Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

# An effective multiagent evolutionary algorithm integrating a novel roulette inversion operator for engineering optimization

Junling Zhang [a,*], Changyong Liang [a], Yongqing Huang [b], Jian Wu [c], Shanlin Yang [a]

[a] Institute of Computer Network Systems, Hefei University of Technology, Tunxi Road No. 193, Hefei 230009, China
[b] Department of Computer Science and Technology, Tongling University, Tongling 244000, China
[c] School of Business Administration, Zhejiang Normal University, Jinhua 321004, China

A R T I C L E   I N F O

*Keywords:*
Multiagent system
Inversion operator
Evolutionary algorithm
Numerical optimization
Linear system
Design optimization

A B S T R A C T

Multiagent systems have been studied and widely used in the field of artificial intelligence and computer science to catalyze computation intelligence. In this paper, a multiagent evolutionary algorithm called RAER based on the ERA multiagent modeling pattern is proposed, where ERA has the same architecture as Swarm including three parts of Environment, Reactive rules and Agents. RAER integrates a novel roulette inversion operator (RIO) proposed in this paper and theoretically proved to conquer the irrationality of the inversion operator (IO) designed by John Holland when used for real code stochastic optimization algorithms. Experiments for numerical optimization of 4 benchmark functions show that the RIO operator bears better functioning than IO operator. And experiments for numerical optimization of 12 benchmark functions are used to examine the performance and scalability of RAER along the problem dimensions ranging 20–10000, results indicate that RAER outperforms other comparative algorithms significantly. Also, two engineering optimization problems of a stable linear system approximation and a welded beam design are used to examine the applicability of RAER. Results show that RAER has better search ability and faster convergence speed. Especially for the approximation problem, REAR can find the proper optima belonging to different fixed search areas, which is significantly better than other algorithms and shows that RAER can search the problem domains more thoroughly than other algorithms. Hence, RAER is efficient and practical.

## 1. Introduction

Many practical engineering optimization problems can be formulated as numerical optimization problems and do not have explicit presentation or continuity of control variables, which are necessary for classical gradient-based optimization techniques. In order to overcome this difficulty, evolutionary algorithms (EAs) that only use objective functions and constrain values to search the optimal solutions have been proposed, including genetic algorithms (GAs), evolutionary strategies (ESs) and evolutionary programmings (EPs). Genetic Algorithms mimic the evolution process through selection, recombination and mutation [1]. Evolutionary programming and evolutionary strategy make use of mutation operators, but not recombination for the evolution process [2]. For numerical optimization problems, EAs typically maintain a population and adopt several operators in their algorithm structures to keep better abilities of exploration and exploitation trying to prevent premature convergence [2–6]. It has been recognized that EAs are a class of widely applicable and robust techniques for global optimization problems.

* Corresponding author.
*E-mail address:* zjllogic@126.com (J. Zhang).

Recently, Swarm Intelligence (SI) has emerged as an innovative distributed intelligent paradigm for solving optimization problems, which originally took its inspiration from the biological examples by swarming, flocking and herding phenomena in vertebrates [7]. Ref. [8] proposed Particle Swarm Optimization (PSO) as a direct search optimization method, which incorporates swarming behaviors observed in flocks of birds, schools of fish, or swarms of bees, and even human social behavior. Ref. [9] proposed Ant Colony Optimization (ACO) to solve discrete optimization problems, which is inspired from the foraging behavior of real ants. The famous Swarm [9,10] platform originally developed at the Santa Fe Institute is used to catalyze swarm intelligence through multiagent simulation of complex systems, which involves three key concepts: living environment, agents with reactive rules, and a schedule of agents' actions. Based on the Swarm's idea, Refs. [6,11] viewed a digital image as a 2-D cellular/lattice environment in which autonomous agents inhabit and exhibit several reactive behaviors to adaptively extract image features and segments. Inspired from the thoughts of Swarm, Ref. [12] presented a new multiagent modeling pattern called ERA (i.e., Environment, Reactive rules, and Agents) to solve the 7000-queen constraint satisfaction problem. Ref. [13] integrated the ERA pattern and GA to create a multiagent genetic algorithm (MAGA), they solved some high-dimensional benchmark functions and linear system approximation problems with satisfactory results. In this paper, the irrationality of the inversion operator (IO) proposed by John Holland in Ref. [1] when used for real code stochastic optimization is analyzed, and a novel roulette inversion operator (RIO) is designed and theoretically proved to conquer that irrationality. Then the roulette inversion operator and the ERA pattern are fused together to form a new multiagent evolutionary algorithm called RAER to solve global numerical optimization problems.

Comparatively thinking, AEA [6], MAGA [13] and IMCPA [14] also integrate IO in their algorithm structures and also integrate the irrationality of IO. In MAGA, the function of IO lies not only in utilizing the information of better individuals but also in adjusting the population diversity, so IO is given a rather big execution probability (0.8) and the Gaussian mutation operator is chosen. In IMCPA, IO functions with immune memory strategy only to utilize the information of better individuals, which is executed with the same small probability (0.3) as in AEA. And IMCPA solved high-dimensional benchmark functions with satisfactory results. In RAER, we prefer the new inversion operator (RIO) to mainly function as an operator only to utilize the information of better individuals, so RIO is given small execution probability same as in IMCPA and AEA, correspondingly special reactive rules suitable for RIO and reactive rule for maintaining population diversity are designed. The numerical experiments show that RIO bears better functioning than IO, and that RAER achieves better performance than other algorithms. The engineering experiments show that RAER can explore the problem domains more thoroughly than other algorithms.

The rest of this work is organized as follows: Section 2 discusses John Holland's inversion operator and roulette inversion operator for real code stochastic optimization algorithms. Section 3 describes the algorithm design of RAER and analyzes its convergence. Section 4 describes the experimental studies on the problems of unconstrained global numerical optimization. Section 5 describes experimental studies on approximation problem of a stable linear system and design optimization of a welded beam. Finally, conclusions are presented in Section 6.

## 2. Holland's inversion operator and roulette inversion operator

### 2.1. John Holland's inversion operator for real code stochastic optimization algorithms

John Holland's genetic algorithm theory includes not only selection operator, crossover operator and mutation operator but also inversion operator. John Holland's inversion operator is defined as follows:

**Definition 1** [1]. In genetic algorithms, $W = (x_1, x_2, \ldots, x_l)$ is a binary code individual of the current population. $l$ is chromosome length of $W$. $i_1$ and $i_2$ are integers generated randomly between 1 and $l$ inclusive, if $i_1 = i_2$ then do nothing about $W$, if $i_1 > i_2$ then swap $i_1$ and $i_2$ to make sure $i_1 < i_2$ and then do the following operation on $W$,

```
For $k = i_1$ to $i_1 + \lceil (i_2 - i_1 + 1)/2 \rceil - 1$ do
    Swap allele and index at locus $k$ with allele and index at locus $i_1 + i_2 - k$.
End.
```

When used in real code stochastic optimization algorithms, John Holland's inversion operator is mainly modified as follows:

**Definition 2** [6]. In real code stochastic optimization algorithms, $W = (x_1, x_2, \ldots, x_n)$ is an individual of the current population, $x_i \in [\underline{x}_i, \bar{x}_i]$, where $[\underline{x}_i, \bar{x}_i]$ is the domain of the $i$th variable $x_i$. $n$ is chromosome length of $W$. $i_1$ and $i_2$ are integers generated randomly between 1 and $n$ inclusive, if $i_1 = i_2$ then do nothing about $W$. if $i_1 > i_2$ then swap $i_1$ and $i_2$ to make sure $i_1 < i_2$ and then do the following operations on $W$,

*Step 1*: $W$ is firstly mapped on [0,1] according to

$$e_k = \frac{x_k - \underline{x}_k}{\bar{x}_k - \underline{x}_k}, \quad k = 1, \ldots, n.$$

Then, $W$ appears as $W' = (e_1, e_2, \ldots, e_n)$.

*Step 2*: $W'' = (e'_1, e'_2, \ldots, e'_n)$ is obtained by inverting $W'$ between locus at $i_1$ and $i_2$ according to

$$W'' = (e_1, e_2, \ldots, e_{i_1-1}, e_{i_2}, e_{i_2-1}, \ldots, e_{i_1+1}, e_{i_1}, e_{i_2+1}, \ldots, e_n).$$

*Step 3*: Finally, a new individual $W''' = (x'_1, x'_2, \ldots, x'_n)$ is obtained by remapping $W''$ back to $[\underline{x}_k, \bar{x}_k]$ according to

$$x'_k = \underline{x}_k + e'_k \times (\bar{x}_k - \underline{x}_k), \quad k = 1, \ldots, n.$$

Holland's inversion operator formulated as Definition 2 has been successfully applied in numerical optimization. AEA [6] applied inversion operator before recombination operator and solved numerical problems with dimension ranging 20–1000. IMCPA [14] combined inversion operator with artificial immune system and solved numerical problems with dimension ranging 20–10000 and achieved better results than AEA on high-dimensional problems. MAGA [13] applied inversion before an orthogonal crossover to solve 10 benchmark problems with dimension ranging 20–10000 and achieved better results than AEA and IMCPA on high-dimensional problems. In summary, Holland's inversion operator is examined as an effective operator with the following good properties: (1) Inversion operator can help host algorithms make good use of individuals' encoding information by sharing components' distribution information in corresponding domains. (2) When used with experiment design methods (such as orthogonal design) as crossover operator, inversion operator can improve the effect of sampling methods in a domain. (3) When in combination with crossover operator and mutation operator, inversion operator can help host algorithms make good use of useful information obtained by mutation operator to explore probabilistically in each dimension to find out improvable dimension.

About the Holland's inversion operator formulated as Definition 2, we give a theorem as follows:

**Theorem 1.** *Let $W = (x_1, x_2, \ldots, x_n)$ be an individual of the current population, n is chromosome length of W. X is a random integer between 1 and n inclusive independently generated with equal probability sampling. Let $P(X = i_1)$ $(i_1 = 1 \ldots n)$ and $P(Y = i_2)$ $(i_2 = 1 \ldots n)$ be the probability of the case of $X = i_1$ and $X = i_2$, respectively. Let $PI^H(Z = k)$ $(k = 1 \ldots n)$ be the probability for W's kth component to share distribution information with the uth $(u \neq k)$ component by Holland's inversion operator, then*

$$\begin{cases} PI^H(Z = k_1) = 2 \times \left[\frac{k_1}{n} - \left(\frac{k_1}{n}\right)^2\right] & \begin{array}{ll} k_1 = 1 \ldots n/2, & \textit{when n is even}, \\ k_1 = 1 \ldots (n+1)/2, & \textit{when n is odd}, \end{array} \\ PI^H(Z = n + 1 - k_1) = PI^H(Z = k_1). \end{cases}$$

**Proof.** Apparently, $P(X = i_1) = 1/n,\ i_1 = 1 \ldots n;\ P(Y = i_2) = 1/n,\ i_2 = 1 \ldots n;\ P(X = i_1, Y = i_2) = P(X = i_1) \times P(Y = i_2);\ P(i_1 \leqslant X \leqslant i_2) = (i_2 - i_1 + 1)/n$

$$\begin{aligned} PI^H(Z = k) &= P(X = k)P(Y \neq k) + P(X < k)P(Y \geqslant k) + P(X > k)P(Y \leqslant k) - P(X + Y = 2k, k = 2 \ldots n - 1) \\ &= \frac{1}{n} \times \frac{n-1}{n} + \frac{k-1}{n} \times \frac{n-k+1}{n} + \frac{k}{n} \times \frac{n-k}{n} - 2 \times \frac{\mathrm{Min}(k-1, n-k)}{n^2}. \end{aligned} \tag{a}$$

When $k_1 = 1 \ldots n/2$ for the case of $\mathrm{mod}(n,2) = 0$ or $k_1 = 1 \ldots (n+1)/2$ for the case of $\mathrm{mod}(n,2) = 1$, Eq. (a) is the same when $k = k_1$ and $k = n + 1 - k_1$. So only the case of $k = k_1$ is discussed here, therefore

$$\begin{aligned} PI^H(Z = k_1) &= \frac{1}{n} \times \frac{n-1}{n} + \frac{k_1 - 1}{n} \times \frac{n - k_1 + 1}{n} + \frac{k_1}{n} \times \frac{n - k_1}{n} - 2 \times \frac{(k_1 - 1)}{n^2} \\ &= \frac{1}{n^2} \times \{2 \times k_1 \times (n - k_1)\} \\ &= 2 \times \left\{\frac{k_1}{n} - \left(\frac{k_1}{n}\right)^2\right\}. \end{aligned}$$

And $PI^H(Z = n + 1 - k_1) = PI^H(Z = k_1)$. □

From Theorem 1 the following conclusions can be drawn: (1) When $t = n/2$ for the case of $\mathrm{mod}(n,2) = 0$, $PI^H(Z = t) = PI^H(Z = t + 1)$ is maximum. With $\mu = t \to 1$ or $\mu = t + 1 \to n$ $PI^H$ decreases monotonously. And $PI^H(Z = 1) = PI^H(Z = n)$ is minimum. (2) When $t = (n + 1)/2$ for the case of $\mathrm{mod}(n,2) = 1$, $PI^H(Z = t - 1) = PI^H(Z = t) = PI^H(Z = t + 1)$ is maximum. With $\mu = t - 1 \to 1$ or $\mu = t + 1 \to n$ $PI^H$ decreases monotonously. And $PI^H(Z = 1) = PI^H(Z = n)$ is minimum. So, when host algorithms use Holland's inversion operator formulated as Definition 2 to improve random search ability, each dimension of optimization problems will get explored with different probabilities, which is irrational apparently. Therefore, a novel inversion operator called roulette inversion operator is proposed to conquer this irrationality.

### 2.2. Roulette inversion operator

Basically for optimization problems, random search algorithms have no reason to explore some dimensions with larger probabilities than others, on the contrary, they should explore each dimension with same probability. In this section, we propose a novel roulette inversion operator described as Definition 3 to treat each dimension evenly.

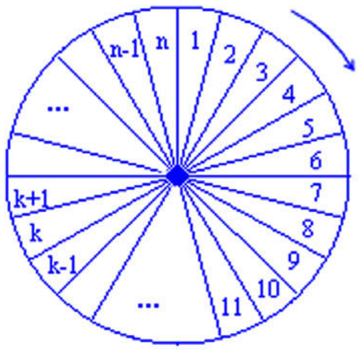

**Fig. 1.** Roulette inversion operator.

**Definition 3.** In real code stochastic optimization algorithms, $W = (x_1, x_2, \ldots, x_n)$ is an individual of the current population, $x_i \in [\underline{x}_i, \bar{x}_i]$, where $[\underline{x}_i, \bar{x}_i]$ is the domain of the $i$th variable $x_i$. $n$ is chromosome length of $W$. Put all components of $W$ clockwise on the roulette divided into $n$ equal parts shown as Fig. 1. Then rotate the roulette to generate two integers randomly as $a$ and $b$ in sequence, if $a = b$ then do nothing about $W$ else then do the following operations on $W$,

*Step 1*: $W$ is firstly mapped on $[0, 1]$ according to

$$e_k = \frac{x_k - \underline{x}_k}{\bar{x}_k - \underline{x}_k}, \quad k = 1, \ldots, n.$$

Then, $W$ appears as $W' = (e_1, e_2, \ldots, e_n)$.

*Step 2*: $W'' = (e'_1, e'_2, \ldots, e'_n)$ is obtained by inverting $W'$ according to the following rules:

(1) if $a < b$ then invert $W'$ by

$$W'' = (e_1, e_2, \ldots, e_{i_1-1}, e_{i_2}, e_{i_2-1}, \ldots, e_{i_1+1}, e_{i_1}, e_{i_2+1}, \ldots, e_n).$$

(2) if $a > b$ then invert $W'$ by

$$W'' = (e_{a+b-1}, e_{a+b-2}, \ldots, e_a, e_{b+1}, \ldots, e_{a-1}, e_b, e_{b-1}, \ldots, e_2, e_1, e_n, \ldots, e_{a+b}) \quad \text{or}$$
$$W'' = (e_{a+b-n-1}, \ldots, e_2, e_1, e_n, e_{a+1}, e_a, e_{b+1}, \ldots, e_{a-1}, \ldots, e_b, e_{b-1}, \ldots, e_{a+b-n}).$$

For examples, let $n = 5$, when $a = 4$ and $b = 1$ then we will get $W'' = (e_4, e_2, e_3, e_1, e_5)$, when $a = 5$ and $b = 3$ then we will get $W'' = (e_2, e_1, e_5, e_4, e_3)$.

*Step 3*: Finally, a new individual $W''' = (x'_1, x'_2, \ldots, x'_n)$ is obtained by remapping $W''$ back to $[\underline{x}_k, \bar{x}_k]$ according to

$$x'_k = \underline{x}_k + e'_k \times (\bar{x}_k - \underline{x}_k), \quad k = 1, \ldots, n.$$

About the roulette inversion operator formulated as Definition 3, we give a *theorem* as follows:

**Theorem 2.** *Let $W = (x_1, x_2, \ldots, x_n)$ be an individual of the current population, $n$ is chromosome length of $W$. $X$ is a random integer between 1 and $n$ inclusive independently generated with equal probability sampling. Let $P(X = a)$ $(a = 1 \ldots n)$ and $P(Y = b)$ $(b = 1 \ldots n)$ be the probability of the case of $X = a$ and $X = b$, respectively. Let $PI^Z(Z = k)$ $(k = 1 \ldots n)$ be the probability for $W$'s $k$th component to share distribution information with the $u$th $(u \neq k)$ component by roulette inversion operator, then*

$$PI^Z(Z = k) = \begin{cases} \frac{1}{2} \times \left[1 - \left(\frac{1}{n}\right)^2\right], & k = 1 \ldots n, \quad \textit{when } n \textit{ is odd;} \\ 1/2, \quad k = 1 \ldots n, & \quad \textit{when } n \textit{ is even.} \end{cases}$$

**Proof.** Apparently, $P(X = a) = 1/n$, $a = 1 \ldots n$, $P(Y = b) = 1/n$, $b = 1 \ldots n$. $P(X = a, Y = b) = P(X = a) \times P(Y = b)$. $P(a \leqslant X \leqslant b) = (b - a + 1)/n$

$$\begin{aligned} PI^Z(Z = k) &= P(X < k)(P(Y \geqslant k) \cup P(Y < X)) + P(X = k)P(Y \neq k) + P(X > k)P(k \leqslant Y < X) \\ &\quad - P(X + Y = 2k, k = 2 \ldots n - 1) \cup P(X + Y = k, k = n) \cup P(X + Y = n + 2k, k = 1) \\ &= \sum_{i=1}^{k-1} \left( P(X = i) - \sum_{m=i}^{k-1} P(X = i)P(Y = m) \right) + P(X = k)P(Y \neq k) \\ &\quad + \sum_{j=k+1}^{n} \left( P(X = j) - \sum_{m=j}^{n} P(X = j)P(Y = m) - \sum_{\eta=1}^{k-1} P(X = j)P(Y = \eta) \right) - \frac{f(n)}{n^2} \\ &= \sum_{i=1}^{k-1} \left( \frac{1}{n} - \frac{1}{n} \times \frac{k - i}{n} \right) + \frac{1}{n} \times \frac{n - 1}{n} + \sum_{j=k+1}^{n} \left( \frac{1}{n} - \frac{1}{n} \times \frac{n - j + 1}{n} - \frac{1}{n} \times \frac{k - 1}{n} \right) - \frac{f(n)}{n^2}. \end{aligned} \quad \text{(b)}$$

$f(n)$ in $(b)$ is a function of $n$, $f(n) = n/2 - 1$ for the case of $\mathrm{mod}(n,2) = 0$ and $f(n) = (n - 1)/2$ for the case of $\mathrm{mod}(n,2) = 1$. Then Eq. (b) appears as

$$PI^Z(Z = k) = \begin{cases} \frac{1}{2} \times \left[1 - \left(\frac{1}{n}\right)^2\right], & k = 1 \ldots n, \quad \text{when } n \text{ is odd;} \\ 1/2, \quad k = 1 \ldots n, & \text{when } n \text{ is even.} \end{cases} \qquad \square$$

From Theorems 1 and 2 the following conclusions can be drawn: (1) When $t = n/2$ and $k = 1 \ldots n$ for the case of $\mathrm{mod}(n,2) = 0$, $PI^H(Z = t) = PI^H(Z = t + 1) = PI^Z(Z = k)$. (2) When $t = (n + 1)/2$ and $k = 1 \ldots n$ for the case of $\mathrm{mod}(n,2) = 1$, $PI^H(Z = t - 1) = PI^H(Z = t) = PI^H(Z = t + 1) = PI^Z(Z = k)$.

So, when host algorithms use roulette inversion operator to improve random search ability, each dimension of optimization problems will get explored with same probability. In fact, roulette inversion operator makes each component of an individual to get corresponding compensation probability of sharing distribution information so that each component is treated as the middle-locus components for the case of using Holland's inversion operator.

## 3. Algorithm design of RAER and its convergence analysis

ERA pattern is a simple but effective way to simulate complex adaptive systems, its virtues of intrinsical parallelism, globality and dynamic nature make itself also an effective way to catalyze computation intelligence [6,11–13]. According to Refs. [6,11–13], the ERA multiagent system modeling pattern enlightened from Swarm can be drawn as follows:

(1) *Environment*: Environment can be defined according to different domains where agents live and interact with each other locally. Useful information can be dispersed and shared everywhere. And there is certain selection pressure belonging to a specified kind of environment.
(2) *Agents*: Agents are units endowed with artificial intelligence. Each agent is able to sense and learn from its local environment.
(3) Interactive rules: Interactive rules are the behavior rules designed for agents so that all agents can interactive with each other to achieve best performance.

In this section, EAR pattern and the roulette inversion operator (RIO) are fused together to form a new multiagent evolutionary algorithm (RAER). Especially, uniform design method is introduced to design special interactive rules suitable for RIO in order to strengthen the search ability of RAER.

### 3.1. Algorithm design of RAER

In what follows, RAER is deigned through the three components of ERA pattern.

(1) Agents' Living Environment:

As validated in Ref. [6,11,13], a 2-D cellular/lattice environment is a suitable and comprehensible abstract environment for multiagent system modeling. So in RAER, we simply introduce a kind of local environment defined in Ref. [13] to simulate agents' living environment.

**Definition 4** [13]. All agents live in a lattice-like connected environment. $L$ is called an agent lattice. The size of $L_{M\times N}$ is $M \times N$, where $M$ and $N$ are integers. Each agent is fixed on a lattice-point and it can only interact with its neighbors. Suppose that the agent located at $(i,j)$ is represented as $L_{i,j}$, $i = 1,2,\ldots,M$, $j = 1,2,\ldots,N$, then the neighbors of $L_{i,j}$ are defined as $Nbs_{ij} = \{L_{i_1 j}, L_{ij_1}, L_{i_2 j}, L_{ij_2}\}$, where

$$i_1 = \begin{cases} i-1 & i \neq 1, \\ M & i = 1, \end{cases} \quad j_1 = \begin{cases} j-1 & j \neq 1, \\ N & j = 1, \end{cases} \quad i_2 = \begin{cases} i+1 & i \neq M, \\ 1 & i = M, \end{cases} \quad j_2 = \begin{cases} j+1 & j \neq N, \\ 1 & j = N. \end{cases}$$

So, $\{Nbs_{ij}, L_{ij}\}$ forms the local competitive environment of $L_{i,j}$.

(2) Agents:

Agents in RAER can exhibit different phenotypes under certain selection pressure (such as Maximize or Minimize) for concrete optimization problems, that is to say each agent is also endowed with a value of adaptability measurement. In RAER agents are randomly initialized in problem domain, that is, let $W_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ is a real code agent, $W_i \in [\underline{X}, \overline{X}]$, then $W_i$ is generated according to

$$x_{i,j} = \underline{x}_j + \mathrm{rand} \times (\bar{x}_j - x_j), \quad j = 1, \ldots, n. \tag{1}$$

(3) Competitive Interactive Rules for Agents:

In RAER, four competitive interactive behavior rules are designed to conduct the agents in $L_{M\times N}$ as follows: competitive exclusion behavior, local dispersal & rebirth behavior, local re-expansion behavior and propagation behavior.

(R1) Competitive exclusion behavior:

Agents must exhibit exclusion behavior through competition under a certain selection pressure. Agents in Ref. [12] exclude other agents by the actions of least-move, better-move and random-move. In Ref. [13], agents with poorer adaptability will be excluded and their resources will be inhabited by those with better adaptability. So in RAER, agents in $\{Nbs_{ij}, L_{ij}\}$ with poorer adaptability will be competitively excluded by the agents in $\{Nbs_{ij}, L_{ij}\}$ with better adaptability.

(R2) Local dispersal and rebirth behavior:

Here, we choose uniform experimental design method in corporation with RIO operator to design local dispersal and rebirth interactive behavior. Uniform design as an important experimental design technique was proposed by Fang and Wang [15] in 1981. Its main objective is to sample a small set of points from a given set of points such that the sampled points are uniformly scattered. Suppose there are $m$ factors and $q$ levels per factor. When $m$ and $q$ are given, the uniform design method selects $q$ combinations out of $q^m$ possible combinations, such that these $q$ combinations are scattered uniformly over the space of all possible combinations. The selected $q$ combinations are expressed in terms of a uniform array $U(m,q) = (U_{i,j})_{q\times m}$, where $U_{i,j}$ is the level of the $j$th factor in the $i$th combination. In practice, two methods are usually used to generate uniform tables according to Eq. (2). We refer the readers to [15–17] for more details about $U^1$ and $U^2$. $U^1$ and $U^2$ can be used in combination to get better performance according to concrete cases

$$U^1_{i,j} = (i\sigma^{j-1} \bmod q) + 1, \quad U^2_{i,j} = (ih_j - 1)[\bmod m] + 1. \tag{2}$$

For example, uniform tables with four factors and five levels can be generated by $U^1$ and $U^2$ as follows:

$$U^1(5,4) = \begin{bmatrix} 2 & 3 & 5 & 4 \\ 3 & 5 & 4 & 2 \\ 4 & 2 & 3 & 5 \\ 5 & 4 & 2 & 3 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \quad U^2(5,4) = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 1 & 3 \\ 3 & 1 & 4 & 2 \\ 4 & 3 & 2 & 1 \\ 5 & 5 & 5 & 5 \end{bmatrix}.$$

In the first combination of $U^1$, the four factors have respective levels 2, 3, 5, 4; in the first combination of $U^2$, the four factors have respective levels 1, 2, 3, 4, etc.

Suppose that the problem domain is $[\underline{X}, \overline{X}]$ where $\underline{X} = (\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_n)$ and $\overline{X} = (\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)$, then the domain of $i$th dimension is quantized into $[\beta_{i,1}, \beta_{i,1}, \ldots, \beta_{i,q}]$, where

$$\beta_{i,j} = \begin{cases} \underline{x}_i, & j = 1, \\ \underline{x}_i + (j-1) \times \left(\frac{\bar{x}_i - \underline{x}_i}{q-1}\right), & 2 \leqslant j \leqslant (q-1), \\ \bar{x}_i, & j = q. \end{cases} \tag{3}$$

We divides the $n$ variables of an individual into $m$ group according to $U(m,q)$ according to

$$\begin{cases} f_1 = (x_1, x_{1\cdot m+1}, \ldots, x_{(g-1)\cdot m+1}, x_{g\cdot m+1}), \\ f_2 = (x_2, x_{1\cdot m+2}, \ldots, x_{(g-1)\cdot m+2}, x_{g\cdot m+2}), \\ \cdots \\ f_k = (x_k, x_{1\cdot m+k}, \ldots, x_{(g-1)\cdot m+k}, x_{g\cdot m+k}), \qquad g = (n-k)/m, \quad k = \mathrm{mod}(n,m), \\ f_{k+1} = (x_{k+1}, x_{1\cdot m+k+1}, \ldots, x_{(g-1)\cdot m+k+1}), \\ \cdots \\ f_m = (x_m, x_{1\cdot m+m}, \ldots, x_{(g-1)\cdot m+m}), \end{cases} \tag{4}$$

so that the uniform design method can be applied to high-dimensional problems.

*Local dispersal*: let a certain $\{Nbs_{ij}, L_{ij}\}$ be composed of $W_1$–$W_5$, and $W_1$ is the agent with best adaptability. So $W_2$–$W_5$ will be competitively excluded and their positions will be filled with four agents generated based on $W_1$. In RAER, local dispersal action consists of near distance dispersal and farther distance dispersal with execution probabilities of $P_{nd}$ and $P_{fd}(P_{fd} = 1 - P_{nd})$, respectively. Let $P_d$ be a uniformly distributed random number between [0,1]. If $P_d \leqslant P_{nd}$ then new agents are generated according to

$$w_i = \max(\min(w_1 + \mathrm{sign}(\mathrm{rand} - 0.5) \times \mathrm{rand} \times (w_1 - w_i), \overline{X}), \underline{X}) \tag{5}$$

or

$$w_i = \max(\min(w_1 + \mathrm{random}(-2qr_n, 2qr_n), \overline{X}), \underline{X}), \tag{6}$$

where sign(·) stands for the sign function, rand is a generator for uniformly distributed random numbers between [0,1], $q$ is the levels of selected uniform table $(U_{i,j})_{q\times m}$, $r_n$ is niche radius that usually equals 0.1, random$(-2qr_n, 2qr_n)$ is a generator for uniformly distributed random numbers between $[-2qr_n, 2qr_n]$, then the local dispersal domain $[l_p, u_p]$ for sampling decided by $W_1$ and the new $W_i$ can be gained according to

$$l_p = [\min(x_{1,1}, x_{i,1}), \ldots, \min(x_{1,n}, x_{i,n})], \quad u_p = [\max(x_{1,1}, x_{i,1}), \ldots, \max(x_{1,n}, x_{i,n})]. \tag{7}$$

If $P_d > P_{nd}$ then new agents are generated according to

$$W_i = RIO(W_1), \tag{8}$$

where *RIO* stands for the above-discussed roulette inversion operator, then the local dispersal domain $[l_p, u_p]$ for sampling decided by $W_1$ and the new $W_i$ can be gained according to following method.

Suppose $W_1 = (x_1, x_2, \ldots, x_{i-1}, x_i, \ldots, x_j, x_{j+1}, \ldots, x_n)$ is a real code agent, when handled with roulette inversion operator, substring $(x_i, \ldots, x_j)$ is chosen to be inverted and $x'_k (k = 1 \ldots K,\ K = j - i + 1)$ will exchange distribution information with $x'_{K+1-k}$, $x'_k \in [\underline{x}'_k, \bar{x}'_k]$, $x'_{K+1-k} \in [\underline{x}'_{K+1-k}, \bar{x}'_{K+1-k}]$, let $d_k = (x'_k - \underline{x}'_k)/(\bar{x}'_k - \underline{x}'_k)$, $d_{K+1-k} = (x'_{K+1-k} - \underline{x}'_{K+1-k})/(\bar{x}'_{K+1-k} - \underline{x}'_{K+1-k})$, $d = |d_k - d_{K+1-k}|$, then the new search domain of $x'_k$, i.e. $[\underline{x}''_k, \bar{x}''_k]$, is decided according to

$$\underline{x}''_k = \max((d_k - d) \times (\bar{x}'_k - \underline{x}'_k) + \underline{x}'_k, \underline{x}'_k), \quad \bar{x}''_k = \min((d_k + d) \times (\bar{x}'_k - \underline{x}'_k) + \underline{x}'_k, \bar{x}'_k) \tag{9}$$

and the $x'_{K+1-k}$th dimension's search domain $[\underline{x}''_{K+1-k}, \bar{x}''_{K+1-k}]$ is decided according to

$$\begin{aligned} \underline{x}''_{K+1-k} &= \max((d_{K+1-k} - d) \times (\bar{x}'_{K+1-k} - \underline{x}'_{K+1-k}) + \underline{x}'_{K+1-k}, \underline{x}'_{K+1-k}), \\ \bar{x}''_{K+1-k} &= \min((d_{K+1-k} + d) \times (\bar{x}'_{K+1-k} - \underline{x}'_{K+1-k}) + \underline{x}'_{K+1-k}, \bar{x}'_{K+1-k}). \end{aligned} \tag{10}$$

*Rebirth*: Apply the uniform experimental method with probability $P_s$ to sample effective points in local search domain decided by Eq. (7) or Eqs. (9) and (10) according to selected uniform table $(U_{i,j})_{q\times m}$, then the best two of $q$ points are selected to replace $W_1$ and $W_i$.

(R3) Local re-expansion behavior:

Agents with Better adaptability in RAER are allowed to exhibit better phenotypes through re-expansion behavior in their local environments. We adopt miniature agent lattice $\mathrm{RL}_{m\times n}$ ($m$, $n$ are small integers) with same structure as $L_{M\times N}$ to perform re-expansion behavior on deserved agents for the purpose of local refinement. Here we design an adaptive generation strategy for $\mathrm{RL}_{m\times n}$ as follows:

Suppose $W_i = (x_{i1}, x_{i2}, \ldots, x_{iu}), x_{ij} \in [\underline{x}_j, \bar{x}_j], j = 1 \ldots u$, is the agent to re-expand, then the agents $RW_t = (x_{t1}, x_{t2}, \ldots, x_{tu})$ in $\mathrm{RL}_{m\times n}$ except $W_i$ are generated according to

$$\begin{aligned} &x_{tj} = \mathrm{Min}\{\max\{x_{ij} \times random(1 - sR, 1 + sR), \underline{x}_j\}, \bar{x}_j\}, \quad j = 1 \ldots u, \quad t = 1 \ldots m \times n - 1, \\ &\quad \text{if } (x_{tj} - x_{ij}) > r_n \text{ then } x_{tj} = \min(\max((x_{ij} + r_n), \underline{x}_j), \bar{x}_j), \\ &\quad \text{if } (x_{tj} - x_{ij}) < -r_n \text{ then } x_{tj} = \min(\max((x_{ij} - r_n), \underline{x}_j), \bar{x}_j), \end{aligned} \tag{11}$$

where $sR$ is re-expansion multiplier, random$(1 - sR, 1 + sR)$ is a generator for uniformly distributed random numbers between $[1 - sR, 1 + sR]$, $r_n$ is niche radius.

Since the purpose of re-expansion behavior is to locally refine the better solutions, so a mutation strategy of a small mutation rate should be adopted here, we choose the following Eq. (12) strategy in $\mathrm{RL}_{m\times n}$ to replace Eq. (6) in host agent lattice $L_{M\times N}$:

$$x_{ik} = \begin{cases} \underline{x}_k + \mathrm{rand}^1 \times (\bar{x}_k - \underline{x}_k), & \mathrm{rand} < 1/n, \\ x_{ik}, & \text{otherwise} \end{cases} \quad k = 1 \ldots n, \tag{12}$$

where $\mathrm{rand}^1$, rand are independent generators for uniformly distributed random numbers between [0,1].

(R4) Propagation behavior:

In addition to the intrinsical diffusion [13] ability of the agent lattice, an extra propagation behavior is designed as follows in RAER to ensure the globality requirement of ERA pattern:

*Step* 1: Let $L_{current}$ stands for the current agent lattice, let $L'_{current} = \mathrm{Operations}(L_{current})$ stands for the agent lattice just before getting into next iteration as initial lattice $L_{next}$ after several operations.
*Step* 2: Sort all agents in $L'_{current}$ in descending order by adaptability, i.e., let $L''_{current} = Sort\_Des(L'_{current})$.
*Step* 3: Generate RP $\times$ $M \times N$ randomly according to Eq. (1) to replace the corresponding agents in $L''_{current}$ with poorer *fitness*, where RP is the diversity control multiplier, then we get $L'''_{current}$.
*Step* 4: Randomly redistribute agents in $L'''_{current}$ to get $L_{next}$ so that the new agents could get everywhere in agent lattice carrying useful information.

### 3.2. Implementation of RAER

Overall RAER is composed of a larger host agent lattice $L_{M\times N}$ functioning as a global explorer and a miniature agent lattice $\mathrm{RL}_{m\times n}$ functioning as a local search exploiter. In $L_{M\times N}$, population diversity can be maintained through three ways: $L_{M\times N}$'s intrinsical diffusion [13], propagation behavior and roulette inversion operation. While in $\mathrm{RL}_{m\times n}$, the main purpose is to exploit the resources in the local scopes of better agents in $L_{M\times N}$. In order to reduce the computation cost and be comparable, $\mathrm{RL}_{m\times n}$ is generated by Eq. (11) according to the agent with best fitness in $L_{M\times N}$. The details of RAER are described by the following Algorithms 1 and 2:

**Algorithm 1** (*RAER Algorithm*). Initialize $L_{M\times N}$ according to Eq. (1), set $P_{nd}$, $P^5_{nd}$, $P^6_{nd} = 1 - P^5_{nd}$, $P_{fd} = 1 - P_{nd}$, $P_s$, choose $U_m(q^n)$, set generation counter $UGen = 1$, set termination signal $done$ = false.

```
while $\sim$ done
  $MAX = [\,]$;
  for $L_{ij}$ in $L_{M\times N}$
    if $L_{ij} == MAX_{ij}$ // $MAX_{ij}$ is the agent with best adaptability in $\{Nbs_{ij}, L_{ij}\}$.
      Do nothing.
    else
      if $U(0,1) \leqslant P_{nd}$ //$U(0,1)$ generates random numbers between [0,1].
        if $U(0,1) \leqslant P^5_{nd}$ Execute Eq. (5) on $L_{ij}$. else Execute Eq. (6) on $L_{ij}$. end
      else Execute Eq. (8) on $L_{ij}$; $MAX = [MAX; L_{ij}MAX_{ij}]$.
      end
    end
  end
  for $L_{ij}$ in $L_{M\times N}$
    if $L_{ij} == MAX_{ij}$ Do nothing.
    else
      if $U(0,1) \leqslant P_s$
        if $L_{ij}$ is in $MAX(:,1)$
          Execute the uniform design sampling method on the space determined by $[L_{ij}MAX_{ij}]$ in $MAX$ according to Eqs.
          (9) and (10) to get two better points to replace $L_{ij}$ and $MAX_{ij}$.
    else
          Execute the uniform design sampling method on the space determined by $[L_{ij}MAX_{ij}]$ according to Eq. (7) to get
          two better points to replace $L_{ij}$ and $MAX_{ij}$.
        end
      end
    end
  end
          Execute the local re-expansion behavior according to Algorithm 2 on the agent $MAX(UGen)$ with best adaptabil-
ity in $L_{M\times N}$.
Execute propagation operator on $L_{M\times N}$.
Examine the termination signal $done$.
$UGen = UGen + 1$;
end.
```

**Algorithm 2** (*Local re-expansion behavior*). Initialize $RL_{M\times N}$ according to Eq. (11), set $\mathrm{RP}_{nd}$, $\mathrm{RP}^5_{nd}$, $\mathrm{RP}^{12}_{nd} = 1 - \mathrm{RP}^5_{nd}$, $\mathrm{RP}_{fd} = 1 - \mathrm{RP}_{nd}$, $\mathrm{RP}_s$, choose $U_m(q^n)$, set maximum generation $MGen$, set generation counter$UGen = 1$, set termination signal $done$ = false.

```
while $\sim$ done
  $MAX = [\,]$;
  for $L_{ij}$ in $L_{M\times N}$
    if $L_{ij} == MAX_{ij}$ // $MAX_{ij}$ is the agent with best adaptability in $\{Nbs_{ij}, L_{ij}\}$.
      Do nothing.
    else
      if $U(0,1) \leqslant RP_{nd}$ //$U(0,1)$ generates random numbers between [0,1].
        if $U(0,1) \leqslant RP^5_{nd}$ Execute Eq. (5) on $L_{ij}$. else Execute Eq. (12) on $L_{ij}$. end
      else Execute Eq. (8) on $L_{ij}$;$MAX = [MAX; L_{ij}MAX_{ij}]$.
      end
    end
  end
  for $L_{ij}$ in $L_{M\times N}$
    if $L_{ij} == MAX_{ij}$ Do nothing.
    else
      if $U(0,1) \leqslant RP_s$
        if $L_{ij}$ is in $MAX(:,1)$
          Execute the uniform design sampling method on the space determined by $[L_{ij}MAX_{ij}]$ in $MAX$ according to Eqs.
          (9) and (10) to get two better points to replace $L_{ij}$ and $MAX_{ij}$.
        else
            Execute the uniform design sampling method on the space determined by $[L_{ij}MAX_{ij}]$ according to Eq. (7) to get
            two better points to replace $L_{ij}$ and $MAX_{ij}$.
          end
        end
      end
    end
    if UGen > $MGen$ $done$=$true$; else$UGen$ = $UGen$ + 1; end
  end.
```

## 3.3. Convergence analysis of RAER

Here the convergence of RAER is analyzed by real coding directly. Suppose that the search domain is $S = [\underline{X}, \overline{X}]$, and the required precision is $\varepsilon$. Thus the search space $S$ can be changed to a discrete space $S'$. The number of elements in $S'$, $|S'|$, is equal to $\prod_{i=1}^{n}\lceil(\underline{x}_i - \bar{x}_i)/\varepsilon\rceil$, and each element $w \in S'$ is a candidate solution, namely an agent. Then $S'$ can be divided into subsets $ST_j = \{Fitness(w) = E_j | w \in S'\}$, $E_j \in E = \{E_1, E_2, \ldots, E_{-E-}\}$ according to certain adaptability of $w$ called $Fitness(w)$, where $E = \{E_1, E_2, \ldots, E_{|E|}\}$ is the adaptability grade set of elements in $S'$ and $E_1 > E_2 > \ldots > E_{|E|}$. Obviously $E_1$ is the grade of agents of best adaptability and $ST_1$ consists of all the agents of best adaptability. So in the RAER the $L_{M\times N}$ belongs to a certain $ST_j$ intrinsically. Let $M'_t$ stands for the set of all agents in $L_{M\times N}$, where $t = \min\{j | M' \cap ST_j \neq \emptyset\}$, obviously $|M'| = M \times N$. Let $P_{t,k}\left(M'_t \overset{operations}{\rightarrow} M'_k, k > t\right)$ stands for the probability of that $M'$ of higher adaptability grade is changed to $M'$ of lower adaptability grade after the operations in RAER, and $P_{t,k}\left(M'_t \overset{operations}{\rightarrow} M'_k, k \leqslant t\right)$ stands for the probability of that $M'$ of lower adaptability grade is changed to $M'$ of higher adaptability grade after the operations in RAER. The convergence of RAER is proved as follows.

**Definition 5.** RAER converges to global optimum if and only if $\lim_{k\to\infty} P(M'(k) = M'_1) = 1$, where $k$ is the counter of weed growth period, $P(\cdot)$ stands for the probability.

**Theorem 3** [18]. *Let $P'$:$n' \times n'$ be a reducible stochastic matrix which means that by applying the same permutations to rows and columns, $P'$ can be brought into the form $\begin{bmatrix} C & 0 \\ R & T \end{bmatrix}$, where $C$:$m' \times m'$ is a primitive stochastic matrix and $R$, $T \neq \mathbf{0}$. Then*

$$P'^{\infty} = \lim_{k\to\infty} P'^{k} = \lim_{k\to\infty} \begin{bmatrix} C^k & 0 \\ \sum\limits_{i=0}^{k-1} T^i R C^{k-i} & T^k \end{bmatrix} = \begin{bmatrix} C^{\infty} & 0 \\ R^{\infty} & 0 \end{bmatrix}$$

*is a stable stochastic matrix with $P'^{\infty} = 1'p^{\infty}$, where $p^{\infty} = p^0 P'^{\infty}$ is unique regardless of the initial distribution, and $p^{\infty}$ satisfies: $p_i^{\infty} > 0$ for $1 \leqslant i \leqslant m'$ and $p_i^{\infty} = 0$ for $m' \leqslant i \leqslant n'$.*

**Theorem 4.** *In RAER, $\forall i, t \in \{1, 2, \ldots, |E|\}$, $P_{t,i}\left(M'_t \overset{operations}{\rightarrow} M'_i, i > t\right) = 0$, and $P_{t,i}\left(M_t \overset{operations}{\rightarrow} M_i, i \leqslant t\right) > 0$.*

**Proof.** Firstly, in RAER elitist strategy is adopted, that is, the agent of best adaptability in every $\{Nbs_{ij}, L_{ij}\}$ is allowed to stay put so that in at least one agent of globally best adaptability can stay in $L_{M\times N}$ $(UGen)$ and go to next weed growth period $L_{M\times N}(UGen + 1)$. So $P_{t,i}\left(M'_t \overset{operations}{\rightarrow} M'_i,\ i > t\right) = 0$.

Secondly, let $M'_t$ stand for the set of all agents in $L_{M\times N}$ in the $k$th weed growth period, so there must exist an agent $x^* = (x_1, x_2, \ldots, x_n) \in ST_t$, suppose $x^*$ is selected into $\mathrm{RL}_{m\times n}$ to be re-expanded. And $\exists x' \in ST_i, i \leqslant t$, suppose that there are $m_1$ variables, $(x'_1, x'_2, \ldots, x'_{m_1})$, in $x^*$ which are different from the corresponding $x^*$. Then the probability to generate $x^*$ to $x'$ by Eq. (12) in Algorithm 2 is

$$PT = (1 - 1/n)^{(n-m_1)} \times \prod_{i=1}^{m_1} \varepsilon/(n \times (\bar{x}'_i - x'_i)) > 0.$$

Thus,

$$P_{t,i}\left(M_t \overset{\text{operations}}{\rightarrow} M_i, i \leqslant t\right) > \mathrm{RP}_{nd} \times (1 - \mathrm{RP}_{nd}^5) \times PT > 0.$$

It follows from Theorem 4 that there is always a positive probability to transit from an agent lattice to the one with identical or higher energy and a zero probability to the one with lower energy. Thus, once RAER goes to $M'_1$, it never goes out. □

**Theorem 5.** *RAER converges to global optimum.*

**Proof.** It is clear that $M'_t,\ t = 1, 2, \ldots, |E|$, as a state in a homogeneous finite Markov chain. According to Theorem 4, the transition matrix of the Markov chain can be written as follows:

$$P' = \begin{bmatrix} P_{1,1} & 0 & \cdots & 0 \\ P_{2,1} & P_{2,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ P_{|E|,1} & P_{|E|,2} & \cdots & P_{|E|,|E|} \end{bmatrix} = \begin{bmatrix} C & 0 \\ R & T \end{bmatrix},$$

where $R = (P_{2,1}, P_{3,1}, \ldots, P_{|E|,1})^T > 0$, $T \neq 0$, $C = (P_{1,1}) = (1) \neq 0$. □

According to Theorem 3, we can get

$$P'^{\infty} = \lim_{k\to\infty} P'^{k} = \lim_{k\to\infty} \begin{bmatrix} C^k & 0 \\ \sum_{i=0}^{k-1} T^i R C^{k-i} & T^k \end{bmatrix} = \begin{bmatrix} C^{\infty} & 0 \\ R^{\infty} & 0 \end{bmatrix},$$

where $C^{\infty} = 1$, $R^{\infty} = (1, 1, \ldots, 1)^T$, so $P'^{\infty}$ is a stable stochastic matrix, and

$$P'^{\infty} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & \cdots & 0 \end{bmatrix}.$$

Therefore,

$$\lim_{k\to\infty} P(M'(k) = M'_1) = 1.$$

And according to Definition 5 we can get that RAER converges to global optimum.

## 4. Experimental studies on unconstrained global numerical optimization

### 4.1. Discussion on parameter settings of RAER

In general, finding the most appropriate combination of parameters occurring in an evolutionary algorithm is considered to be the most important and perhaps most difficult task [19]. In order to suggest the parameters of RAER for practical application (such as engineering problems that usually involve difficult optimization problems), the following difficult benchmark function $f_{01}$ is utilized to discuss the parameter settings of REAR. $f_{01}$ is often used to test the comprehensive performance of optimization algorithms, but seldom tested in references with their dimensions larger than 30 [4].

$f_{01}$: Ellipsoidal Function:

$$\mathrm{Min} f(x_i) = \sum_{i=1}^{N} (x_i - i)^2, x_i \in [-100, 100]^N;$$

Tables 1–6 show the statistical optimization results for $f_{01}$ according to various parameter settings. Based on these results, how to set the parameters to achieve better performance of REAR is suggested. The termination criterion of RAER are set as $|f| < 1\mathrm{E}{-2}$. For every set of different parameter settings, we performed the tests 50 times. The following indices are used to measure the performance of RAER: average number of function evaluations ($\mathrm{FE}_{\mathrm{avg}}$), average generations of convergence ($G_{\mathrm{avg}}$), standard deviation of $G_{\mathrm{avg}}$ (*Std* of $G_{\mathrm{avg}}$), standard deviation of best solutions (*Std* of solutions).

Table 1 demonstrates the performance of REAR along the problem dimension ranging 30–100 when scale of $L_{M\times N}$ varies. When $L_{M\times N}$ is larger than $L_{3\times 3}$, RAER performs with better efficiency. For $f_{01}$ with dimension of 30 REAR can achieve good stability and efficiency except that the *Std* of $G_{\mathrm{avg}}$ gets a little larger with scale of $L_{M\times N}$ decreasing. And when problem dimension increases, the smaller the scale of $L_{M\times N}$, the larger the *Std* of $G_{\mathrm{avg}}$, which shows that the smaller the scale of $L_{M\times N}$, the less stability the REAR, although RAER performs with good efficiency when $L_{M\times N}$ larger than $L_{3\times 3}$. According to the statistical results in Table 1, when $L_{M\times N}$ is larger than $L_{6\times 6}$, REAR can achieve good stability and efficiency, and larger $L_{M\times N}$ may make RAER more stable for large-scale problems.

Table 2 shows the effect of $\mathrm{RL}_{m\times n}$ scale and MGen on the performance of REAR. The performance of RAER were monitored for variation of $\mathrm{RL}_{m\times n}$ ($\mathrm{RL}_{2\times 2}$–$\mathrm{RL}_{4\times 5}$) and MGen(4–12). From Table 2, we can find that the $G_{\mathrm{avg}}$ and std of $G_{\mathrm{avg}}$ decrease when the scale of $\mathrm{RL}_{m\times n}$ increases, and when $\mathrm{RL}_{m\times n}$ is larger than $\mathrm{RL}_{3\times 3}$ RAER can achieve good stability and efficiency. We can also find the same case for $\mathrm{RL}_{3\times 3}$ when MGen is larger than 5. In general, when large-scale of $\mathrm{RL}_{m\times n}(\geqslant \mathrm{RL}_{3\times 3})$ or large MGen($\geqslant$6)

**Table 1**
Results of REAR for different scale of $L(M,N)$ along problem dimension ranging 30–100.

| $L(M,N)$ | $N$ | $G_{avg}$ | *Std* of $G_{avg}$ | $FE_{avg}$ | *Std* of solutions |
|---|---|---|---|---|---|
| $L(3,3)$ | 30 | 427 | 96 | 60033 | 8.1E−03 |
| | 50 | 1485 | 549 | 215690 | 1.8E−03 |
| | 100 | 2529 | 861 | 399567 | 4.8E−03 |
| $L(4,5)$ | 30 | 221 | 68 | 49528 | 5.8E−03 |
| | 50 | 661 | 103 | 149403 | 3.9E−03 |
| | 100 | 927 | 358 | 212061 | 7.1E−03 |
| $L(5,5)$ | 30 | 208 | 69 | 50164 | 2.0E−03 |
| | 50 | 587 | 96 | 139374 | 7.4E−03 |
| | 100 | 843 | 297 | 208691 | 3.5E−03 |
| $L(6,6)$ | 30 | 177 | 48 | 49785 | 3.9E−03 |
| | 50 | 498 | 72 | 148755 | 2.2E−03 |
| | 100 | 716 | 246 | 209753 | 6.1E−03 |
| $L(8,8)$ | 30 | 158 | 24 | 44869 | 5.3E−03 |
| | 50 | 405 | 59 | 129769 | 2.8E−03 |
| | 100 | 647 | 162 | 196123 | 5.9E−03 |
| $L(10,10)$ | 30 | 133 | 18 | 46327 | 3.5E−03 |
| | 50 | 384 | 49 | 127452 | 2.4E−03 |
| | 100 | 589 | 147 | 199721 | 5.2E−03 |
| $L(12,12)$ | 30 | 117 | 19 | 48956 | 5.5E−03 |
| | 50 | 351 | 42 | 139169 | 3.9E−03 |
| | 100 | 533 | 111 | 207919 | 4.4E−03 |

*Note*: $P_{nd} = 0.75,\ P^5_{nd} = 0.5,\ P_s = 0.2,\ \mathrm{RP} = 0.1,\ \mathrm{RL}_{3\times3},\ \mathrm{RP}_{nd} = 0.2,\ \mathrm{RP}^5_{nd} = 0.9,\ \mathrm{RP}_s = 0.1,\ \mathrm{MGen} = 6.$

**Table 2**
Results of RAER for different scale of $\mathrm{RL}_{m\times n}$ and different values of MGen.

| RL$(m,n)$ | MGen | $G_{avg}$ | *Std* of $G_{avg}$ | $FE_{avg}$ | *Std* of solutions | RL$(m,n)$ | MGen | $G_{avg}$ | *Std* of $G_{avg}$ | $FE_{avg}$ | *Std* of solutions |
|---|---|---|---|---|---|---|---|---|---|---|---|
| RL(2,2) | 6 | 186 | 41 | 54253 | 6.2E−03 | RL(3,3) | 4 | 159 | 37 | 48783 | 2.8E−03 |
| RL(2,3) | 6 | 179 | 36 | 54941 | 1.4E−03 | RL(3,3) | 5 | 149 | 28 | 47794 | 1.9E−03 |
| RL(3,3) | 6 | 133 | 18 | 46327 | 3.5E−03 | RL(3,3) | 6 | 133 | 18 | 46327 | 3.5E−03 |
| RL(3,4) | 6 | 128 | 19 | 46946 | 1.6E−03 | RL(3,3) | 8 | 123 | 19 | 45688 | 3.8E−03 |
| RL(4,4) | 6 | 119 | 13 | 47213 | 3.7E−03 | RL(3,3) | 10 | 121 | 17 | 46721 | 3.4E−03 |
| RL(4,5) | 6 | 104 | 12 | 48424 | 4.5E−03 | RL(3,3) | 12 | 111 | 13 | 47822 | 4.9E−03 |

*Note*: $L_{10\times10},\ P_{nd} = 0.75,\ P^5_{nd} = 0.5,\ P_s = 0.2,\ \mathrm{RP} = 0.1,\ \mathrm{RP}_{nd} = 0.2,\ \mathrm{RP}^5_{nd} = 0.9,\ \mathrm{RP}_s = 0.1.$

**Table 3**
Results of REAR for different values of $P_{nd}$.

| $P_{nd}$ | $G_{avg}$ | *Std* of $G_{avg}$ | $FE_{avg}$ | *Std* of solutions | $P_{nd}$ | $G_{avg}$ | *Std* of $G_{avg}$ | $FE_{avg}$ | *Std* of solutions |
|---|---|---|---|---|---|---|---|---|---|
| 0.0 | 245 | 107 | 63104 | 1.8E−03 | 0.6 | 153 | 21 | 48149 | 1.1E−02 |
| 0.1 | 209 | 80 | 52874 | 4.0E−03 | 0.75 | 133 | 18 | 46327 | 3.5E−03 |
| 0.25 | 190 | 42 | 55855 | 3.4E−03 | 0.8 | 140 | 23 | 47504 | 5.9E−03 |
| 0.4 | 179 | 34 | 53991 | 3.3E−03 | 0.9 | 135 | 26 | 46741 | 9.2E−03 |
| 0.5 | 159 | 24 | 49734 | 1.5E−03 | 1.0 | 168 | 25 | 57059 | 2.7E−03 |

*Note*: $L_{10\times10},\ P^5_{nd} = 0.5,\ P_s = 0.2,\ \mathrm{RP} = 0.1,\ \mathrm{RL}_{3\times3},\ \mathrm{RP}_{nd} = 0.2,\ \mathrm{RP}^5_{nd} = 0.9,\ \mathrm{RP}_s = 0.1,\ \mathrm{MGen} = 6.$

**Table 4**
Results of REAR for different values of $\mathrm{RP}_{nd}$.

| $\mathrm{RP}_{nd}$ | $G_{avg}$ | *Std* of $G_{avg}$ | $FE_{avg}$ | *Std* of solutions | $\mathrm{RP}_{nd}$ | $G_{avg}$ | *Std* of $G_{avg}$ | $FE_{avg}$ | *Std* of solutions |
|---|---|---|---|---|---|---|---|---|---|
| 0.0 | 172 | 28 | 55735 | 6.7E−03 | 0.5 | 145 | 22 | 49671 | 2.9E−03 |
| 0.1 | 144 | 24 | 47853 | 6.3E−03 | 0.6 | 170 | 25 | 59457 | 3.3E−03 |
| 0.2 | 133 | 18 | 46327 | 3.5E−03 | 0.7 | 192 | 17 | 67207 | 6.1E−03 |
| 0.3 | 139 | 16 | 47284 | 4.9E−03 | 0.8 | 206 | 27 | 71382 | 1.5E−03 |
| 0.4 | 137 | 15 | 47890 | 2.3E−03 | 1.0 | 243 | 22 | 83376 | 5.5E−03 |

*Note*: $L_{10\times10},\ P_{nd} = 0.75,\ P^5_{nd} = 0.5,\ P_s = 0.2,\ \mathrm{RP} = 0.1,\ \mathrm{RL}_{3\times3},\ \mathrm{RP}^5_{nd} = 0.9,\ \mathrm{RP}_s = 0.1,\ \mathrm{MGen} = 6.$

**Table 5**
Results of REAR for different values of $P_{nd}^5$.

| $P_{nd}^5$ | $G_{avg}$ | *Std* of $G_{avg}$ | $FE_{avg}$ | *Std* of solutions | $P_{nd}^5$ | $G_{avg}$ | *Std* of $G_{avg}$ | $FE_{avg}$ | *Std* of solutions |
|---|---|---|---|---|---|---|---|---|---|
| 0.0 | 223 | 16 | 57476 | 4.9E−04 | 0.6 | 127 | 19 | 45621 | 3.8E−03 |
| 0.1 | 211 | 16 | 61069 | 4.2E−03 | 0.7 | 139 | 20 | 48229 | 5.7E−03 |
| 0.3 | 172 | 22 | 53716 | 9.3E−04 | 0.9 | 149 | 27 | 52559 | 1.9E−04 |
| 0.5 | 133 | 18 | 46327 | 3.5E−03 | 1.0 | 160 | 24 | 54674 | 1.0E−04 |

*Note*: $L_{10\times10},\ P_{nd}=0.75,\ P_s=0.2,\ \mathrm{RP}=0.1,\ \mathrm{RL}_{3\times3},\ \mathrm{RP}_{nd}=0.2,\ \mathrm{RP}_{nd}^5=0.9,\ \mathrm{RP}_s=0.1,\ \mathrm{MGen}=6.$

**Table 6**
Results of REAR for different values of RP.

| RP | $G_{avg}$ | *Std* of $G_{avg}$ | $FE_{avg}$ | *Std* of solutions | RP | $G_{avg}$ | *Std* of $G_{avg}$ | $FE_{avg}$ | *Std* of solutions |
|---|---|---|---|---|---|---|---|---|---|
| 0.00 | 313 | 67 | 100791 | 3.7E−03 | 0.2 | 143 | 21 | 48334 | 5.5E−03 |
| 0.05 | 218 | 43 | 73598 | 7.4E−03 | 0.3 | 159 | 19 | 51339 | 5.8E−03 |
| 0.1 | 133 | 18 | 46327 | 3.5E−03 | 0.4 | 181 | 16 | 59664 | 2.9E−03 |

*Note*: $L_{10\times10},\ P_{nd}=0.75,\ P_s=0.2,\ P_{nd}^5=0.5,\ \mathrm{RL}_{3\times3},\ \mathrm{RP}_{nd}=0.2,\ \mathrm{RP}_{nd}^5=0.9,\ \mathrm{RP}_s=0.1,\ \mathrm{MGen}=6.$

are given RAER may achieve good stability and efficiency, and the larger the scale of $\mathrm{RL}_{m\times n}$ or MGen, the more computational cost the REAR may need.

Table 3 shows the effect of different values of $P_{nd}$ on the performance of RAER. When $P_{nd}$ is given small values ($P_{nd} < 0.5$), RAER performs with less stability and needs more computational cost. When $P_{nd}$ is given values between 0.5 and 0.9, $FE_{avg}$ and *Std* of $G_{avg}$ become smaller. When $P_{nd}$ is set to 1.0, in fact, REAR behaves similar to another stable multiagent genetic algorithm in Ref. [20], but needs more computational cost. In general, when $P_{nd}$ is given values between 0.5 and 0.9, REAR may achieve good stability and efficiency.

Table 4 demonstrates the effect of different values of $\mathrm{RP}_{nd}$ on the performance of REAR. When $\mathrm{RP}_{nd}$ is set to 0.0, the REAR performs relatively unstable (*Std* of $G_{avg}$ is 28). When $\mathrm{RP}_{nd}$ increases larger than 0.5, the $FE_{avg}$ increases rather fast and *Std* of $G_{avg}$ gets large relatively. When $\mathrm{RP}_{nd}$ is set to 0.1–0.5, REAR needs less computational cost ($FE_{avg}$ shows smaller), and when $\mathrm{RP}_{nd}$ is set to 0.2–0.4, REAR performs more stable ($FE_{avg}$ and *Std* of $G_{avg}$ both shows smaller).

To evaluate the effect of $P_{nd}^5$ on the performance of REAR, we have monitored the performance of REAR for variation of $P_{nd}^5$ (0.0–1.0). The experimental results are presented in Table 5. The results show that REAR needs more computational cost when the two local dispersal strategies work respectively ($P_{nd}^5 = 0.0,\ P_{nd}^5 = 1.0$) than the case when they work collaboratively. According to the statistical results REAR may achieve good stability and efficiency when $P_{nd}^5$ is chosen between 0.5 and 0.7.

Table 6 shows the effect of varying diversity control multiplier RP on the performance of the RAER. When the RP is larger than 0.05 RAER can achieve better stability. And when RP varies between 0.1 and 0.3 RAER performs more efficiently. In essence, the RP strategy is a basic method to maintain diversity in evolutionary algorithms [21,22], and obviously the larger the RP, the more heterogeneous the population. But population diversity and selective pressure are inversely related [23], maintaining population diversity offsets the effect of increasing selective pressure [24]. Table 6 shows when RP is larger than 0.3 RAER takes more generations to achieve best solutions although its stability is also good.

According to the above-discussed statistical experimental results, parameter settings of REAR are suggested in Table 7. For more details about $\mathrm{RP}_{nd}^5$, see Refs. [25,26]. For more details about $P_s$ and $\mathrm{RP}_s$, see Refs. [13,20]. For more details about uniform design technique, see Ref. [25].

## 4.2. Comparative experiments on the performance of roulette inversion operator (RIO) and Holland's Inversion Operator (IO)

In this section, function $f_{01}$ and following three benchmark functions are utilized to test the capability of the proposed roulette inversion operator (RIO) and Holland's inversion operator comparatively. In experiments, we replace the RIO operator with IO operator in RAER to form a new algorithm denoted as RAER_IO so that the performance of RIO and IO can be

**Table 7**
Parameter settings of REAR.

| Parameters | Value | Parameters | Value |
|---|---|---|---|
| $L_{M\times N}$ | $L_{10\times10}$ ($\geqslant L_{4\times5}$) | $\mathrm{RL}_{m\times n}$ | $\mathrm{RL}_{3\times3}$ ($\geqslant \mathrm{RL}_{3\times3}$) |
| $P_{nd}$ | 0.75 (0.5–0.9) | $\mathrm{RP}_{nd}$ | 0.2 (0.2–0.4) |
| $P_{nd}^5$ | 0.5 (0.5–0.7) | $\mathrm{RP}_{nd}^5$ | 0.9 |
| RP | 0.1 (0.1–0.3) | $\mathrm{RP}_s$ | 0.1 |
| $P_s$ | 0.2 | MGen | 6 ($\geqslant$6) |
| $U(m, q)$ | $U(5,4)$, $U(11,4)$ | | |

examined by testing the performance of RAER and RAER_IO. The parameters for the following experiments in this section are set according to Table 7:

$f_{02}$: Hwang & He's Function 15:

$$\mathrm{Max} f(x_i) = sum_{i=1}^{N}(-1)^{i+1}x_i^2, \quad x_i \in [-100, 100]^N;$$

$f_{03}$: De-Jong's Function with noise:

$$\mathrm{Min} f(x_i) = \sum_{i=1}^{N}(x_i^4) + \mathrm{random}(0,1), \quad x_i \in [-1.28, 1.28]^N;$$

$f_{04}$: Sphere Function:

$$\mathrm{Min} f(x_i) = \sum_{i=1}^{N} x_i^2, \quad x_i \in [-100, 100]^N;$$

Average function evaluations ($C_{avg}$) and standard deviation (*std*) of optimal solutions by RAER and RAER_IO among 20 independent runs on rather difficult $f_{01}$ and $f_{02}$ along dimensions ranging 10–50 are listed in Table 8. For more clarity, here we utilize $O(N^a)$ to approximate the relationship of and $C_{avg}$ and $N$ as shown in Fig. 2. The complexities of RAER on $f_{01}$ and $f_{02}$ are $O(N^{1.92})$ and $O(N^{0.53})$, which are lower than those $O(N^{1.94})$ and $O(N^{0.58})$ of RAER_IO. From Table 8 and Fig. 1, conclusions can be drawn that the performance of RAER is better than RAER_IO, that is, the performance of RIO is better than IO. And the advantage of RIO becomes larger along with the increase of problems' dimensions, which is consistent with Theorems 1 and 2.

The experiment results of three algorithm ARSAGA [4], RAER and RAER_IO on $f_{01}$–$f_{04}$ with dimensions of 30 are listed in Table 9. As can be seen, the $C_{avg}$ of RAER and RAER_IO are lower than ARSAGA significantly and the optimal solutions achieved are more precise. Especially for $f_{04}$, as the precision of optimal solution required is high, the precision of each component of the optimal solution is high correspondingly. And in comparison with RAER_IO, REAR can treat each dimension evenly so as to explore the search space more effectively. So RAER only needs the $C_{avg}$ about 4 times less than that of RAER_IO to achieve the optimal solution with same high precision (E−15).

## 4.3. The performance of proposed algorithm RAER on benchmark functions

Here, the proposed algorithm, RAER, is tested by the following 12 benchmark unconstrained global numerical optimization problems. These functions have different characteristics: large search space, determinism/randomicity, unimodal/multimodal, and quadratic/nonquadratic. For examples, $F_{02}$ has a lot of local minima in the given search space, the function value

**Table 8**
Average function evaluations ($C_{avg}$) and standard deviations (*std*) of REAR and REAR_IO on $f_{01}$ and $f_{02}$ when $N$ = 10–50.

| *N* | $f_{01}$ | | | | $f_{02}$ | | | |
|---|---|---|---|---|---|---|---|---|
| | RAER | | RAER_IO | | RAER | | RAER_IO | |
| | *std*(E−3) | $C_{avg}$ | *std*(E−3) | $C_{avg}$ | *std*(E−3) | $C_{avg}$ | *std*(E−3) | $C_{avg}$ |
| 10 | 1.8 | 6131 | 1.4 | 6157 | 2.4 | 9595 | 2.5 | 11967 |
| 20 | 7.2 | 18870 | 8.9 | 22446 | 1.7 | 17483 | 1.9 | 20403 |
| 30 | 3.5 | 46327 | 7.3 | 47485 | 2.8 | 22636 | 1.9 | 29505 |
| 40 | 1.9 | 79920 | 3.3 | 84478 | 1.7 | 25108 | 1.9 | 32628 |
| 50 | 2.4 | 127452 | 2.8 | 134629 | 1.7 | 27272 | 1.9 | 34379 |

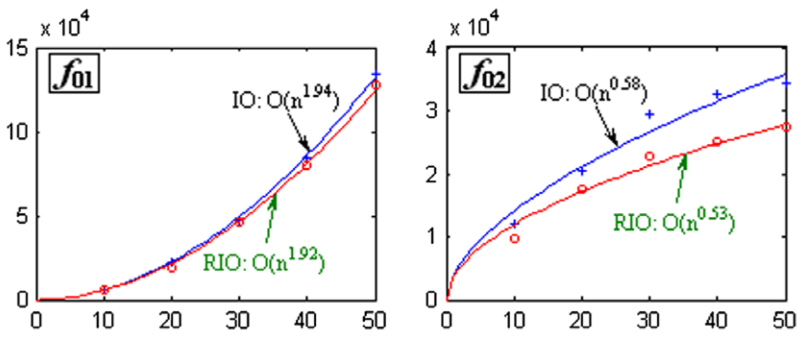

**Fig. 2.** The relationship between $C_{avg}$ and $N$ of RAER, REAR_IO on $f_{01}$ and $f_{02}$.

**Table 9**
Average function evaluations ($C_{avg}$) and standard deviations (*std*) of REAR, REAR_IO and ARSAGA on $f_{01}$–$f_{04}$ when $N = 30$.

| Fun | | $C_{avg}$ | | | *Std* | | |
|---|---|---|---|---|---|---|---|
| *F* | *N* | ARSAGA | RAER_IO | RAER | ARSAGA | RAER_IO | RAER |
| $f_{01}$ | 30 | 363979 | 47485 | 46327 | 2.7E−2 | 7.3E−3 | 3.5E−3 |
| $f_{02}$ | 30 | 254541 | 29505 | 22636 | 6.6E−2 | 1.9E−3 | 2.8E−3 |
| $f_{03}$ | 30 | 15406 | 3396 | 2972 | 4.7E−3 | 1.1E−3 | 1.0E−3 |
| $f_{04}$ | 30 | 107380 | 49284 | 11953 | 0 | 0(E−15) | 0(E−15) |

of $F_{10}$ randomly varies when calculated every time. And for the test in this section, the scalability of RAER along the problem dimensions is studied with great care, that is, the performance of RAER on $F_{01}$–$F_{08}$ with 20–10 000 dimensions and more difficult $F_{09}$–$F_{12}$ with 20–100 dimensions is tested. In order to compare the performance of RAER with small population as that in Refs. [4,13], $L_{M\times N}$ is set as $L_{4\times 5}$, other parameters are set according to Table 7.

$F_{01}$: Generalized Schwefel's Problem 2.26:

$$\mathrm{Min} f(x_i) = \sum_{i=1}^{N} -x_i \sin\left(\sqrt{|x_i|}\right), x_i \in [-500, 500]^N.$$

$F_{02}$: Rastrigin's Problem:

$$\mathrm{Min} f(x_i) = \sum_{i=1}^{N} (x_i^2 - A\cos(2\pi x_i) + A), A = 10, x_i \in [-5.12, 5.12]^N.$$

$F_{03}$: Ackley's Problem:

$$\mathrm{Min} f(x_i) = -20\exp\left[-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}\right] - \exp\left[\frac{1}{N}\sum_{i=1}^{N}\cos(2\pi x_i)\right] + 20 + \exp(1), \quad x_i \in [-32, 32]^N.$$

$F_{04}$: Griewank's Problem:

$$\mathrm{Min} f(x_i) = \frac{1}{4000}\sum_{i=1}^{N} x_i^2 - \prod_{i=1}^{N}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, \quad x_i \in [-600, 600]^N.$$

$F_{05}$: Generalized Penalized Function 1:

$$\mathrm{Min} f(x_i) = \frac{\pi}{N}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{N-1}(y_i - 1)^2 \times [1 + 10\sin^2(\pi y_{i+1})] + (y_N - 1)^2\right\} + \sum_{i=1}^{N} u(x_i, 5, 100, 4),$$

$$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a, \\ 0, & -a \leqslant x_i \leqslant a, \\ k(-x_i - a)^m, & x_i < -a, \end{cases}$$

$$y_i = 1 + \frac{1}{4}(x_i + 1); \quad x_i \in [-50, 50]^N;$$

$F_{06}$: Generalized Penalized Function 2:

$$\mathrm{Min} f(y_i) = \frac{\pi}{N}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{N-1}(y_i - 1)^2 \times [1 + 10\sin^2(\pi y_{i+1})] + (y_N - 1)^2\right\} + \sum_{i=1}^{N} u(y_i, 5, 100, 4),$$

$$u(y_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a, \\ 0, & -a \leqslant x_i \leqslant a, \\ k(-x_i - a)^m, & x_i < -a, \end{cases}$$

$$y_i \in [-50, 50]^N;$$

$F_{07}$: Sphere Function:

$$\mathrm{Min} f(x_i) = \sum_{i=1}^{N} x_i^2, \quad x_i \in [-100, 100]^N;$$

$F_{08}$: Schwefel's Problem 2.22:

$$\mathrm{Min} f(x_i) = \sum_{i=1}^{N} |x_i| + \prod_{i=1}^{N} |x_i|, \quad x_i \in [-10, 10]^N;$$

$F_{09}$: Zakharov's Function:

$$\mathrm{Min} f(x_i) = \sum_{i=1}^{N} x_i^2 + \left(\sum_{i=1}^{N} \frac{i}{2} x_i\right)^2 + \left(\sum_{i=1}^{N} \frac{i}{2} x_i\right)^4, \quad x_i \in [-5, 10]^N;$$

$F_{10}$: De-Jong's Function with noise:

$$\mathrm{Min} f(x_i) = \sum_{i=1}^{N} (x_i^4) + \mathrm{random}(0,1), \quad x_i \in [-1.28, 1.28]^N;$$

$F_{11}$: Hwang & He's Function 15:

$$\mathrm{Max} f(x_i) = \sum_{i=1}^{N} (-1)^{i+1} x_i^2, \quad x_i \in [-100, 100]^N;$$

$F_{12}$: Ellipsoidal Function:

$$\mathrm{Min} f(x_i) = \sum_{i=1}^{N} (x_i - i)^2, \quad x_i \in [-100, 100]^N;$$

### *4.3.1. Comparative experiments between FEP, OGA/Q, MAGA and RAER on functions $F_{01}$–$F_{08}$ with 30 dimensions*

FEP [27], OGA/Q [5] and MAGA [13] are optimization methods proposed recently and obtained good performance on numerical optimization problems. FEP is a modified evolutionary programming (EP) algorithm and outperforms classical EP (CEP) for a suite of benchmark problems. OGA/Q is a modified version of the classical genetic algorithm (CGA) using orthogonal design experimental sampling method to generate the initial population and offspring of the crossover operator. OGA/Q was compared with traditional GAs and five existing algorithms and the results showed that OGA/Q outperforms all the other methods. MAGA implements a multiagent genetic algorithm based on ERA pattern and is a competent algorithm for solving high-dimensional optimization problems.

In Ref. [27], the termination criterion of FEP was to run 1500 generations for $F_{03}$ and $F_{05}$–$F_{07}$, 2000 generations for $F_{04}$ and $F_{08}$, 5000 generations for $F_{02}$, and 9000 generations for $F_{01}$. In Ref. [5], the termination criterion of OGA/Q was the quality of the solution cannot be further improved in successive 50 generations after 1000 generations. In Ref. [13], the termination of MAGA is to run 150 generations for each benchmark function. For the experiments in this section, the termination criterions of RAER are set as follows: $f < -12569.4866$ for $F_{01}$, $|f| < 4.8\mathrm{E}{-16}$ for $F_{03}$, $|f| < 1\mathrm{E}{-18}$ for $F_{05}$, $|f| < 1\mathrm{E}{-17}$ for $F_{06}$, and $|f| < 1\mathrm{E}{-19}$ (considered as 0) for the other functions ($F_{02}$, $F_{04}$, $F_{07}$, $F_{08}$). The mean number and the standard deviations of function evaluations averaged over 30 trials are shown in Table 10, where $N = 30$.

As shown in Table 10, the performance of RAER and MAGA is much better than FEP for all functions. And for all eight functions the computational cost of RAER and MAGA is much lower than those of OGA/Q. For $F_{01}$ RAER only needs 5936 function evaluations (about a half that of MAGA) to find global optimum $-12569.4866$. For $F_{05}$ RAER needs slightly more function evaluations than MAGA to achieve E−19 (E−18 for MAGA) and for the other functions RAER can achieve $|f| < 1\mathrm{E}{-19}$ (could be considered as 0) with function evaluations lower than 10000, which is equivalent with MAGA. To sum up, for $F_{01}$–$F_{08}$ with $N = 30$, RAER is significantly better than FEP and OGA/Q and better than MAGA in most cases, which shows that RAER is competent for the numerical optimization problems.

**Table 10**
Comparison between FEP, OGA/Q, MAGA and REAR on $F_{01}$–$F_{08}$ when $N = 30$.

| $F$ | $F_{min}$ | Mean function value (standard deviation) | | | | Mean number of function evaluations | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | FEP | OGA/Q | MAGA | RAER | FEP | OGA/Q | MAGA | RAER |
| $F_{01}$ | −12569.5 | −12554.5 (52.6) | −12569.4537 ($6.447 \times 10^{-4}$) | −12569.4866 ($7.121 \times 10^{-6}$) | −12569.4866 ($2.858 \times 10^{-6}$) | 900000 | 302166 | 10862 | 5936 |
| $F_{02}$ | 0 | $4.6 \times 10^{-2}$ ($1.2 \times 10^{-2}$) | 0 (0) | 0 (0) | 0 (0) | 500000 | 224710 | 11427 | 9052 |
| $F_{03}$ | 0 | $1.8 \times 10^{-2}$ ($2.1 \times 10^{-3}$) | $4.440 \times 10^{-16}$ ($3.989 \times 10^{-17}$) | $4.440 \times 10^{-16}$ (0) | $4.530 \times 10^{-16}$ ($3.517 \times 10^{-17}$) | 150000 | 112421 | 9656 | 8772 |
| $F_{04}$ | 0 | $1.6 \times 10^{-2}$ ($2.2 \times 10^{-2}$) | 0 (0) | 0 (0) | 0 (0) | 200000 | 134000 | 9777 | 6618 |
| $F_{05}$ | 0 | $9.2 \times 10^{-6}$ ($3.6 \times 10^{-6}$) | $6.019 \times 10^{-6}$ ($1.159 \times 10^{-6}$) | $1.142 \times 10^{-18}$ ($4.390 \times 10^{-18}$) | $8.346 \times 10^{-19}$ ($1.894 \times 10^{-19}$) | 150000 | 134556 | 10545 | 10593 |
| $F_{06}$ | 0 | $1.6 \times 10^{-4}$ ($7.3 \times 10^{-5}$) | $1.869 \times 10^{-4}$ ($2.615 \times 10^{-5}$) | $1.039 \times 10^{-17}$ ($4.196 \times 10^{-17}$) | $7.964 \times 10^{-18}$ ($1.889 \times 10^{-18}$) | 150000 | 134143 | 11269 | 9905 |
| $F_{07}$ | 0 | $5.7 \times 10^{-4}$ ($1.3 \times 10^{-4}$) | 0 (0) | 0 (0) | 0 (0) | 150000 | 112559 | 9502 | 8471 |
| $F_{08}$ | 0 | $8.1 \times 10^{-2}$ ($7.7 \times 10^{-3}$) | 0 (0) | 0 (0) | 0 (0) | 200000 | 112612 | 9591 | 9047 |

### *4.3.2. Comparative experiments between RAER and MAGA on functions $F_{01}$–$F_{08}$ with 20–1000 dimensions*

In this subsection, comparative experiments with MAGA [13] are taken to study the performance of RAER on $F_{01}$–$F_{08}$ with dimensions varying 20–1000. The termination criterion of MAGA is one of the objectives, $|f_{best} - f_{min}| < \varepsilon \times |f_{min}|$ or $|f_{best}| < \varepsilon$ if $f_{min} = 0$, is achieved, where $\varepsilon = 1\text{E}{-4}$, $f_{best}$ and $f_{min}$ represent the best solution found until the current generation and the global optimum, respectively. The termination criterion of RAER is set as follows: $f < -418.98288 \times N$ for $F_{01}$, $|f| < 1\text{E}{-5}$ for $F_{03}$, $|f| < 1\text{E}{-4}$ for $F_{02}$ and $F_{04}$–$F_{08}$. The mean number and the standard deviations of function evaluations averaged over 30 trials are shown in Table 11.

As shown in Table 11, for $F_{01}$–$F_{08}$ the results of RAER is better than that of MAGA and with the dimension increasing to 1000 RAER only needs thousands of function evaluations at all selected dimensions to achieve solutions of high quality. Especially for $F_{01}$, RAER can achieve solutions with standard deviation of E−04 other than solutions with large standard deviation of MAGA under its termination criterion.

Refs. [28,6,29] also applied BGA, AEA and IMCPA on $F_{01}$–$F_{04}$ with dimensions of 20, 100, 200, 400, 1000. The termination criterions of BGA and AEA were the same as that of MAGA, but they used $\varepsilon = 1\text{E}{-4}$ for $F_{01}$, $\varepsilon = 1\text{E}{-1}$ for $F_{02}$, and $\varepsilon = 1\text{E}{-3}$ for $F_{03}$ and $F_{04}$. The termination criterions of IMCPA were: $\varepsilon = 1\text{E}{-2}$ for $F_{01}$, $\varepsilon = 1\text{E}{-1}$ for $F_{02}$, $\varepsilon = 1\text{E}{-3}$ for $F_{03}$, and $\varepsilon = 1\text{E}{-4}$ for $F_{04}$. So here we list all the results obtained from different algorithms in Table 12 for a comparison.

**Table 11**
Comparison between MAGA and REAR on $F_{01}$–$F_{08}$ when $N$ = 20–1000.

| $N$ | $F_{01}$ | | $F_{02}$ | | $F_{03}$ | | $F_{04}$ | |
|---|---|---|---|---|---|---|---|---|
| | MAGA ($\times 10^{-0}$) | RAER ($\times 10^{-4}$) | MAGA ($\times 10^{-5}$) | RAER ($\times 10^{-5}$) | MAGA ($\times 10^{-6}$) | RAER ($\times 10^{-6}$) | MAGA ($\times 10^{-5}$) | RAER ($\times 10^{-5}$) |
| 20 | 2483 | 2151 | 4301 | 3079 | 3583 | 3196 | 2566 | 2303 |
| | (0.2) | (0.373) | (1.9) | (1.745) | (9.3) | (1.330) | (2.3) | (2.159) |
| 100 | 5443 | 4409 | 10265 | 3449 | 5410 | 4682 | 4447 | 4154 |
| | (0.9) | (0.9013) | (1.6) | (1.711) | (8.8) | (0.934) | (2.1) | (1.338) |
| 200 | 7284 | 5049 | 14867 | 3923 | 6051 | 5304 | 5483 | 4359 |
| | (2.7) | (1.535) | (2.9) | (1.543) | (6.9) | (0.984) | (2.1) | (1.540) |
| 400 | 12368 | 5613 | 17939 | 4601 | 6615 | 5977 | 6249 | 5356 |
| | (6.1) | (5.308) | (2.7) | (1.737) | (7.1) | (0.843) | (1.8) | (1.661) |
| 800 | 19992 | 6505 | 20306 | 4783 | 7069 | 6544 | 6883 | 5893 |
| | (11) | (4.890) | (3.6) | (1.263) | (7.5) | (0.787) | (1.7) | (2.061) |
| 1000 | 22827 | 7439 | 20083 | 4894 | 7288 | 6772 | 7358 | 6218 |
| | (14) | (8.425) | (3.7) | (1.379) | (8.6) | (0.624) | (1.8) | (0.916) |
| $N$ | $F_{05}$ | | $F_{06}$ | | $F_{07}$ | | $F_{08}$ | |
| | MAGA ($\times 10^{-5}$) | RAER ($\times 10^{-5}$) | MAGA ($\times 10^{-5}$) | RAER ($\times 10^{-5}$) | MAGA ($\times 10^{-5}$) | RAER ($\times 10^{-5}$) | MAGA ($\times 10^{-5}$) | RAER ($\times 10^{-5}$) |
| 20 | 2827 | 2729 | 3745 | 2891 | 2420 | 2221 | 2956 | 1379 |
| | (1.9) | (1.584) | (1.9) | (1.81) | (1.6) | (1.822) | (1.0) | (1.287) |
| 100 | 4907 | 3182 | 7929 | 3837 | 4199 | 3848 | 5638 | 2059 |
| | (2.8) | (1.368) | (2.1) | (1.208) | (1.3) | (1.605) | (0.79) | (0.998) |
| 200 | 6870 | 3313 | 9732 | 3865 | 4966 | 4893 | 6757 | 2205 |
| | (3.2) | (1.603) | (2.2) | (1.308) | (1.6) | (1.462) | (0.87) | (0.795) |
| 400 | 9305 | 3874 | 12820 | 3988 | 5576 | 5291 | 7753 | 2293 |
| | (3.7) | (1.501) | (3.3) | (0.984) | (1.2) | (1.409) | (0.83) | (0.781) |
| 800 | 10572 | 4089 | 16070 | 4197 | 6079 | 5653 | 8692 | 2578 |
| | (3.0) | (1.608) | (3.2) | (1.67) | (1.4) | (1.304) | (0.83) | (0.833) |
| 1000 | 11214 | 4261 | 17829 | 4363 | 6273 | 5781 | 9456 | 2993 |
| | (3.6) | (1.642) | (3.5) | (0.735) | (1.4) | (0.816) | (0.72) | (0.775) |

**Table 12**
Comparison between BGA, AEA, IMCPA, MAGA and REAR on $F_{01}$–$F_{04}$ when $N$ = 20–1000.

| $N$ | BGA | AEA | IMCPA | MAGA | RAER | BGA | AEA | IMCPA | MAGA | RAER |
|---|---|---|---|---|---|---|---|---|---|---|
| | $F_{01}$ | | | | | $F_{02}$ | | | | |
| 20 | 16100 | 1603 | 3939 | 2483 | 2151 | 3608 | 1247 | 1469 | 4301 | 3079 |
| 100 | 92000 | 5106 | 11896 | 5443 | 4409 | 25040 | 4789 | 4988 | 10265 | 3449 |
| 200 | 248000 | 8158 | 16085 | 7284 | 5049 | 52948 | 10370 | 5747 | 14867 | 3923 |
| 400 | 700000 | 13822 | 26072 | 12368 | 5613 | 112634 | 23588 | 12563 | 17939 | 4601 |
| 1000 | – | 23867 | 60720 | 22827 | 7439 | 337570 | 46024 | 24408 | 20083 | 4894 |
| | $F_{03}$ | | | | | $F_{04}$ | | | | |
| 20 | 19420 | 7040 | 1776 | 3583 | 3196 | 66000 | 3581 | 2421 | 2566 | 2303 |
| 100 | 53860 | 22710 | 5784 | 5410 | 4682 | 361722 | 17228 | 6713 | 4447 | 4154 |
| 200 | 107800 | 43527 | 9728 | 6051 | 5304 | 748300 | 36760 | 8460 | 5483 | 4359 |
| 400 | 220820 | 78216 | 13915 | 6615 | 5977 | 1630000 | 61975 | 15365 | 6249 | 5356 |
| 1000 | 548306 | 160940 | 26787 | 7288 | 6772 | – | 97660 | 30906 | 7358 | 6218 |

As can be drawn in Table 12, AEA and IMCPA get some advantages over MAGA and RAER with dimensions of 20, but it deserves attention that MAGA and RAER have stricter termination criterions than AEA and IMCPA. When dimensions are larger than 20 MAGA and RAER are better than other algorithms, and RAER is slightly better than MAGA. Especially for $F_{01}$, RAER can achieve standard deviations of E−04 but other algorithms' deviations are rather large. In general, RAER can achieve better solutions of functions with high dimensions than BGA, AEA as MAGA under the strictest termination criterions at lower computational costs, and RAER also can be considered as equivalent to BGA or AEA for solving lower-dimensional functions.

### 4.3.3. Comparative experiments between RAER and MAGA on functions $F_{01}$–$F_{08}$ with 1000–10000 dimensions

In this subsection, RAER is applied on $F_{01}$–$F_{08}$ with higher dimensions ranging 1000–10000. In order to compare the performance of RAER and MAGA, here we execute RAER for 30 runs on $F_{01}$–$F_{08}$ with dimensions increasing from 1000 to 10000

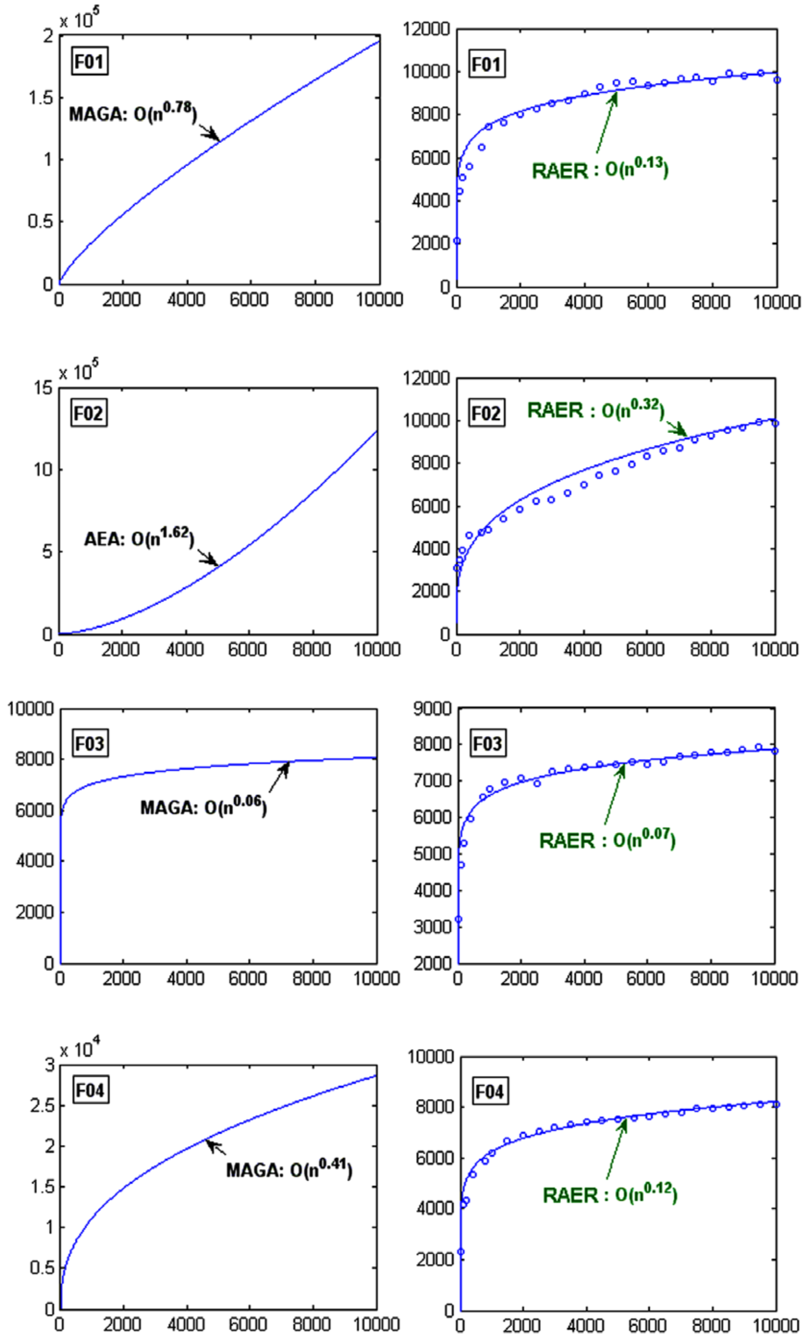

**Fig. 3.** Comparison between MAGA and RAER on $F_{01}$–$F_{08}$ with 20–10000 dimensions.

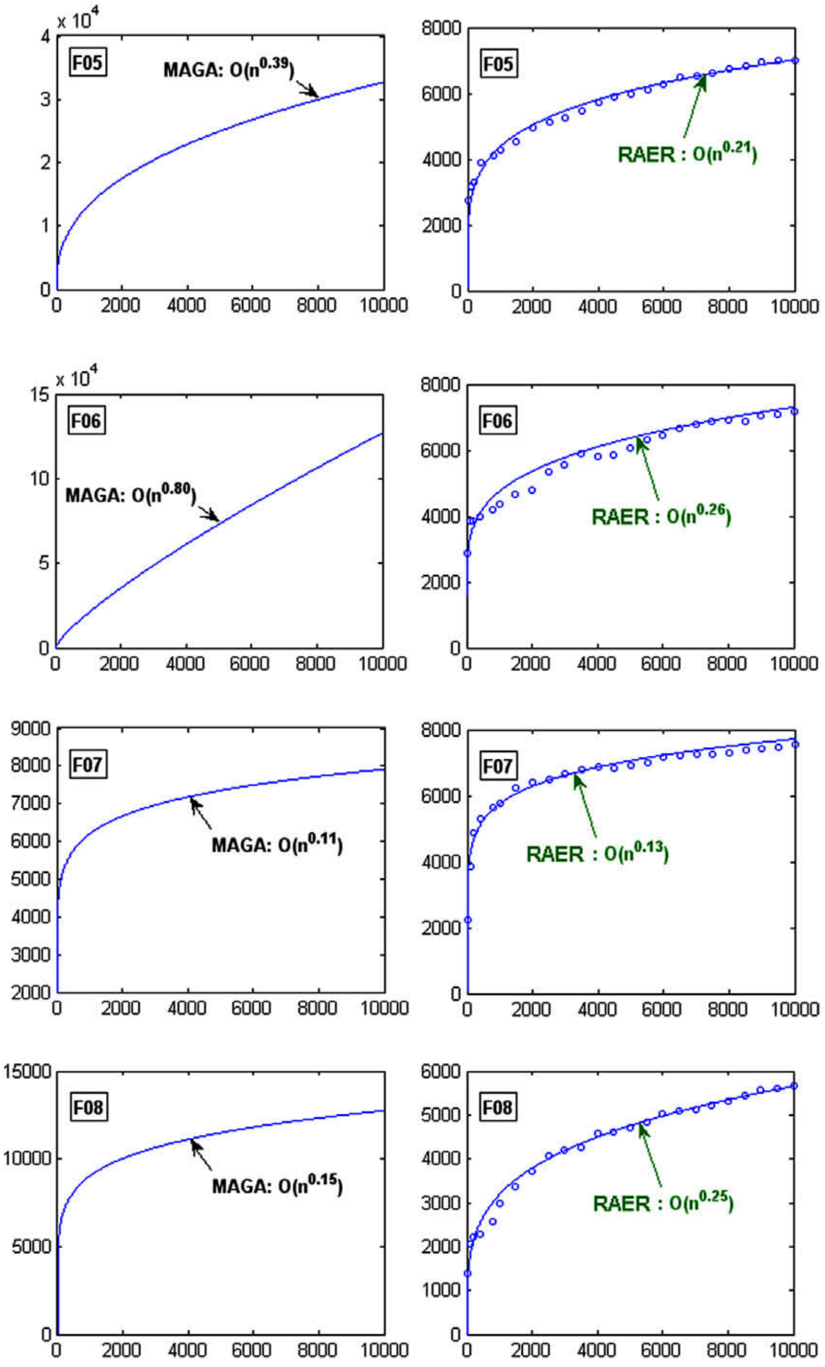

**Fig. 3** (*continued*)

in steps of 500. The termination criterions of RAER and MAGA are same as those in Section 4.3.2. The mean function evaluations needed by RAER at each step for all 8 functions are depicted in Fig. 3. Furthermore, in order to study the complexity of RAER, the data of mean function evaluation for each function is approximated by $O(n^a)$ $(0 < a < 1)$ in Fig. 3. And for more clarity, the comparisons between MAGA and RAER in the mean number of function evaluations for functions with 10000 dimensions and the derived $O(n^a)$ are shown in Table 13.

From Table 13, we can see that for all functions RAER only needs less than 10000 evaluations even when $N$ increases to 10000. And the mean function evaluations of RAER for $F_{01}$ and $F_{04}$–$F_{06}$ when $N$ equals 10000 are much less than those of MAGA. And as can be seen in Fig. 3, for all eight functions $F_{01}$–$F_{08}$, the complexities of RAER is better than $O(n)$. For $F_{01}$, $F_{04}$, $F_{05}$ and $F_{06}$, the complexities of RAER for other functions are significantly lower than those of MAGA. For $F_{03}$, $F_{07}$, the

**Table 13**
Comparison between MAGA and RAER in the mean number of function evaluations on $F_{01}$–$F_{08}$ with $N = 10000$ and derived $O(n^a)$.

| | | $F_{01}$ | $F_{02}$ | $F_{03}$ | $F_{04}$ |
|---|---|---|---|---|---|
| Mean FE ($N = 10000$) | MAGA | 195292 | 14315 | 7961 | 28815 |
| | RAER | **9615** | **9879** | **7842** | **8137** |
| $O(n^a)$ | MAGA | $O(n^{0.78})$ | – | $O(n^{0.06})$ | $O(n^{0.41})$ |
| | RAER | $O(n^{0.13})$ | $O(n^{0.32})$ | $O(n^{0.07})$ | $O(n^{0.12})$ |
| | | $F_{05}$ | $F_{06}$ | $F_{07}$ | $F_{08}$ |
| Mean FE ($N = 10000$) | MAGA | 27645 | 121370 | 7784 | 12233 |
| | RAER | **6983** | **7179** | **7569** | **5675** |
| $O(n^a)$ | MAGA | $O(n^{0.39})$ | $O(n^{0.80})$ | $O(n^{0.11})$ | $O(n^{0.15})$ |
| | RAER | $O(n^{0.21})$ | $O(n^{0.26})$ | $O(n^{0.13})$ | $O(n^{0.25})$ |

**Table 14**
Comparison between NHGA, ARSAGA, LX-M and RAER on $F_{09}$–$F_{12}$ when $N = 20$–100.

| $N$ | $F_{09}$ | | $F_{10}$ | | $F_{11}$ | | $F_{12}$ | |
|---|---|---|---|---|---|---|---|---|
| | NHGA | RAER ($\times 10^{-6}$) | ARSAGA ($\times 10^{-3}$) | RAER ($\times 10^{-3}$) | ARSAGA | RAER ($\times 10^{-3}$) | LX-M ($\times 10^{-7}$) | RAER |
| 20 | – | 3219 | – | 1972 | – | 7132 | – | 34861 |
| | | (1.956) | | (2.575) | | (2.070) | | ($1.256 \times 10^{-2}$) |
| 30 | – | 4591 | 15406 | 2027 | 254541 | 19087 | 101331 | 89349 |
| | | (2.179) | (4.73) | (2.348) | ($\times 10^{-3}$) | (2.679) | (7.05) | ($9.365 \times 10^{-7}$) |
| 50 | 84327 | 8381 | – | 2292 | – | 21209 | – | 125685 |
| | ($\times 10^{-5}$) | (0.798) | | (2.420) | | (1.313) | | ($2.687 \times 10^{-2}$) |
| 100 | 141430 | 22401 | – | 3163 | – | 31103 | – | 175923 |
| | ($\times 10^{-3}$) | (1.052) | | (1.757) | | (2.104) | | ($3.257 \times 10^{-2}$) |

complexities of RAER and MAGA are very approximate. Only for $F_{08}$ the complexity of RAER is higher than that of MAGA but still as low as $O(n^{0.25})$. For $F_{02}$ the number of function evaluations of MAGA does not change with the dimension obviously, so Fig. 3 depicts the approximation line of AEA for $F_{02}$, and we can see that the complexity of RAER for $F_{02}$ is $O(n^{0.32})$ that is significantly lower than $O(n^{1.62})$ of AEA.

### 4.3.4. *Comparative experiments between RAER and other algorithms on more difficult functions $F_{09}$–$F_{12}$ with 20–100 dimensions*

In this subsection, comparative experiments are taken to examine the performance of RAER on more difficult functions $F_{09}$–$F_{12}$ with dimensions ranging 20–100. $F_{09}$ is the Zakharov's function, which is often used as a benchmark function for testing the performance of optimization algorithm. $F_{10}$ is the De-Jong's function with noise, whose function value randomly changes with algorithms' searching the problem domain. $F_{11}$ and $F_{12}$ are difficult functions and often used to test the comprehensive performance of optimization algorithms, but seldom tested in references with their dimensions larger than 30.

The termination criterions of RAER are set as follows: $|f| < 1E-5$ for $F_{09}$, $|f| < 1E-2$ for $F_{10}$ and $F_{11}$, $|f| < 1E-6$ for $F_{12}$ when $N = 30$ and $|f| < 1E-1$ when $N \neq 30$. The mean number and the standard deviations of function evaluations of RAER averaged over 30 trials are shown in Table 14. Table 14 also shows the comparative results of recently proposed good algorithms of NHGA [3], ARSAGA [4] and LX-M [19].

As can be seen from Table 14 RAER outperforms other comparative algorithms for all functions $F_{09}$–$F_{12}$ with dimensions ranging 20–100. For $F_{09}$ RAER only needs thousands of function evaluations to find optimum solution with precision of E−06 when $N = 20$–50 and 22401 function evaluations when $N = 100$, while NHGA needs 141430 function evaluations to get optimum solution with precision of E−03 when $N = 100$. For De-Jong's function with noise $F_{10}$ RAER needs 2027 function evaluations to achieve solution with the same precision as ARSAGA when $N = 30$, but ARSAGA needs 15406 function evaluations. For more difficult functions $F_{11}$ and $F_{12}$, when $N = 30$, ARSAGA needs 254541 and 363979 function evaluations respectively to get optimum solutions. LX-M needs 101331 function evaluations to get optimum solution for $F_{12}$ when $N = 30$. In comparison, RAER needs 19087 and 89349 function evaluations for $F_{11}$ and $F_{12}$, respectively, which is better than ARSAGA and LX-M significantly.

## 5. Performance of RAER on engineering optimization problems

In this section, the performance of RAER is examined on two practical engineering optimization problems of a stable linear system approximation and a welded beam design. In order to examine the search ability of RAER, for the two practical problems, RAER is applied to search the optimum solutions in both fixed problem domains and dynamically expanded problem domain with the search space expansion scheme in Ref. [30]. We refer the readers to gain more details about the search space expansion scheme to [13,30]. For the experiments with fixed problem domains, the parameters are set as: $M = N = 10$ in $L_{M\times N}$, $P_{nd} = 0.75$, $P_{nd}^5 = 0.5$, $P_s = 0.2$, $m = n = 3$ in $\mathrm{RL}_{m\times n}$, $\mathrm{RP}_{nd} = 0.2$, $\mathrm{RP}_{nd}^5 = 0.9$, $\mathrm{RP}_s = 0.1$, MGen $= 6$, $U^1$ (11,4) and $U^2$

(11,4) are chosen to collaborate. For the experiments with dynamically expanded problem domains, the parameters are set as: $M = N = 10$ in $L_{M\times N}$, $P_{nd} = 0.9$, $P^5_{nd} = 0.5$, $P_s = 0.1$, $m = n = 3$ in $\mathrm{RL}_{m\times n}$, $\mathrm{RP}_{nd} = 0.9$, $\mathrm{RP}^5_{nd} = 0.9$, $\mathrm{RP}_s = 0.05$, $\mathrm{MGen} = 8$, $U^1$ (5,4) is chosen, and search space expansion factor is fixed as 2, and search space checking period $N_E$ is fixed as 8.

## 5.1. Optimal approximation of a stable linear system

This system is quoted from [14,30], its transfer function is as follows:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{k_{r1}k_d(\tau_{od}s+1)e^{-\theta_d s}}{(\tau_r s+1)(\tau_1 s+1)(\tau_2 s+1) - k_{r2}k_d(\tau_{od}s+1)e^{-\theta_d s}}, \tag{13}$$

where $k_{r1} = 0.258$, $k_{r2} = 0.281$, $k_d = 1.4494$, $\theta_d = 0.2912$, $\tau_r = 0.3683$, $\tau_1 = 1.9624$, $\tau_2 = 0.43256$. It is desired to find the second-order models:

$$H_2(s) = \frac{k_{2,p}(s+\tau_{2,z})}{a_{2,0}+a_{2,1}s+s^2} \times e^{-\tau_{2,d}s}, \tag{14}$$

with the 51 frequency values, $\omega_i = 10^{-2+0.1i}$, $i = 0,1\ldots,N = 50$, such that the performance index given by following formula is minimized:

$$J = \sum_{i=0}^{N} |G(j\omega_i) - H_2(j\omega_i)|^2, \tag{15}$$

while it is subject to the constraint of $H_2(0) = G(0)$, so under this requirement, the unknown parameter is simply related to others by the relation:

$$a_{2,0} = \frac{k_{2,p}(1-k_{r2}k_d)}{k_{r1}k_d}\tau_{2,z}. \tag{16}$$

Hence the parameters to be determined are $a_{2,1}$, $k_{2,p}$, $\tau_{2,z}$, $\tau_{2,d}$. Due to the fact that the original system is stable, each parameter lies in the interval $[0,+\infty)$.

In comparison with AIRA [14], we first apply RAER to search the optimal vector of parameters $H^4 = [a_{2,1}k_{2,p}\tau_{2,z}\tau_{2,d}]$ in fixed problem domains as $[0,10]^4$, $[0,50]^4$, $[0,100]^4$, $[0,150]^4$. Then RAER is applied to search $H^4$ in dynamical problem domain with the search space expansion scheme in Ref. [30], where the initial search space for the parameter vector $H^4$ is set as $[0,0.1]^4$.

For different problem domains, the termination criterions of RAER are set as shown in Table 15, where FPDS, DEPD, MGN and FE stand for fixed problem domains, dynamically expanded problem domain, maximum generation number and function evaluations, respectively. Results averaged over 30 trials are shown in Table 16, and comparative results of different algorithms are shown in Table 17.

In Table 16, the best solutions and typical solutions of each specified fixed problem domain, the best solutions and worst solutions of dynamically expanded problem domain, are listed, where $J_{min}$ is the index value of best solutions. When contrasting values of the optimal parameter $\tau_{2,z}$ gained by different algorithms in Tables 16 and 17, we can see that RAER can find the proper best and typical solutions belonging to the different fixed problem domains, which shows RAER can search the problem domain more thoroughly than other algorithms and is a significant virtue in comparison with other algorithms.

As can be seen from Table 17, RAER can achieve better solutions in different fixed problem domains. Likewise when searching in the dynamically expanded problem domain, RAER also can achieve better solutions than DEA [30], AIRA [14] and MAGA [13], furthermore, the function evaluations of RAER is 16805, which is better than other algorithms. In Fig. 4, the best and worst solutions obtained by RAER in dynamically expanded problem domain are demonstrated for frequency response in comparison of the original complex system. From Fig. 4 we cannot distinguish them even for the worst solution.

**Table 15**
Termination criterions for system approximation experiments.

| Domains | Termination criterions |
|---|---|
| FPDS | |
| $[0,10]^4$ | MGN $\leqslant$ 300 or $J <$ 8E−5 |
| $[0,50]^4$ | MGN $\leqslant$ 300 or $J <$ 1E−5 |
| $[0,100]^4$ | MGN $\leqslant$ 300 or $J <$ 6.5E−6 |
| $[0,150]^4$ | MGN $\leqslant$ 300 or $J <$ 6.5E−6 |
| DEPD | |
| $[0,0.1]^4$ | FE > 18000 or $J <$ 6.5E−6 |

**Table 16**
Results of RAER for approximation of the stable linear system.

| Domains | $H^4$ | $k_{2,p}$ | $\tau_{2,z}$ | $\tau_{2,d}$ | $a_{2,1}$ | $J_{min}$ (*Std*) |
|---|---|---|---|---|---|---|
| FPDS | | | | | | |
| $[0,10]^4$ | Best | 0.021229726 | **6.269420661** | 0.460862234 | 1.241596135 | 4.4087E−05 |
| | Typical | 0.014482588 | **9.709461718** | 0.439455492 | 1.308269354 | |
| $[0,50]^4$ | Best | 0.003929332 | **34.83656140** | 0.348223864 | 1.272989762 | 6.7633E−06 |
| | Typical | 0.006752900 | **20.08049145** | 0.362592639 | 1.261696197 | |
| | | 0.003705140 | **36.91193331** | 0.343563870 | 1.271429771 | |
| | | 0.002739416 | **49.24709055** | 0.332892915 | 1.255846022 | |
| $[0,100]^4$ | Best | 0.001363110 | **99.90386088** | 0.327554018 | 1.265931435 | 5.8084E−06 |
| | Typical | 0.002068533 | **65.69535780** | 0.331692854 | 1.263796676 | |
| $[0,150]^4$ | Best | 0.000918607 | **148.3250622** | 0.325052004 | 1.266667868 | 5.7398E−06 |
| | Typical | 0.001201192 | **113.6601367** | 0.327956415 | 1.269246385 | |
| | | 0.000971244 | **139.6834740** | 0.322292049 | 1.262009375 | |
| DEPD | Best | 0.002213217 | **61.57862161** | 0.332936985 | 1.267172384 | 5.9744E−06 |
| | Worst | 0.015734784 | **8.519882399** | 0.423444103 | 1.248896066 | (3.8709E−06) |

**Table 17**
Comparative best results of different algorithms for approximation of the stable linear system.

| Domains | Methods | $k_{2,p}$ | $\tau_{2,z}$ | $\tau_{2,d}$ | $a_{2,1}$ | $J$ | FE |
|---|---|---|---|---|---|---|---|
| FPDS | | | | | | | |
| $[0,10]^4$ | DEA | 0.0266114 | **6.10646821** | 0.4894422 | 1.65105264 | 1.4308E−04 | – |
| | AIRA | 0.021430231 | **6.130164886** | 0.454259829 | 1.226172374 | 4.4889E−05 | |
| | MAGA | – | – | – | – | – | |
| | RAER | 0.021229726 | **6.269420661** | 0.460862234 | 1.241596135 | 4.4087E−05 | |
| $[0,50]^4$ | DEA | 0.00495562 | **35.0505313** | 0.39077937 | 1.7606862 | 6.2586E−05 | – |
| | AIRA | 0.006473125 | **20.835707811** | 0.360175108 | 1.254970252 | 8.2269E−06 | |
| | MAGA | – | – | – | – | – | |
| | RAER | 0.003929332 | **34.83656140** | 0.348223864 | 1.272989762 | 6.7633E−06 | |
| $[0,100]^4$ | DEA | 0.00415886 | **65.9114728** | 0.5907097 | 2.714033300 | 1.0021E−03 | – |
| | AIRA | 0.00337488 | **39.802784056** | 0.332274274 | 1.250285692 | 8.5174E−06 | |
| | MAGA | – | – | – | – | | |
| | RAER | 0.00136311 | **99.90386088** | 0.327554018 | 1.265931435 | 5.8084E−06 | |
| $[0,150]^4$ | DEA | 0.0025538 | **68.1706405** | 0.3787929 | 1.7606862 | 6.1129E−05 | – |
| | AIRA | 0.006163363 | **21.47851000** | 0.357316587 | 1.256812898 | 7.7369E−06 | |
| | MAGA | – | – | – | – | – | |
| | RAER | 0.000918607 | **148.3250622** | 0.325052004 | 1.266667868 | 5.7398E−06 | |
| DEPD | DEA | 0.0025538 | **68.1706405** | 0.3787929 | 1.7606862 | 6.1129E−05 | 19800 |
| | AIRA | 0.004717881 | **28.822747965** | 0.351042212 | 1.264931127 | 6.5526E−06 | 18015 |
| | MAGA | 0.0172520 | **7.7668024** | 0.4346119 | 1.2483717 | 2.7180E−05 | 19735 |
| | RAER | 0.002213217 | **61.57862161** | 0.332936985 | 1.267172384 | 5.9743E−06 | 16805 |

## 5.2. *Minimum cost of a welded beam*

This engineering problem is taken from [4,31], the initial configuration of this optimization problem is shown in Fig. 5. A beam denoted as ***A*** and made by low-carbon steel (C-1010) is to be welded to a rigid support member ***B***. The length ***L*** of the welded beam is fixed as 14 in. and it is designed to support a load ***F*** of 6000 lb. The width and height of this solid beam is denoted as $b$ and $t$, respectively. Besides, the length of welding is $l$ and the width of welding is $h$. Hence, the design variable vector can be denoted as $x^4 = [h,l,t,b]$. The objectives of design are to minimize the cost of the system, which can be formulated as follows:

$$\begin{aligned}
&\text{Minimize} \quad f_1(\vec{x}) = 1.10471h^2l + 0.04811tb(14.0+l) \\
&\text{s.t.}: \quad \phi_1(\vec{x}) \equiv \bar{\tau} - \tau(\vec{x}) \geqslant 0, && (17)\\
&\qquad\quad \phi_2(\vec{x}) \equiv \bar{\sigma} - \sigma(\vec{x}) \geqslant 0, && (18)\\
&\qquad\quad \phi_3(\vec{x}) \equiv b - h \geqslant 0, && (19)\\
&\qquad\quad \phi_4(\vec{x}) \equiv P_c(\vec{x}) - P \geqslant 0. && (20)
\end{aligned}$$

where $\vec{x} = (h,l,t,b)$, $\delta(\vec{x}) = 2.1952/(t^3b)$, $\sigma(\vec{x}) = \frac{504000}{t^2b}$, $P_c(\vec{x}) = 64746.022(1-0.028236t)tb^3$, $\tau' = \frac{P}{\sqrt{2}hl}$, $\tau'' = \frac{P(L+0.5l)\sqrt{0.25(l^2+(h+t)^2)}}{2\{0.707hl(l^2/12+0.25(h+t)^2)\}}$.
Ref. [4] used penalty functions to transform this single objective constrained problem to the following optimization problem formulated as:

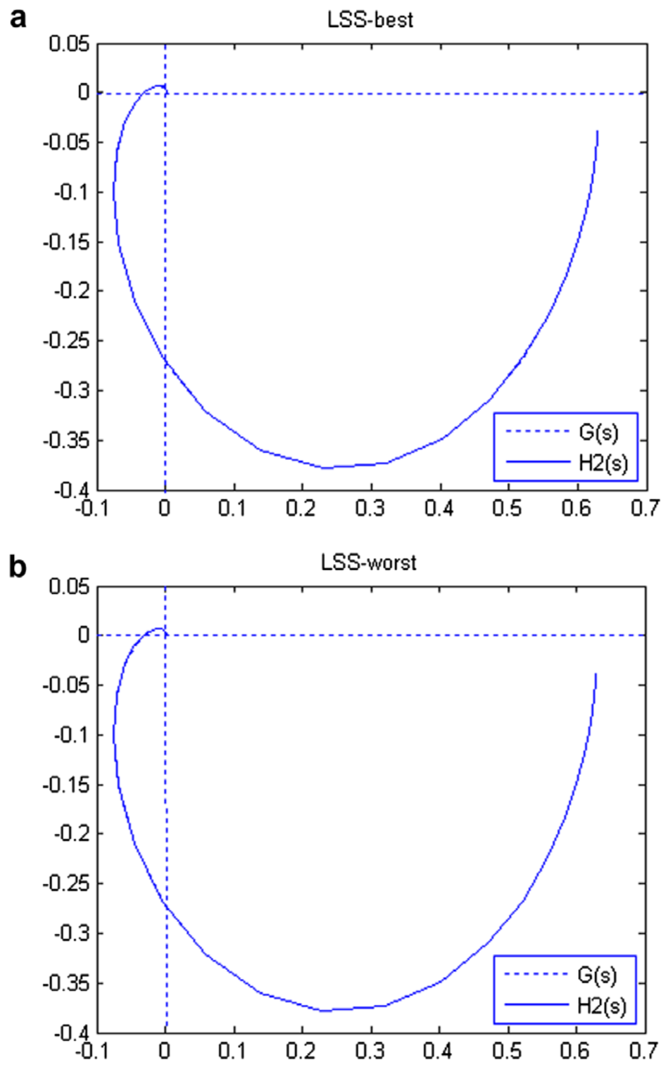

**Fig. 4.** The frequency responses of the original system $G(s)$ and the approximate models $H_2(s)$ obtained by RAER in dynamically expanded problem domain (DEPD); (a) best solution and (b) worst solution.

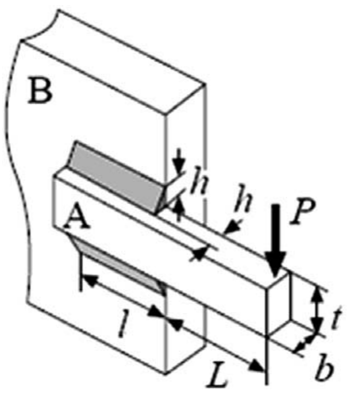

**Fig. 5.** Initial configuration of a welded beam.

$$f = (1 + C_1)h^2 l + C_2 tb(L + l) + \sum_{i=1}^{m} P_i, \tag{21}$$

where $C_1$ is the cost of per volume of the welded material, $C_2$ is the cost per volume of the bar stock, and $m$ is the total number of structural constraints. The $i$th penalty function $P_i$ is defined as

$$P_i = \begin{cases} 0 & \text{when the } i\text{th constraint is met,} \\ X & \text{when the } i\text{th constraint is violated,} \end{cases} \tag{22}$$

where $X$ is a big positive constant. Then this problem is also subjected to the following constraints [31]:

$$\phi_5(\vec{x}) \equiv l \geqslant 0, \tag{23}$$
$$\phi_6(\vec{x}) \equiv t \geqslant 0, \tag{24}$$
$$\phi_7(\vec{x}) \equiv h - 0.125 \geqslant 0, \tag{25}$$
$$\phi_8(\vec{x}) \equiv 0.25 - \delta(\vec{x}) \geqslant 0. \tag{26}$$

As the same as Section 5.1, RAER is also applied to search $x^4 = [h, l, t, b]$ in both fixed and dynamically expanded problem domains. For clarity, the search spaces of the design variables and the values of some specifications are listed in Table 18, where FPD and DEPD stand for fixed problem domains and dynamically expanded problem domain specifically.

We perform RAER in fixed problem domains and dynamically expanded problem domain under different termination criterions and corresponding results averaged over 20 trials are listed in Table 19, where TC stands for termination criterion, Std stands for standard deviations and FE stands for function evaluations. The best solutions gained by RAER and other algorithms are listed in Table 20 for comparison. The convergences of $f$ with respect to the number of generations of the best and the worst solutions gained in FPD ($f < 2.383$) and DEPD ($f < 2.3812$) are shown in Figs. 6a and 7a, respectively. The convergences of the parameter values of the best solutions gained in FPD ($f < 2.383$) and DEPD ($f < 2.3812$) are shown in Figs. 6b and 7b, respectively.

As can be seen from Table 19, the proposed algorithm RAER can achieve good solutions with rather low function evaluations. The Std is very low for each different case, especially for the case in DEPD with the TC of $f < 2.3812$, the Std is as low as 8.22E−06. So RAER shows very excellent performance of reliability and stability no matter in FPD or DEPD even though the object is an engineering optimization problem with many constraints.

In Table 20, the best solution gained by algorithm GP violates the $\phi_1(\vec{x})$, the corresponding $\tau(\vec{x}) = 13600.306$. The best solution gained by algorithm SIMPLEX violates the $\phi_1(\vec{x})$ and $\phi_2(\vec{x})$, the corresponding $\tau(\vec{x}) = 13601.079, \sigma(\vec{x}) = 30002.365$. The best solution gained by algorithm RANDOM violates the $\phi_1(\vec{x})$, the corresponding $\tau(\vec{x}) = 13600.278$. The best solution gained by algorithm GEBENOPT obviously violates the $\phi_3(\vec{x})$. The best solution gained by algorithm ARSAGA violates the $\phi_1(\vec{x})$ and $\phi_4(\vec{x})$, the corresponding $\tau(\vec{x}) = 31110.786, P_c(\vec{x}) = 5997.696$. Only the best solutions gained by APPROX, DAVID, CGA and the proposed RAER do not violate all the eight strict constraints.

**Table 18**
Search spaces of design variables and values of specifications.

| Items | Configuration | |
|---|---|---|
| | FPD | DEPD |
| Search space | | |
| $h$ | [0.125 20.0] | [0.125 0.5] |
| $l$ | [0.1 20.0] | [0.1 0.5] |
| $t$ | [0.1 20.0] | [0.1 0.5] |
| $b$ | [0.1 20.0] | [0.1 0.5] |
| $C_1$ | 0.10471 (\$/in$^3$) | |
| $C_2$ | 0.04811 (\$/in$^3$) | |
| $\bar{\tau}$ | 13600 psi | |
| $\bar{\sigma}$ | 30000 psi | |

**Table 19**
Results of REAR in different search areas under gradually stricter termination criterions.

| Methods | TC | Mean cost | | Std | FE |
|---|---|---|---|---|---|
| | | Best | Worst | | |
| FPD | $f < 2.385$ | 2.38294 | 2.38498 | 0.00064 | 22055 |
| | $f < 2.383$ | 2.38155 | 2.38297 | 0.00034 | 28897 |
| DEPD | $f < 2.385$ | 2.38331 | 2.3850 | 0.00053 | 11262 |
| | $f < 2.383$ | 2.38226 | 2.3830 | 0.00024 | 12034 |
| | $f < 2.3812$ | 2.38117 | 2.38120 | 8.22E−06 | 18467 |

**Table 20**
Comparison of best results gained by different algorithms.

| Algorithms | *h* | *l* | *t* | *b* | *f* |
|---|---|---|---|---|---|
| RAER | | | | | |
| DEPD | **0.244364** | **6.218168** | **8.292168** | **0.244365** | **2.38117** |
| FPD | **0.244267** | **6.220117** | **8.293960** | **0.244360** | **2.38155** |
| APPROX [27] | 0.2444 | 6.2189 | 8.2915 | 0.2444 | 2.38154 |
| DAVID [27] | 0.2434 | 6.2552 | 8.2915 | 0.2444 | 2.38411 |
| GP [27] | 0.2455 | 6.1960 | 8.2730 | 0.2455 | 2.39 |
| SIMPLEX [27] | 0.2792 | 5.6256 | 7.7512 | 0.2796 | 2.53 |
| RANDOM [27] | 0.4575 | 4.7313 | 5.0853 | 0.6600 | 4.12 |
| GEBENOPT [27] | 0.2489 | 6.1097 | 8.2484 | 0.2485 | 2.40 |
| CGA [27] | 0.2489 | 6.1730 | 8.1789 | 0.2245 | 2.43 |
| ARSAGA [4] | 0.2231 | 1.5815 | 12.8468 | 0.2245 | 2.25 |

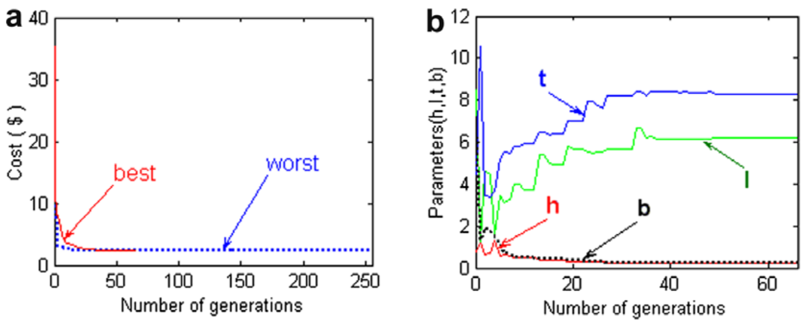

**Fig. 6.** (a) and (b) are the convergences of *f*(Cost) and the parameter values with respect to the number of generations for RAER in fixed problem domain (FPD), respectively.

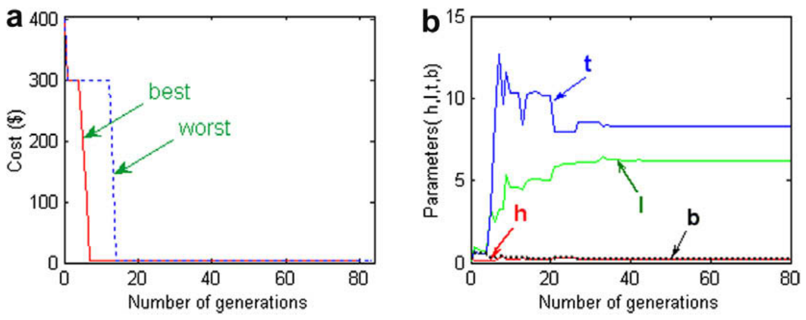

**Fig. 7.** (a) and (b) are the convergences of *f*(Cost) and the parameter values with respect to the number of generations for RAER in dynamically expanded problem domain (DEPD), respectively.

From Tables 19 and 20 it shows that the proposed algorithm RAER can achieve better solutions than other algorithms in FPD and DEPD with functions evaluations of 28897 and 18467 under strictest termination criterion. By comparison, ARSAGA needs 26466 function evaluations to achieve its best solutions violating two constraints. Furthermore, Figs. 6 and 7 show that the proposed algorithm RAER has fast convergence speed to the desired solution no matter in FPD or DEPD.

## 6. Conclusions

Based on the ERA multiagent modeling pattern, a multiagent evolutionary optimization algorithm, named RAER, has been proposed. RAER integrates a novel roulette inversion operator (RIO) proposed to conquer the irrationality of the inversion operator (IO) designed by John Holland when used for real code stochastic algorithms. Theoretical analysis and experiments show that the RIO operator bears better functioning than the IO operator. 12 benchmark functions are tested to examine the scalability of RAER along the problem dimension and results show that RAER can obtain high quality solutions with lower computation cost than other comparative algorithms. The computation complexities of RAER gained by approximation of function evaluations with dimensions of 20–10000 by $O(n^a)$ $(0 < a < 1)$ show that in most cases RAER's complexities are lower than other comparative algorithms. The experiments on more difficult $F_{09}$–$F_{12}$ with dimensions of 20–100 also show that RAER outperforms other comparative algorithms recently proposed.

The applicability of RAER is examined by applying RAER to search optimum solutions in both fixed problem domains and dynamically expanded problem domains for two engineering optimization problems. The experiments for the optimal approximation of a stable linear system show that RAER has a better performance, especially search in fixed problem domains, RAER can find the typical and proper optimal solutions belonging to different fixed problem domains, which is a significant good property in comparison with other algorithms. The experiments for a welded beam design constrained optimization problem show that RAER, under strict constrains, still has good search ability and fast convergence speed, which is better than other comparative algorithms. Hence, the proposed algorithm RAER not only has better performance in function optimization problems but also has good applicability in engineering optimization problems.

## Acknowledgements

The work was supported by the National Natural Science Foundation of China under Grants 70631003 and 70771037, the "863" Project under Grant 2006AA04A126, and the Zhejiang Provincial Natural Science Foundation of China under Grant Y6080215. The authors also wish to thank the anonymous expert reviewers for their valuable comments and helpful suggestions.

## References

[1] J.H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, MIT Press, Cambridge, 1992.
[2] A. Isaacs, T. Ray, W. Simth, A hybrid evolutionary algorithm with simplex local search, in: 2007 IEEE Congress on Evolutionary Computation (CEC 2007), Singapore, 2007, pp. 1701–1708.
[3] Lingyun Wei, Mei Zhao, A niche hybrid genetic algorithm for global optimization of continuous multimodal functions, Applied Mathematics and Computation 160 (2005) 649–661.
[4] Shun Fa Hwang, Rong Song He, A hybrid real-parameter genetic algorithm for function optimization, Advanced Engineering Informatics 20 (2006) 07–21.
[5] Y.W. Leung, Y. Wang, An orthogonal genetic algorithm with quantization for global numerical optimization, IEEE Transactions on Evolutionary Computation 5 (2001) 41–53.
[6] Z.J. Pan, L.S. Kang, "An adaptive evolutionary algorithms for numerical optimization" in Simulated Evolution and Learning, Lecture Notes in Artificial Intelligence, Springer-Verlag, Berlin, Germany, 1997.
[7] Ajith Abraham, Crina Grosan, Vitorino Ramos (Eds.), Swarm Intelligence in Data Mining, Springer-Verlag, New York, 2006.
[8] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Networks, Piscataway, NJ, 1995, pp. 1942–1948.
[9] E. Bonabeau, M. Dorigo, G. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems, Oxford University Press, Oxford, 1999.
[10] Swarm Development Group, Swarm simulation system. <http://www.swarm.org/index.php?title=Main_Page#Swarm>.
[11] J. Liu, Y.Y. Tang, Adaptive segmentation with distributed behavior-based agents, IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (6) (1999) 544–551.
[12] J. Han, Data mining techniques. <ftp://ftp.fas.sfu.ca/pub/cs/han/kdd/sigmod96_tuto_des.ps>.
[13] W.C. Zhong, J. Liu, M.Z. Xue, et al, A multiagent genetic algorithm for global numerical optimization, IEEE Transactions on Systems, Man, and Cybernetics – Part B 34 (2) (2004) 1128–1141.
[14] Maoguo Gong, Haifeng Du, Licheng Jiao, Optimal approximation of linear systems by artificial immune response, Science in China: Series F Information Science 49 (1) (2006) 63–79.
[15] Y. Wang, K.T. Fang, A note on uniform distribution and experimental design, KEXUE TONGBAO 26 (6) (1981) 485–489 (in Chinese).
[16] K.T. Fang, Uniform Design and Design Tables, Science, Beijing, China, 1994 (in Chinese).
[17] K.T. Fang, Y. Wang, Number-Theoretic Methods in Statistics, Chapman & Hall, London, UK, 1994.
[18] M. Iosifescu, Finite Markov Processes and Their Applications, Wiley, Chichester, 1980.
[19] Kusum Deep, Manoj Thakur, A new crossover operator for real coded genetic algorithms, Applied Mathematics and Computation 188 (1) (2006) 895–911.
[20] Mingzhi Xue, Weicai Zhong, Jing Liu, et al, Orthogonal multi-agent genetic algorithm and its performance analysis, Control and Decision 19 (3) (2004) 290–294.
[21] García-Villoria Alberto, Pastor Rafael, Introducing dynamic diversity into a discrete particle swarm optimization, Computers and Operations Research 36 (3) (2009) 951–966.
[22] W. Zhang, Y. Liu, M. Clerc, An adaptive PSO algorithm for reactive power optimization, in: Sixth International Conference on Advances in Power Control, Operation and Management, Hong Kong, 2003.
[23] D. Whitley, The GENITOR algorithm and selection pressure: why rank-based allocation of reproductive trials is best, in: J.D. Schaffer (Ed.), Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, San Mateo, 1989, pp. 116–121.
[24] Lozano Manuel, Herrera Francisco, Ramon Cano Jose, Replacement strategies to preserve useful diversity in steady-state genetic algorithms, Information Sciences 178 (23) (2009) 4421–4433.
[25] Yiu Wing Leung, Yu Ping Wang, Multiobjective programming using uniform design and genetic algorithm, IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews 30 (3) (2000) 293–304.
[26] Ishibuchi Hisao, Murata Tadahiko, A multi-objective genetic local search algorithm and its application to flowshop scheduling, IEEE Transaction on System,Man, and Cybernetics – Part C: Applications and Reviews 28 (3) (1998) 392–403.
[27] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Transactions on Evolutionary Computation 3 (1999) 82–102.
[28] H. Mühlenbein, D. Schlierkamp-Vose, Predictive models for the breeder genetic algorithm, Evolutionary Computation 1 (1) (1993) 25–49.
[29] Haifeng Du, Maoguo Gong, Licheng Jiao, et al, A novel algorithm of artificial immune system for high-dimensional function numerical optimization, Progress in Natural Science 15 (5) (2005) 463–471.
[30] S.L. Cheng, C.Y. Huang, Optimal approximation of linear systems by a differential evolution algorithm, IEEE Transactions on Systems, Man, and Cybernetics – Part A 31 (6) (2001) 698–707.
[31] G.V. Reklaitis, A. Ravindran, K.M. Ragsdell, Engineering Optimization Methods and Applications, Wiley, New York, 1983.