Hirsch et al. proposed the continuous greedy randomized adaptive search procedure (CGRASP) with repulsion to locate the roots of NESs, in which CGRASP and local search phase were implemented[28]. Ramadas and Fernandes designed an improved

harmony search (I-HS)[29] and combined it with the repulsion technique for multiple roots location. In I-HS, the repulsion-based and re-initialization methods enhanced the population diversity, and harmony search was employed as a heuristic search algorithm to find the optima. Gong et al. developed a DE variant (RADE) that combined the crowding technique, repulsion, and parameter adaptation to find different roots[30]. However, the repulsion radius is difficult to set. To this end, Liao et al. presented the dynamic repulsion radius method and developed a generic framework with EAs, i.e., DREA, to enhance the performance in solving NESs[10].

- **Niching-based method**: Niching can change the search behavior of the algorithm to ensure that multiple optima can be sought simultaneously. Zhou and Jiang combined the deterministic crowding method, genetic algorithm, and quasi-Newton method to solve NESs[31]. Liao et al. developed a memetic niching based EA, called MENI-EA, to determine the roots of NESs[15]. In MENI-EA, the crowding technique can preserve population diversity, and four numerical methods for NESs refined the accuracy of the roots. In Ref. [11], the improved fuzzy neighborhood based crowding technique with orientation integrated into DE, referred to as FNODE, was utilized to locate multiple roots of NESs. Liao et al. proposed an enhanced ring topology to enhance the efficiency in exchanging neighborhood information. Combined with the swarm intelligence algorithm, it can significantly enhance the performance in problem-solving[12]. In Ref. [14], a decomposition-based DE with re-initialization (DDE/R) was designed to determine multiple roots of NESs. In DDE/R, the decomposition method can divide the population into different subpopulations; a subpopulation control strategy was employed to improve the search capability. Wu et al. developed a clustering-based DE to solve NESs[13]. Gao and Li proposed that NCODE and DBDE can be integrated into LSTP for solving the transformed problem of NESs and achieved promising results. Moreover, a new test set was designed in this article[32].

2.4.2 Multi-objective based method

The multi-objective technique requires obtaining a series of Pareto optimal solutions, which are similar to the multiple roots location. In recent years, the use of multi-objective techniques to solve NESs has become a hot topic.

Grosan and Abraham first transformed NESs into a multi-objective optimization problem and then employed NSGA-II to solve it[33]. Song et al.[26] proposed MONES to deal with NESs. In MONES, the NESs problem was transformed into a bi-objective optimization problem, and NSGA-II was employed to solve it[26]. However, MONES suffers from the "dimensional curse" problem. To overcome this shortcoming, Gong et al. developed a weighted bi-objective transformation technique (A-WeB) and achieved competitive performance in solving NESs[34]. Gao et al.[9] presented a two-phase EA (TPEA) to find the roots of NESs. In TPEA, the first stage mainly determined the high-quality solutions, and the second stage mainly enhanced the population diversity[9]. In Ref. [35], MOPEA, which combined the diversity indicator, multi-objective technique, and clustering method, was proposed to determine different roots of NESs.

3 Our Approach

In this section, first, we introduce the motivation of this study. Then, we describe our approach in detail.

3.1 Motivation

In RL, agents learn the appropriate behavior through trial and error during their interactions with environments. RL obtains the state information through the immediate interaction with the environment, and evaluates the action taken through the feedback reinforcement signal to learn the optimal strategy. Recently, RL-assisted DE algorithms used to deal with optimization problems have attracted considerable attention[24]. However, RL-assisted DE algorithms have several weakness in solving NESs. For example:

- **State definition**: The state definition should be given in advance by the user. In Ref. [20], the authors defined two states: "1" and "2". "1" indicates that the offspring's fitness is better than the parent's fitness, whereas "2" indicates that the offspring has worse fitness than the parent. The state definition only focuses on the comparison of fitness between offspring and parent, and ignores the gap between fitness and global optima.
- **Action design**: In Refs. [20, 22], the authors only considered changing different parameter values to assist the algorithm evolution. Notably, such an action design cannot solve NESs effectively.
- **Reward setting**: In Ref. [20], fixed reward values were assigned to the implemented actions according to the state reached. In Ref. [22], the dynamic change of reward value was used to define the level of reward or

punishment. However, these two methods still have several defects.

Motivated by the aforementioned considerations, in this study, we proposed the ERLDE to deal with NESs. In ERLDE, both state definition and action design introduce the NESs problem characteristics, and the actions are assigned using the unbalanced assignment method. In the subsequent subsection, we describe these methods in detail.

3.2 State definition

For each individual in DE, we propose a state definition method based on fitness as follows:

$$state_i = \begin{cases} 1, \text{ if } f(X_i) > 0 \text{ and } f(X_i) \leqslant 0.01; \\ 2, \text{ if } f(X_i) > 0.01 \text{ and } f(X_i) \leqslant 0.25; \\ 3, \text{ if } f(X_i) > 0.25 \text{ and } f(X_i) \leqslant 0.5; \\ 4, \text{ otherwise} \end{cases} \tag{9}$$

where $f(X_i)$ is the fitness of X_i and " 1, 2, 3, 4" represent different states. Specifically, "1" indicates that the individual fitness value is close to the optima and represents the "smallest" state; "4" indicates that the individual fitness value differs significantly from the optima and represents the "largest" state; and "2" and "3" indicate the "smaller" and "larger" states, respectively.

Notably, this method draws on the ideas presented in Ref. [22], but it is different from it in terms of the following aspects:

- In Ref. [22], ΔF controls the range of fitness values to divide the state. However, the division according to ΔF may lead to the problem of nonuniform states in solving NESs. Therefore, based on our experience, we directly use fixed fitness values to divide the state.
- The state definition based on fitness helps individuals perceive the change of fitness and facilitate subsequent action selection.

3.3 Action design

Action design ensures that the agent has the abilities to complete the target task. In DE, different mutation operations have different search characteristics and can assist individuals in exploring the search region[36]. Moreover, neighborhood size plays an important role in solving NESs[11]. To this end, we propose the combination of actions, which is described in detail as follows:

First, "DE/rand/1" and "DE/best/1" are used as two basic mutation strategies. Generally, "DE/rand/1" has a strong exploration capability but slowly converges. By contrast, "DE/best/1" rapidly converges, but it easily falls into the local optima.

To facilitate individuals in applying mutation strategies to search promising regions, different neighborhood sizes are adopted. Specifically, the two neighborhood sizes of "DE/rand/1" are "5" and "8", whereas the two neighborhood sizes of "DE/best/1" are "3" and "*NP*". Thus, the combined action can be expressed as follows:

$$action_j = \begin{cases} \text{DE/rand/1,5}; \\ \text{DE/rand/1,8}; \\ \text{DE/best/1,3}; \\ \text{DE/best/1,}NP \end{cases} \tag{10}$$

According to the states and actions described previously, the Q-table is shown in Table 1.

In Table 1, where a_1, a_2, a_3, and a_4 represent different actions shown in Eq. (10). Equation (4) is used to update the corresponding Q values. Moreover, the softmax strategy determines the probability of selecting action a_j in state s_i:

$$\pi(s_i, a_j) = \frac{e^{Q_t(s_i,a_j)/T}}{\sum_{j=1}^{m} e^{Q_t(s_i,a_j)/T}} \tag{11}$$

where $Q_t(s_i, a_j)$ represents the corresponding Q values obtained in the Q-table at time t and T is a parameter.

3.4 Reward setting

After performing a certain action, the agent will receive a reward to indicate the effectiveness of that action. In this study, unbalanced assignment method assigns a reward value to the implemented action. Depending on the state, the reward value is assigned differently:

$$reward = \begin{cases} 2.0, \text{ if } state = 1; \\ 1.5, \text{ if } state = 2; \\ 1.0, \text{ if } state = 3; \\ 0.5, \text{ if } state = 4 \end{cases} \tag{12}$$

If the fitness of the offspring is better than that of the parent, then the actions in different states can be assigned according to Eq. (12).

Table 2 shows the reward values for the different states. Notably, the given reward is different in

Table 1 Form of the Q-table.

State	Value of different actions in different states			
	Action a_1	Action a_2	Action a_3	Action a_4
Smallest	$Q(1,1)$	$Q(1,2)$	$Q(1,3)$	$Q(1,4)$
Smaller	$Q(2,1)$	$Q(2,2)$	$Q(2,3)$	$Q(2,4)$
Larger	$Q(3,1)$	$Q(3,2)$	$Q(3,3)$	$Q(3,4)$
Largest	$Q(4,1)$	$Q(4,2)$	$Q(3,4)$	$Q(4,4)$

Table 2 Reward values of different actions in different states.

State	Reward value			
	Next state "1"	Next state "2"	Next state "3"	Next state "4"
"1"	2	0	0	0
"2"	2	1.5	0	0
"3"	2	1.5	1.0	0
"4"	2	1.5	1.0	0.5

different states because of the following reasons: On one hand, for the states with high fitness, a small reward is assigned to the actions that have been performed because the fitness in this state is easy to change, and the small reward ensures that the algorithm has a higher probability of selecting other actions. On the other hand, assuming that the individual fitness is low, the difficulty of changing its fitness increases. Once the selected action is conducive to the improvement of fitness, a larger reward value can be assigned so that there is a higher probability of selecting this action in the subsequent evolution.

3.5 Framework of ERLDE

The framework of ERLDE is demonstrated in Algorithm 1, where NP is the population size; NFE is the number of function evaluations; $NFEs_{\max}$ is the maximal NFE; and $\mathcal{A}$ is an archive that saves the found roots. ERLDE works as follows:

- To enhance the population diversity, the neighborhood based crowding technique proposed in Ref. [37] is used in ERLDE. The neighborhood-based crowding technique adopts the Euclidean distance to select individuals in the neighborhood for mutation, which enables the algorithm to simultaneously search for potential regions that may have roots.
- In Line 7, the action is selected by using Eq. (11). In Lines 8 and 9, n individuals closest to X_i are selected to form the neighborhood population, and the offspring U_i is generated by DE operations.
- In Lines 13−22, through the comparison of fitness, the reward is assigned to the selected action, and the state of X_i is updated. If the fitness of X_i is poor, then the reward given is 0. In Line 23, the values of the Q-table are updated using Eq. (4).
- In Lines 27−29, the found root X_i is saved in $\mathcal{A}$ and reinitializes it. In Lines 30 and 31, the state and Q-table are redetermined.

In Algorithm 1, each individual has its state, Q-table, and reward, which enable ERLDE to perceive the changes of the current individual fitness and guide the next evolution to improve the search efficiency of ERLDE. Figure 2 shows the basic flow of ERLDE.

Algorithm 1 Framework of ERLDE

Input: $iter$, NFE, $NFEs_{\max}$
Output: The final archive $\mathcal{A}$.
1: Randomly initialize the population;
2: All elements in the Q-table are specified as 0;
3: Assess fitness of the population;
4: $NFE = NFE + NP$;
5: **While** $NFE < NFEs_{\max}$ **do**
6: **for** i=1: NP **do**
7: Choose an action for each individual from Q-table according to Eq. (11);
8: n individuals closest to X_i are selected to form neighborhood population;
9: Generate the offspring U_i by using mutation, crossover and selection operations;
10: Calculate the fitness of U_i;
11: $NFE = NFE + 1$;
12: **end for**
13: **for** i=1: NP **do**
14: Select the individual X_j closest to U_i from the population;
15: **if** $f(U_i) \leqslant f(X_j)$ **then**
16: $X_j = U_i$;
17: $f(X_j) = f(U_i)$;
18: Assign reward value to the action according to Eq. (12);
19: Update the state of X_i;
20: **else**
21: The reward value is assigned to 0;
22: **end if**
23: Update Q-table with Eq. (4);
24: **end for**
25: **for** i=1: NP **do**
26: **if** $f(X_i) \leqslant \epsilon$ **then**
27: Save X_i into archive $\mathcal{A}$;
28: Re-initialize X_i;
29: Calculate the fitness of X_i;
30: Determine the state of X_i;
31: All elements in the Q-table of X_i are initialized to 0;
32: $NFE = NFE + 1$;
33: **end if**
34: **end for**
35: $iter = iter + 1$
36: **end while**

4 Experimental Results and Analysis

In this section, a comprehensive experimental

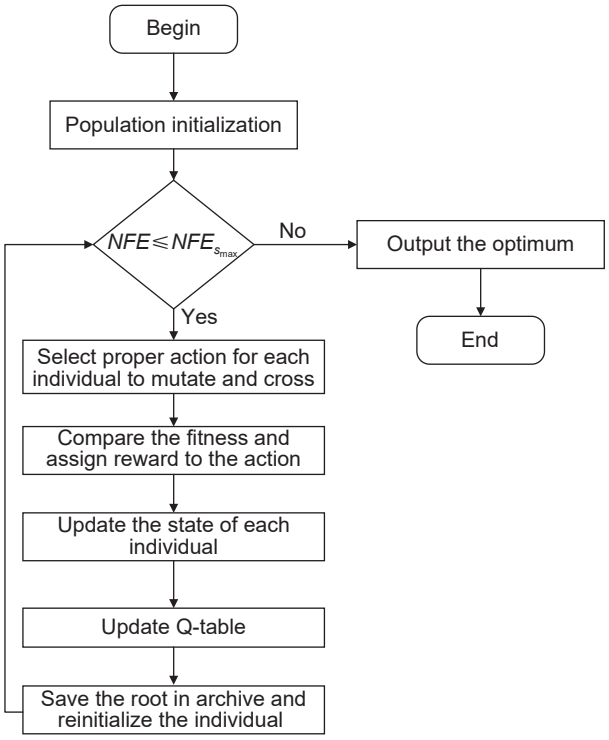

Fig. 2 Flowchart of ERLDE.

comparison is used to evaluate the superiority of the proposed ERLDE algorithm. First, the experimental setting and evaluation criteria are introduced. Second, the results of ERLDE are compared with those of other algorithms and analyzed.

4.1 Experimental setting

The 30 NESs problems from Ref. [30] are used to evaluate the performance of ERLDE. Their characteristics are listed in Table A1 in the Appendix. Moreover, two commonly used evaluation criteria, i.e., *RR* and *SR*, are employed to assess the performance of ERLDE and other comparison algorithms. *RR* calculates the average ratio of roots in all of the runs, whereas *SR* denotes the probability of successfully finding all roots in multiple runs. For more detailed, the reader can refer to Ref. [14]. All the algorithms are executed 30 times. The Friedman and the Wilcoxon tests are utilized to compare the performance of different algorithms.

4.2 Comparison algorithms

In the experiment, ERLDE is compared with 11 algorithms. The main features are as follows:

● **MONES**[26] : A bi-objective transformation technique with NSGA-II;

● **A-WeB**[34] : A weighted bi-objective transformation technique (A-WeB) with DE;

● **RADE**[30]: The combination of niching technique, repulsion-based method, and parameter setting;

● **DREA**[10] : An EA framework based on the dynamic repulsion technique;

● **FNODE**[11]: Fuzzy neighborhood size with DE;

● **KSDE**[13]: A clustering-based DE;

● **DDE/R**[14] : A decomposition-based DE with reinitialization;

● **TPEA**[35]: A two-phase EA;

● **ANDE**[38] : Automatic niching DE with contour prediction;

● **LBPADE**[39]: Local binary pattern-based adaptive DE;

● **LSTP**[32]: A new two-phase algorithm.

Among the aforementioned methods, MONES, A-WeB, and TPEA are multi-objective-based methods; RADE and DREA are repulsion-based methods; FNODE, KSDE, DDE/R, and LSTP are niching-based methods; and ANDE and LBPADE are the methods for solving multimodal optimization problems. The related parameters are shown in Table 3.

4.3 Interpretation of experimental results

The detailed results of *RR* and *SR* are given in Tables A2 and A3 in the Appendix, respectively. Moreover, the Friedman and the Wilcoxon tests are shown in Tables 4 and 5, respectively. The detailed analysis of

Table 3 Parameter settings for different algorithms.

Algorithm	Parameter settings
ERLDE	$NP = 100,\ F = 0.5, CR = 0.9, \alpha = 0.1, \gamma = 0.9$
MONES	$NP = 100, H_m = NP$
A-WeB	$NP = 100, H_m = NP$
RADE	$NP = 100, H_m = 200$
DREA	$u_{CR} = 0.5, u_F = 0.5, c = 0.1$
FNODE	$NP = 100, F = 0.5, CR = 0.9, m = 11$
KSDE	$NP = 100, F = 0.5, CR = 0.9, \theta = 0.5$
DDE/R	$NP = 100, F = 0.5, CR = 0.9, t = 20, \ell = 20$
TPEA	$NP = 100, F = 0.5, CR = 0.7, nt = 20, \lambda = 5$
ANDE	$F = 0.9, CR = 0.1$
LBPADE	$NP = 100, b = 0.9, a = 0.1, n = 0.9, m = 0.1, \mu = 1 \times 10^{-4}$
LSTP	$NP = 100,\ F = 0.5, CR = 0.9, K = 3, c = 0.8$

Note: H_m is the number of individuals in the parent and offspring after merging. u_{CR} is a parameter that generates a crossover rate for each individual. u_F is a parameter that generates a mutation factor for each individual. c is a positive constant between 0 and 1. ℓ is the maximum number of individuals in the generated sub population. λ is the neighborhood size. b is the upper boundary of mutation factor. a is the lower boundary of mutation factor. μ is a parameter that controls the way of dealing with boundary. K is the neighborhood size. And c is a certain constant in [0,1] to allocate computer resources for each phase.

Table 4 Average rankings of 12 algorithms obtained by the Friedman test in terms of root ratio (*RR*) and success rate (*SR*).

Algorithm	Ranking (*RR*)	Ranking (*SR*)
ERLDE	**4.8167**	**4.6667**
MONES	9.4833	9.2000
A-WeB	8.6167	8.5333
RADE	7.2333	7.2500
DREA	6.5333	6.8833
FNODE	5.4167	5.3500
KSDE	6.0000	6.0833
DDE/R	5.0167	5.1167
TPEA	4.9167	4.9000
ANDE	6.9833	6.8667
LBPADE	5.5167	5.6167
LSTP	7.4667	7.5333

Table 5 Wilcoxon test results obtained according to the *RR* and *SR* of different algorithms.

Algorithm	*RR*			*SR*		
	R^+	R^-	p-value	R^+	R^-	p-value
MONES	406.0	59.0	$\mathbf{1.49\times10^{-4}}$	407.5	57.5	**0.00**
A-WeB	387.5	77.5	$\mathbf{6.76\times10^{-4}}$	395.0	70.0	$\mathbf{2.81\times10^{-4}}$
RADE	376.0	89.0	$\mathbf{5.40\times10^{-4}}$	378.0	87.0	$\mathbf{2.34\times10^{-3}}$
DREA	332.5	132.5	$\mathbf{3.77\times10^{-2}}$	354.5	110.5	$\mathbf{6.33\times10^{-3}}$
FNODE	304.0	161.0	1.37×10^{-1}	306.0	159.0	1.26×10^{-1}
KSDE	323.0	142.0	5.40×10^{-2}	327.5	137.5	$\mathbf{4.54\times10^{-2}}$
DDE/R	268.5	195.5	4.17×10^{-1}	270.0	195.0	$\mathbf{4.09\times10^{-2}}$
TPEA	282.5	182.5	2.96×10^{-1}	285.5	179.5	2.69×10^{-1}
ANDE	348.5	116.5	$\mathbf{1.60\times10^{-2}}$	350.5	114.5	$\mathbf{1.30\times10^{-2}}$
LBPADE	294.5	170.5	1.75×10^{-1}	295.0	170.0	1.67×10^{-1}
LSTP	349.0	116.0	$\mathbf{9.12\times10^{-3}}$	366.0	99.0	$\mathbf{1.61\times10^{-3}}$

Note: R^+ is the rank sum for the functions in which our algorithm is superior to other algorithms, R^- is the sum of ranks for the opposite, and p-value is to judge whether the two methods are significantly different.

the results is as follows:

● Tables A2 and A3 in the Appendix show that ERLDE obtains the best *RR* and *SR*, i.e., $\{0.99, 0.95\}$, followed by DDE/R and TPEA. Moreover, Table 4 shows that ERLDE achieves the best ranking compared with the other methods in the Friedman test.

● Table 5 shows that, in the Wilcoxon test, the statistical results obtained by ERLDE are significantly better than those obtained by MONES, A-WeB, RADE, DREA, ANDE, and LSTP because all p-values are $\leqslant 0.05$. Moreover, compared with those obtained by FNODE, KSDE, DDE/R, TPEA, and LBPADE, the results obtained by ERLDE are not significantly different. However, ERLDE provides higher R^+ values than R^- values. Thus, ERLDE is better than these five methods.

● Although ERLDE has achieved superior results, there are still several shortcomings. Table A2 in the Appendix shows that ERLDE locates all roots in 30 NESs problems, except F04, F12, F13, and F17. For F04, A-WeB, DREA, DDE/R, and TPEA can locate all of the roots. Moreover, DDE/R can successfully solve F12. For F13, KSDE finds all roots. RADE, DREA, FNODE, DDE/R, TPEA, ANDE, and LBPADE achieve better *RR* than ERLDE. For F17, DREA, KSDE, and DDE/R determine all roots whereas ERLDE misses some roots. Thus, ERLDE performs poorly on function that contains more roots (i.e., F04, F12, F13, and F17). The reason may be that the design of the action may be defective, resulting in the inadequate search performance of ERLDE. In the future, we can learn from the characteristics of other algorithms, such as FNODE, KSDE, and TPEA, to design more reasonable actions to enhance the capability of ERLDE to solve NESs.

5 Discussion

In Section 4, the superiority of ERLDE has been analyzed. In this section, we mainly study the influence of different action designs and reward functions on the performance of ERLDE in solving NESs.

5.1 Influence of different action designs

ERLDE uses a combinational approach to design actions, and combines mutation strategies and neighborhood size as the actions to be performed by the agent. To better understand the performance of ERLDE, the influence of different action designs is discussed. For this purpose, ERLDE is compared with the following methods:

● **ERLDE-1**: only the fixed neighborhood size "5" is used;

● **ERLDE-2**: only the fixed neighborhood size "10" is used;

● **ERLDE-3**: only the fixed neighborhood size "15" is used;

● **ERLDE-4**: only the fixed neighborhood size "20" is used;

● **ERLDE-5**: only "DE/rand/1" is used;

● **ERLDE-6**: only "DE/best/1" is used.

Notably, all other parameters remain the same for a fair comparison.

The detailed results of *RR* and *SR* are given in Tables A4 and A5 in the Appendix. The statistical results of the Friedman and Wilcoxon tests are presented in Tables 6 and 7, respectively. The results are analyzed as follows:

● **ERLDE-1 to ERLDE-4**: In these four ERLDE variants, different neighborhood sizes are adopted. As shown in Tables A4 and A5 in the Appendix, the best *RR* and *SR* are obtained when the neighborhood size is equal to "10". Meanwhile, with the increase of neighborhood size, the performance of different ERLDE variants decreases gradually, which indicates that the neighborhood size has an important influence on the performance of the algorithm. Specifically, these four methods behave differently in several functions. For F04, ERLDE-1 locates all roots, whereas ERLDE-2, ERLDE-3, and ERLDE-4 miss some roots. For F12, ERLDE-1 performs well, whereas ERLDE-2, ERLDE-3,andERLDE-4performpoorly.ForF17andF25,ERLDE-2, ERLDE-3, and ERLDE-4 successfully find all roots while ERLDE-1 misses some roots. The analysis shows that low dimension and multiple roots are the characteristics of F03, F12, and F22. Therefore, different neighborhood sizes have a significant impact on the search performance of the algorithm for different functional terrain.

● **ERLDE-5 and ERLDE-6**: In these two ERLDE variants, different mutation strategies are adopted. ERLDE-5 obtains better *RR* and *SR* than ERLDE-6. Concretely, ERLDE-5 performs better than ERLDE-6 on F01 and F17. However, ERLDE-6 obtains better *RR* than ERLDE-5 on F04 and F12. In general, ERLDE-5 and ERLDE-6 perform differently in different functions.

● As shown in Tables A4 and A5 in the Appendix, ERLDE obtains the best *RR* and *SR*, followed by ERLDE-2 and ERLDE-5. Moreover, Table 6 indicates that ERLDE achieves the best ranking in both *RR* and *SR*. Meanwhile, Table 7 shows that ERLDE provides higher R^+ than R^- in all cases, which indicates that ERLDE has better performance than the other algorithms. Thus, combining mutation strategies and neighborhood size with actions is an effective method to improve algorithm performance in solving NESs.

Figure 3 illustrates the evolution of action selection on F04. The X-axis denotes the number of iterations (*Iter*), and the Y-axis denotes the number of uses of different actions (*Num_actions*). Figure 3a shows that

Table 6 Average rankings of different ERLDE variants obtained by the Friedman test in terms of *RR* and *SR*.

Algorithm	Ranking (*RR*)	Ranking (*SR*)
ERLDE	**3.7000**	**3.8000**
ERLDE-1	4.0333	4.0833
ERLDE-2	3.8167	3.8500
ERLDE-3	3.9667	3.9500
ERLDE-4	4.3667	4.2667
ERLDE-5	3.9333	3.8833
ERLDE-6	4.1833	4.1667

Table 7 Wilcoxon test results obtained according to the *RR* and *SR* of different ERLDE variants.

Algorithm	*RR*			*SR*		
	R^+	R^-	*p*-value	R^+	R^-	*p*-value
ERLDE-1	289.5	175.5	2.36×10^{-1}	274.5	190.5	3.58×10^{-1}
ERLDE-2	246.5	218.5	7.53×10^{-1}	233.0	232.0	9.83×10^{-1}
ERLDE-3	262.5	202.5	5.30×10^{-1}	247.5	217.5	7.37×10^{-1}
ERLDE-4	300.0	165.0	1.41×10^{-1}	289.5	175.5	2.36×10^{-1}
ERLDE-5	262.5	202.5	5.30×10^{-1}	247.5	217.5	7.37×10^{-1}
ERLDE-6	290.0	175.0	2.32×10^{-1}	275.0	190.0	3.51×10^{-1}

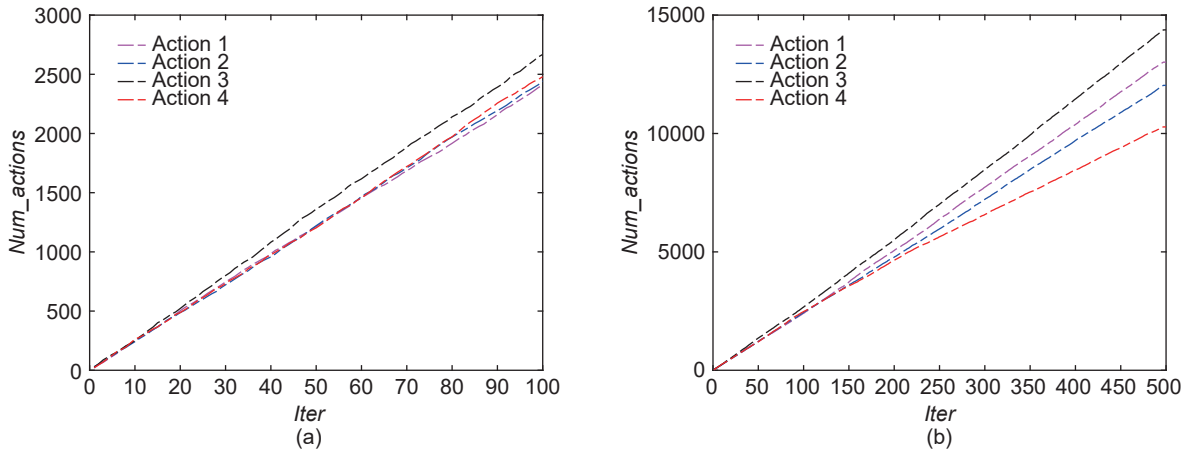

Fig. 3 Comparison of the used number of different actions.

action 3 was the most frequently used, followed by action 4. This scenario indicates that, in the initial stage, adopting the "DE/best/1" : strategy can achieve rapid convergence. Because of the high fitness of many individuals in the early stage of evolution, action 4 is also a common strategy of ERLDE. Figure 3b shows that the number of choices for action 3 continues to increase. Moreover, the selection times of action 1 and action 2 exceed that of action 4. The main reasons are as follows: First, to accelerate the convergence of ERLDE, action 3 is continuously adopted to refine individual fitness. Second, in the middle and late stages of evolution, ERLDE selects action 2 and action 3 for the mutation to improve the diversity of the population. Third, because the individuals fitness was relatively low in the subsequent period, the number of action 4 was gradually reduced.

5.2 Impact on the reward function

In RL, agents continuously improve strategies according to the reward function from the environment during exploration. It represents the quality of the state obtained by the agent after performing the action. This section discusses the effect of different reward functions on algorithm performance.

In ERLDE, the unbalanced assignment method specifies different reward values according to different states to improve search efficiency. This section adopts the method of balanced allocation, that is, if the fitness of the offspring is better than that of the parent, then the reward value of the action is 1. The corresponding reward values are shown in Table 8. For a fair comparison, the other parameters of ERLDE remain unchanged.

The detailed results of *RR* and *SR* are shown in Table A6 in the Appendix. The results of the Wilcoxon test are given in Table 9. Table A6 shows that ERLDE obtains better *RR* and *SR* than ERLDE-7. Specifically, ERLDE achieves better results than ERLDE-7 on F12, F13, and F17. As shown in Table 9, ERLDE provides higher R^+ than R^- in all cases, which shows that ERLDE obtains superior results. The reasons may be that the unbalanced assignment method is conducive to the choice of action. Generally, the individual fitness is high and can be easily changed in the initial stage. Assigning a low reward value to a certain action enables ERLDE to select other actions. When the individual's fitness is low, it is difficult to change. In this case, if large value is assigned, then ERLDE will have a high probability of selecting the action in the subsequent process.

Table 8 New reward values of different actions in different states.

State	Reward values			
	Next state "1"	Next state "2"	Next state "3"	Next state "4"
"1"	1	0	0	0
"2"	1	1	0	0
"3"	1	1	1	0
"4"	1	1	1	1

Table 9 Results obtained by the Wilcoxon test for ERLDE in terms of *RR* and *SR* compared with ERLDE-7.

Comparison results	R^+	R^-	*p*-value
RR	300.0	165.0	1.41×10^{-1}
SR	289.5	175.5	2.36×10^{-1}

6 Solving the New Test Set

In Ref. [32], a set of 18 NESs (MNE1-MNE18), which is more complex than the original test sets, is proposed to simulate real-world problems (Table A1 in the Appendix). Seven algorithms, i.e., LSTP, DREA, MONES, A-WeB, FNODE, DDE/R, and EMO-MMO[40], are selected for comparison with ERLDE.

The detailed results of *RR* and *SR* are shown in Tables A7 and A8 in the Appendix, respectively. Moreover, the statistical results of the Friedman and Wilcoxon tests are given in Tables 10 and 11. From the results we can deduce that:

- Tables A7 and A8 in the Appendix show that LSTP achieves the best *RR* and *SR*, followed by ERLDE. Concretely, LSTP is better than ERLDE on MNE3, MNE7, MNE8, MNE11, MNE13, MNE14, MNE16, and MNE18. ERLDE performs well on MNE9, MNE10, and MNE15. LSTP is better than ERLDE in terms of obtaining the average value of *RR* and *SR*, as well as the performance on different functions. However, compared with DREA, MONES, A-WeB, FNODE, DDE/R, and EMO-MMO, ERLDE achieves better results.
- Table 10 shows that LSTP obtains the best ranking in terms of *RR* and *SR*, followed by ERLDE. Moreover, ERLDE provides higher R^+ than R^- in all cases, except for LSTP. The statistical results indicate that LSTP is the best, followed by ERLDE.
- LSTP proposed a new transformation technique on the logarithmic to solve NESs, which can locate the optimal solution in a steep area. However, the values of *RR* and *SR* obtained by LSTP are worse than those

Table 10 Average rankings of ERLDE, LSTP, DREA, MONES, A-WeB, FNODE, DDE/R, and EMO-MMO in *RR* and *SR* by the Friedman test.

Algorithm	Ranking (*RR*)	Ranking (*SR*)
ERLDE	3.1389	4.0833
LSTP	2.1667	3.6111
DREA	4.8611	4.5278
MONES	4.8889	4.9167
A-WeB	5.6944	4.7778
FNODE	3.8056	4.4444
DDE/R	5.3889	4.7222
EMO-MMO	6.0556	4.9167

Table 11 Results obtained by the Wilcoxon test for LSTP, DREA, MONES, A-WeB, FNODE, DDE/R, and EMO-MMO in terms of *RR* and *SR* compared with ERLDE.

Algorithm	*RR*			*SR*		
	R^+	R^-	p-value	R^+	R^-	p-value
LSTP	61.0	110.0	1.00	53.0	100.0	1.00
DREA	129.0	42.0	5.40×10^{-1}	101.5	69.5	4.69×10^{-1}
MONES	137.0	34.0	$\mathbf{2.27\times10^{-2}}$	118.5	52.5	1.42×10^{-1}
A−WeB	141.5	29.5	$\mathbf{7.48\times10^{-3}}$	118.5	52.5	1.42×10^{-1}
FNODE	115.5	55.5	1.84×10^{-1}	94.0	77.0	6.70×10^{-1}
DDE/R	140.0	31.0	$\mathbf{1.60\times10^{-2}}$	108.0	63.0	2.73×10^{-1}
EMO-MMO	161.0	10.0	$\mathbf{8.75\times10^{-4}}$	118.5	52.5	1.42×10^{-1}

obtained by ERLDE, as shown in Tables A2 and A3 in the Appendix. Notably, the algorithm design of ERLDE is unsuitable for log-based transformation technique. Therefore, ERLDE does not perform as well as LSTP in the new test set. Even so, compared with DREA, MONES, A-WeB, FNODE, DDE/R, and EMO-MMO, whether the original test set or the new test set, ERLDE shows better performance, which reflects the superiority of ERLDE to some extent. In summary, according to the shortcomings of ERLDE, reasonable algorithm design based on different transformation techniques can be investigated in the future.

7 Conclusion

In this study, the ERLDE is proposed to solve NESs. In ERLDE, different states are defined based on individual fitness. Moreover, a combination of mutation strategy and neighborhood size is proposed as an alternative action for agents. Furthermore, the unbalanced assignment method is adopted to change the reward value to select optimal actions. To verify the performance of ERLDE, 30 NESs problems and 18 new test sets are selected. The experimental results indicate that ERLDE can provide superior performance on *RR* and *SR*.

Notably, how to design the state, action, and reward functions in RL to assist EAs in solving NESs is still a problem to be solved. Specifically, how to integrate the characteristics of nonlinear equations into RL to improve its efficiency is also a future research direction.

Appendix

Characteristics of the new test set are shown in Table A1. Tables A2 and A3 show the comparison between ERLDE and other algorithms with respect to *RR* and *SR*, respectively. Tables A4 and A5 show the comparison between ERLDE and other ERLDE variants with respect to *RR* and *SR*, respectively. Comparison between ERLDE and ERLDE-7 with respect to *RR* and *SR* is shown in Table A6. Tables A7 and A8 show the comparison of different algorithms on

Table A1 Brief information of the test problems.

Problem	m	Range	*LE*	*NE*	*NoR*	$NFEs_{\max}$
F01	20	$[-1,1]^n$	0	2	2	50 000
F02	2	$[-1,1]^n$	1	1	11	50 000
F03	2	$[-1,1]^n$	0	2	15	50 000
F04	2	$[-10,10]^n$	0	0	13	50 000
F05	10	$[-2,2]^n$	0	10	1	50 000
F06	2	$[-1,1]^n$	1	1	8	50 000
F07	2	$[-1,1],[-10,0]$	0	2	2	50 000
F08	2	$[0,1]^n$	0	2	7	50 000
F09	5	$[-10,10]^n$	4	1	3	100 000
F10	3	$[-5,5],[-1,3],[-5,5]$	0	3	2	50 000
F11	2	$[-1,1],[-10,10]$	0	2	4	50 000

(to be continued)

Table A1 Brief information of the test problems. (continued)

Problem	*m*	Range	*LE*	*NE*	*NoR*	$NFEs_{\max}$
F12	2	$[-1,2]^n$	0	2	10	50 000
F13	3	$[-0.6,0.6],[-0.6,0.6],[-5,5]$	0	3	12	50 000
F14	2	$[-5,5]^n$	0	2	9	50 000
F15	2	$[0.25,1],[1.5,2\pi]$	0	2	2	50 000
F16	2	$[0,2\pi]^n$	0	2	13	50 000
F17	8	$[-1,1]^n$	1	7	16	100 000
F18	2	$[-2,2]^n$	0	2	6	50 000
F19	20	$[-2,2]^n$	19	1	2	200 000
F20	3	$[-1,1]^n$	0	3	7	50 000
F21	2	$[-2,2]^n$	0	2	4	50 000
F22	2	$[-2,2]^n$	0	2	6	50 000
F23	3	$[-20,20]^n$	0	3	16	500 000
F24	3	$[0,1]^n$	0	3	8	100 000
F25	3	$[-3,3]^n$	0	3	2	50 000
F26	2	$[-1,-0.1],[-2,2]$	0	2	2	50 000
F27	2	$[-5,1.5],[0,5]$	0	2	3	50 000
F28	2	$[0,2],[10,30]$	0	2	2	50 000
F29	3	$[0,2],[-10,10],[-1,1]$	0	3	5	50 000
F30	2	$[-2,2],[0,1.1]$	0	2	4	50 000

Note: *m* is the dimension size, *LE* is the number of linear equations, *NE* is the number of nonlinear equation, and *NoR* is the number of roots.

Table A2 Comparison between ERLDE and other algorithm with respect to *RR*.

Problem	ERLDE	MONES	A-WeB	RADE	DREA	FNODE	KSDE	DDE/R	TPEA	ANDE	LBPADE	LSTP
F01	**1.00**	0.98	0.62	**1.00**	0.00	**1.00**	**1.00**	0.93	0.92	0.02	0.86	**1.00**
F02	**1.00**	0.97	1.00	0.99	0.99	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.95
F03	**1.00**	0.94	0.95	0.99	0.95	**1.00**	**1.00**	**1.00**	**1.00**	0.99	1.00	0.98
F04	0.99	0.59	**1.00**	0.91	**1.00**	0.96	0.98	**1.00**	**1.00**	0.89	0.98	0.83
F05	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.99	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F06	**1.00**	**1.00**	0.94	0.99	0.95	0.99	0.98	**1.00**	**1.00**	0.95	**1.00**	0.96
F07	**1.00**	0.83	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.98
F08	**1.00**	0.77	0.83	0.99	**1.00**	**1.00**	0.90	**1.00**	**1.00**	0.97	**1.00**	0.97
F09	**1.00**	0.00	0.89	0.97	1.00	1.00	0.99	0.98	1.00	0.98	**1.00**	0.98
F10	**1.00**	0.50	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F11	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.91	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F12	0.97	0.43	0.88	0.63	0.87	0.86	0.99	**1.00**	0.94	0.92	0.72	0.64
F13	0.64	0.54	0.09	0.89	0.91	0.73	**1.00**	0.72	0.92	0.78	0.81	0.52
F14	**1.00**	0.19	0.97	0.98	**1.00**	**1.00**	0.99	**1.00**	**1.00**	0.99	**1.00**	0.98
F15	**1.00**	**1.00**	**1.00**	**1.00**	0.50	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.98
F16	**1.00**	**1.00**	**1.00**	0.99	**1.00**	**1.00**	0.93	**1.00**	**1.00**	0.89	**1.00**	**1.00**
F17	0.98	0.08	0.66	0.94	**1.00**	0.98	**1.00**	**1.00**	0.98	0.77	0.85	0.85
F18	**1.00**	**1.00**	0.94	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F19	**1.00**	0.00	0.62	0.79	0.00	0.93	**1.00**	0.90	0.72	**1.00**	0.90	**1.00**
F20	**1.00**	0.71	0.95	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.99
F21	**1.00**	**1.00**	0.99	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**

(to be continued)

Table A2 Comparison between ERLDE and other algorithm with respect to *RR*. (continued)

Problem	ERLDE	MONES	A-WeB	RADE	DREA	FNODE	KSDE	DDE/R	TPEA	ANDE	LBPADE	LSTP
F22	**1.00**	0.98	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F23	**1.00**	0.14	0.15	0.56	0.79	0.91	0.92	0.94	0.98	0.55	0.70	0.54
F24	**1.00**	0.73	0.85	0.99	0.91	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F25	**1.00**	0.31	0.23	0.83	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.95	**1.00**	**1.00**
F26	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F27	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F28	**1.00**	0.00	0.94	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.40	**1.00**	**1.00**
F29	**1.00**	0.14	0.93	0.99	0.95	0.99	0.50	**1.00**	**1.00**	**1.00**	**1.00**	0.96
F30	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Avg.	**0.99**	0.66	0.85	0.95	0.89	0.98	0.97	0.98	0.98	0.90	0.96	0.94

Table A3 Comparison between ERLDE and other algorithm with respect to *SR*.

Problem	ERLDE	MONES	A-WeB	RADE	DREA	FNODE	KSDE	DDE/R	TPEA	ANDE	LBPADE	LSTP
F01	**1.00**	0.96	0.36	**1.00**	0.00	**1.00**	**1.00**	0.86	0.82	0.00	0.73	**1.00**
F02	**1.00**	0.76	**1.00**	0.90	0.93	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.56
F03	**1.00**	0.43	0.58	0.95	0.46	**1.00**	**1.00**	**1.00**	**1.00**	0.90	**1.00**	0.80
F04	0.87	0.00	**1.00**	0.31	**1.00**	0.52	0.80	**1.00**	**1.00**	0.10	0.86	0.04
F05	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.96	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F06	**1.00**	**1.00**	0.60	0.93	0.66	0.98	0.86	**1.00**	**1.00**	0.65	**1.00**	0.72
F07	**1.00**	0.66	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.96
F08	**1.00**	0.00	0.12	0.98	**1.00**	**1.00**	0.40	**1.00**	**1.00**	0.85	**1.00**	0.84
F09	**1.00**	0.00	0.68	0.91	**1.00**	**1.00**	0.93	0.93	**1.00**	0.95	**1.00**	0.96
F10	**1.00**	0.03	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F11	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.43	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F12	0.93	0.00	0.28	0.00	0.03	0.28	0.93	**1.00**	0.40	0.55	0.06	0.00
F13	0.07	0.06	0.00	0.19	0.26	0.02	**1.00**	0.00	0.36	0.05	0.06	0.00
F14	**1.00**	0.00	0.76	0.89	**1.00**	**1.00**	0.93	**1.00**	**1.00**	0.95	**1.00**	0.88
F15	**1.00**	**1.00**	**1.00**	**1.00**	0.00	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.96
F16	**1.00**	**1.00**	**1.00**	0.94	**1.00**	**1.00**	0.86	**1.00**	**1.00**	0.20	**1.00**	**1.00**
F17	0.73	0.00	0.00	0.43	0.03	0.76	**1.00**	**1.00**	0.82	0.10	0.00	0.00
F18	**1.00**	**1.00**	0.66	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F19	**1.00**	0.00	0.24	0.69	0.00	0.86	**1.00**	0.80	0.64	**1.00**	0.80	**1.00**
F20	**1.00**	0.03	0.70	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.96
F21	**1.00**	**1.00**	0.98	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F22	**1.00**	0.96	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F23	**1.00**	0.00	0.00	0.00	0.00	0.28	0.43	0.33	0.80	0.00	0.00	0.00
F24	**1.00**	0.03	0.14	0.99	0.43	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F25	**1.00**	0.07	0.02	0.67	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.90	1.00	**1.00**
F26	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F27	**1.00**	**1.00**	**1.00**	0.99	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F28	**1.00**	0.00	0.88	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.05	**1.00**	**1.00**
F29	**1.00**	0.00	0.66	0.97	0.76	0.98	0.00	**1.00**	**1.00**	**1.00**	**1.00**	0.80
F30	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Avg.	**0.95**	0.47	0.66	0.82	0.72	0.89	0.88	0.93	0.93	0.74	0.85	0.78

Table A4 Comparison between ERLDE and other ERLDE variants with respect to *RR*.

Problem	ERLDE	ERLDE-1	ERLDE-2	ERLDE-3	ERLDE-4	ERLDE-5	ERLDE-6
F01	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.63
F02	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F03	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F04	0.99	**1.00**	0.99	0.95	0.82	0.99	**1.00**
F05	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F06	**1.00**	**1.00**	**1.00**	**1.00**	0.97	**1.00**	**1.00**
F07	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F08	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F09	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F10	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F11	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F12	0.97	**0.99**	0.90	0.83	0.75	0.96	**0.99**
F13	0.64	0.59	**0.66**	0.63	0.52	0.64	0.57
F14	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F15	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F16	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F17	0.98	0.69	**1.00**	**1.00**	**1.00**	0.93	0.50
F18	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F19	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F20	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F21	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F22	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F23	**1.00**	**1.00**	1.00	0.98	0.94	**1.00**	**1.00**
F24	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F25	**1.00**	0.97	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F26	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F27	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F28	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F29	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F30	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Avg.	**0.99**	0.97	0.98	0.98	0.97	0.98	0.96

Table A5 Comparison between ERLDE and other ERLDE variants with respect to *SR*.

Problem	ERLDE	ERLDE-1	ERLDE-2	ERLDE-3	ERLDE-4	ERLDE-5	ERLDE-6
F01	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.33
F02	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F03	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F04	0.87	**1.00**	0.67	0.47	0.00	0.93	**1.00**
F05	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F06	**1.00**	**1.00**	**1.00**	**1.00**	0.73	**1.00**	**1.00**
F07	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F08	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F09	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F10	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F11	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F12	0.93	**0.93**	0.67	0.20	0.13	0.80	**0.93**

(to be continued)

Table A5 Comparison between ERLDE and other ERLDE variants with respect to *SR*. (continued)

Problem	ERLDE	ERLDE-1	ERLDE-2	ERLDE-3	ERLDE-4	ERLDE-5	ERLDE-6
F13	**0.07**	0.00	0.00	0.00	0.00	0.00	0.00
F14	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F15	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F16	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F17	0.73	0.13	**1.00**	**1.00**	**1.00**	0.60	0.00
F18	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F19	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F20	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F21	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F22	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F23	**1.00**	**1.00**	**1.00**	0.67	0.20	**1.00**	**1.00**
F24	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F25	**1.00**	0.93	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F26	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F27	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F28	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F29	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F30	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Avg.	**0.95**	0.93	0.94	0.91	0.87	0.94	0.91

Table A6 Comparison between ERLDE and ERLDE-7 with respect to *RR* and *SR*.

Problem	*RR*		*SR*	
	ERLDE	ERLDE-7	ERLDE	ERLDE-7
F01	1.00	1.00	1.00	1.00
F02	1.00	1.00	1.00	1.00
F03	1.00	1.00	1.00	1.00
F04	0.99	**1.00**	0.87	**1.00**
F05	1.00	1.00	1.00	1.00
F06	1.00	1.00	1.00	1.00
F07	1.00	1.00	1.00	1.00
F08	1.00	1.00	1.00	1.00
F09	1.00	1.00	1.00	1.00
F10	1.00	1.00	1.00	1.00
F11	1.00	1.00	1.00	1.00
F12	**0.97**	0.95	**0.93**	0.73
F13	**0.64**	0.57	**0.07**	0.00
F14	1.00	1.00	1.00	1.00
F15	1.00	1.00	1.00	1.00
F16	1.00	1.00	1.00	1.00
F17	**0.98**	0.88	**0.73**	0.27
F18	1.00	1.00	1.00	1.00
F19	1.00	1.00	1.00	1.00
F20	1.00	1.00	1.00	1.00
F21	1.00	1.00	1.00	1.00
F22	1.00	1.00	1.00	1.00
F23	1.00	1.00	1.00	1.00

(to be continued)

Table A6 Comparison between ERLDE and ERLDE-7 with respect to *RR* and *SR*. (continued)

Problem	*RR*		*SR*	
	ERLDE	ERLDE-7	ERLDE	ERLDE-7
F24	1.00	1.00	1.00	1.00
F25	1.00	1.00	1.00	1.00
F26	1.00	1.00	1.00	1.00
F27	1.00	1.00	1.00	1.00
F28	1.00	1.00	1.00	1.00
F29	1.00	1.00	1.00	1.00
F30	1.00	1.00	1.00	1.00
Avg.	**0.99**	0.98	**0.95**	0.93

Table A7 Comparison of different algorithms on the new test set in terms of *RR*.

Instance	ERLDE	LSTP	DR-JADE	MONES	A-WeB	FNODE	DDE/R	EMO-MMO
MNE1	**0.50**	**0.50**	0.06	0.25	0.25	0.47	0.00	0.25
MNE2	0.09	0.00	0.00	**0.25**	**0.25**	0.00	0.00	0.00
MNE3	0.50	**1.00**	0.50	0.00	0.11	0.49	0.52	0.00
MNE4	**0.50**	**0.50**	0.10	**0.50**	**0.50**	**0.50**	0.12	**0.50**
MNE5	**0.50**	**0.50**	0.00	**0.50**	**0.50**	**0.50**	0.00	0.00
MNE6	**1.00**	**1.00**	**1.00**	0.60	0.32	0.40	0.17	0.40
MNE7	0.76	**0.77**	0.46	0.47	0.09	0.67	0.43	0.11
MNE8	0.81	**0.90**	0.43	0.20	0.18	0.67	0.66	0.20
MNE9	**0.72**	0.53	0.40	0.20	0.10	0.60	0.42	0.40
MNE10	**0.76**	0.56	0.42	0.25	0.08	0.42	0.47	0.50
MNE11	0.13	**0.45**	0.01	0.17	0.17	0.11	0.03	0.17
MNE12	**0.35**	0.18	0.00	0.17	0.17	0.20	0.00	0.17
MNE13	0.22	**0.37**	0.00	0.00	0.13	0.34	0.00	0.00
MNE14	0.51	**0.61**	0.48	0.17	0.17	0.47	0.28	0.00
MNE15	**0.88**	0.28	0.57	0.31	0.06	0.17	0.49	0.00
MNE16	0.00	**0.46**	0.44	0.18	0.01	0.20	0.26	0.00
MNE17	0.00	0.29	**0.30**	0.04	0.00	0.26	0.24	0.00
MNE18	0.00	**0.90**	0.52	0.50	0.02	0.66	0.42	0.00
Avg.	0.46	**0.54**	0.32	0.26	0.17	0.40	0.25	0.15

Table A8 Comparison of different algorithms on the new test set in terms of *SR*.

Instance	ERLDE	LSTP	DR-JADE	MONES	A-WeB	FNODE	DDE/R	EMO-MMO
MNE1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MNE2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MNE3	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00
MNE4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MNE5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MNE6	**1.00**	**1.00**	**1.00**	0.00	0.04	0.00	0.00	0.00
MNE7	0.03	**0.12**	0.00	0.00	0.00	0.04	0.00	0.00
MNE8	0.03	**0.52**	0.00	0.00	0.00	0.00	0.00	0.00
MNE9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MNE10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MNE11	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MNE12	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

(to be continued)

Table A8 Comparison of different algorithms on the new test set in terms of *SR*. (continued)

Instance	ERLDE	LSTP	DR-JADE	MONES	A-WeB	FNODE	DDE/R	EMO-MMO
MNE13	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MNE14	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MNE15	**0.23**	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MNE16	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MNE17	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MNE18	0.00	**0.80**	0.04	0.00	0.00	0.52	0.16	0.00
Avg.	0.07	**0.19**	0.06	0.00	0.00	0.03	0.01	0.00

the new test set in terms of *RR* and *SR*, respectively.

Acknowledgment

This work was partly supported by the Natural Science Foundation of Guangxi Province (No. 2020JJA170038), Special Talent Project of Guangxi Science and Technology Base (No. GuiKe AD21220119), and the High-Level Talents Research Project of Beibu Gulf (No. 2020KYQD06)

References

[1] W. Gong, Z. Liao, X. Mi, L. Wang, and Y. Guo, Nonlinear equations solving with intelligent optimization algorithms: A survey, *Complex System Modeling and Simulation*, vol. 1, no. 1, pp. 15–32, 2021.

[2] F. Facchinei and C. Kanzow, Generalized Nash equilibrium problems, *4OR* , vol. 5, no. 3, pp. 173–210, 2007.

[3] H. D. Chiang and T. Wang, Novel homotopy theory for nonlinear networks and systems and its applications to electrical grids, *IEEE Transactions on Control of Network Systems*, vol. 5, no. 3, pp. 1051–1060, 2017.

[4] D. Guo, Z. Nie, and L. Yan, The application of noise-tolerant ZD design formula to robots' kinematic control via time-varying nonlinear equations solving, *IEEE Transactions on Systems*, *Man*, *and Cybernetics*: *Systems*, vol. 48, no. 12, pp. 2188–2197, 2017.

[5] Y. Luo, A new 3-cycle Newton chaos iteration solution method and its application to mechanism synthesis, in *Proc. International Conference on Mechanical Transmissions*, Chongqing, China, 2006, pp. 809–812.

[6] 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.

[7] H. Lu, S. Yang, M. Zhao, and S. Cheng, Multi-robot indoor environment map building based on multi-stage optimization method, *Complex System Modeling and Simulation*, vol. 1, no. 2, pp. 145–161, 2021.

[8] W. Zhang, W. Hou, C. Li, W. Yang, and M. Gen, Multidirection update-based multiobjective particle swarm optimization for mixed no-idle flow-shop scheduling problem, *Complex System Modeling and Simulation*, vol. 1, no. 3, pp. 176–197, 2021.

[9] W. Gao, G. Li, Q. Zhang, Y. Luo, and Z. Wang, Solving nonlinear equation systems by a two-phase evolutionary algorithm, *IEEE Transactions on Systems*, *Man*, *and Cybernetics*: *Systems*, vol. 51, no. 9, pp. 5652–5663, 2021.

[10] Z. Liao, W. Gong, X. Yan, L. Wang, and C. Hu, Solving nonlinear equations system with dynamic repulsion-based evolutionary algorithms, *IEEE Transactions on Systems*, *Man*, *and Cybernetics*: *Systems* , vol. 50, no. 4, pp. 1590–1601, 2020.

[11] W. He, W. Gong, L. Wang, X. Yan, and C. Hu, Fuzzy neighborhood-based differential evolution with orientation for nonlinear equation systems, *Knowledge-Based Systems*, vol. 182, p. 104796, 2019.

[12] Z. Liao, W. Gong, and L. Wang, A hybrid swarm intelligence with improved ring topology for nonlinear equations, *Scientia Sinica Informationis*, vol. 50, no. 3, pp. 396–407, 2020.

[13] J. Wu, W. Gong, and L. Wang, A clustering-based differential evolution with different crowding factors for nonlinear equations system, *Applied Soft Computing*, vol. 98, p. 106733, 2021.

[14] Z. Liao, W. Gong, L. Wang, X. Yan, and C. Hu, A decomposition-based differential evolution with reinitialization for nonlinear equations systems, *Knowledge-Based Systems*, vol. 191, p. 105312, 2020.

[15] Z. Liao, W. Gong, and L. Wang, Memetic niching-based evolutionary algorithms for solving nonlinear equation system, *Expert Systems with Applications*, vol. 149, p. 113261, 2020.

[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, *Nature* , vol. 518, no. 7540, pp. 529–533, 2015.

[17] S. Hussein, L. C. Peng, and S. J. Mohamad, A new reinforcement learning-based memetic particle swarm optimizer, *Applied Soft Computing*, vol. 43, pp. 276–297, 2016.

[18] L. Wang, Z. Pan, and J. Wang, A review of reinforcement learning based intelligent optimization for manufacturing scheduling, *Complex System Modeling and Simulation*, vol. 1, no. 4, pp. 257–270, 2021.

[19] Z. Hu and W. Gong, Constrained evolutionary optimization based on reinforcement learning using the objective function and constraints, *Knowledge-Based Systems*, vol. 237, p. 107731, 2021.

[20] Z. Hu, W. Gong, and S. Li, Reinforcement learning-based

differential evolution for parameters extraction of photovoltaic models, *Energy Reports*, vol. 7, pp. 916–928, 2021.

[21] E. Emary, H. Zawbaa, and C. Grosan, Experienced gray wolf optimization through reinforcement learning and neural networks, *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 3, pp. 681–694, 2018.

[22] Y. Liu, H. Lu, S. Cheng, and Y. Shi, An adaptive online parameter control algorithm for particle swarm optimization based on reinforcement learning, in *Proc. 2019 IEEE Congress on Evolutionary Computation (CEC)*, Wellington, New Zealand, 2019, pp. 815–822.

[23] Z. Li, L. Shi, C. Yue, Z. Shang, and B. Qu, Differential evolution based on reinforcement learning with fitness ranking for solving multimodal multiobjective problems, *Swarm and Evolutionary Computation*, vol. 49, pp. 234–244, 2019.

[24] M. M. Drugan, Reinforcement learning versus evolutionary computation: A survey on hybrid algorithms, *Swarm and Evolutionary Computation*, vol. 44, pp. 228–246, 2019.

[25] C. L. Karr, B. Weck, and L. M. Freeman, Solutions to systems of nonlinear equations via a genetic algorithm, *Engineering Applications of Artificial Intelligence*, vol. 11, no. 3, pp. 369–375, 1998.

[26] W. Song, Y. Wang, H. -X. Li, and Z. Cai, Locating multiple optimal solutions of nonlinear equation systems based on multiobjective optimization, *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 3, pp. 414–431, 2015.

[27] S. Lü, S. Han, W. Zhou, and J. Zhang, Recruitment-imitation mechanism for evolutionary reinforcement learning, *Information Sciences*, vol. 553, pp. 172–188, 2021.

[28] M. J. Hirsch, P. M. Pardalos, and M. G. C. Resende, Solving systems of nonlinear equations with continuous GRASP, *Nonlinear Analysis:Real World Applications*, vol. 10, no. 4, pp. 2000–2006, 2009.

[29] G. C. V. Ramadas and E. M. G. P. Fernandes, Combined mutation differential evolution to solve systems of nonlinear equations, *AIP Conference Proceedings*, vol. 1558, no. 1, pp. 582–585, 2013.

[30] W. Gong, Y. Wang, Z. Cai, and L. Wang, Finding multiple roots of nonlinear equation systems via a repulsion-based adaptive differential evolution, *IEEE Transactions on Systems*, *Man*, *and Cybernetics*: *Systems* , vol. 50, no. 4, pp. 1499–1513, 2020.

[31] L. Zhou and C. S. Jiang, New method for solving nonlinear equation systems, *Journal of Chinese Computer Systems*, vol. 29, no. 9, pp. 1709–1713, 2008.

[32] W. Gao and Y. Li, Solving a new test set of nonlinear equation systems by evolutionary algorithm, *IEEE Transactions on Cybernetics*, doi: 10.1109/TCYB.2021.3108563.

[33] C. Grosan and A. Abraham, A new approach for solving nonlinear equations systems, *IEEE Transactions on Systems*, *Man and Cybernetics—Part A*: *System and Humans*, vol. 38, no. 3, pp. 698–714, 2008.

[34] W. Gong, Y. Wang, Z. Cai, and S. Yang, A weighted biobjective transformation technique for locating multiple optimal solutions of nonlinear equation systems, *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 5, pp. 697–713, 2017.

[35] W. Gao, Y. Luo, J. Xu, and S. Zhu, Evolutionary algorithm with multiobjective optimization technique for solving nonlinear equation systems, *Information Sciences*, vol. 541, pp. 345–361, 2020.

[36] A. M. Ibrahim and M. A. Tawhid, A hybridization of differential evolution and monarch butterfly optimization for solving systems of nonlinear equations, *Journal of Computational Design and Engineering*, vol. 6, no. 3, pp. 354–367, 2019.

[37] B. Y. Qu, P. N. Suganthan, and J. J. Liang, Differential evolution with neighborhood mutation for multimodal optimization, *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 5, pp. 601–614, 2012.

[38] Z. Wang, Z. Zhan, Y. Lin, W. Yu, and J. Zhang, Automatic niching differential evolution with contour prediction approach for multimodal optimization problems, *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 1, pp. 114–128, 2020.

[39] H. Zhao, Z. Zhan, Y. Lin, X. Chen, X. Luo, J. Zhang, S. Kwong, and J. Zhang, Local binary pattern-based adaptive differential evolution for multimodal optimization problems, *IEEE Transactions on Cybernetics*, vol. 50, no. 7, pp. 3343–3357, 2020.

[40] R. Cheng, M. Li, K. Li, and X. Yao, Evolutionary multiobjective optimization-based multimodal optimization: Fitness landscape approximation and peak detection, *IEEE Transactions on Evolutionary*

Zuowen Liao received the BEng and MEng degrees from Wuhan Institute of Chemical Technology, in 2006 and 2010, respectively. He received the PhD degree from China University of Geosciences, Wuhan, China, in 2019. He is currently an associate professor with Beibu Gulf Ocean Development Research Center, Beibu Gulf University, Qinzhou, China. His current research interests include evolutionary computation, memetic computation, and computational intelligence.

Shuijia Li received the BEng degree in software engineering from Hubei University of Arts and Science, Xiangyang, China, in 2017. He is currently pursuing the PhD degree in geosciences information engineering with China University of Geosciences, Wuhan, China. His current research interests include evolutionary computation, transfer optimization, and multitasking evolution and its application.