Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 420652, 9 pages
http://dx.doi.org/10.1155/2014/420652

Research Article

An Improved Real-Coded Population-Based Extremal Optimization Method for Continuous Unconstrained Optimization Problems

Guo-Qiang Zeng, Kang-Di Lu, Jie Chen, Zheng-Jiang Zhang, Yu-Xing Dai, Wen-Wen Peng, and Chong-Wei Zheng

Department of Electrical and Electronic Engineering, Wenzhou University, Wenzhou 325035, China

Correspondence should be addressed to Guo-Qiang Zeng; zeng.guoqiang5@gmail.com

Received 6 January 2014; Accepted 1 April 2014; Published 8 May 2014

Academic Editor: Manyu Xiao

As a novel evolutionary optimization method, extremal optimization (EO) has been successfully applied to a variety of combinatorial optimization problems. However, the applications of EO in continuous optimization problems are relatively rare. This paper proposes an improved real-coded population-based EO method (IRPEO) for continuous unconstrained optimization problems. The key operations of IRPEO include generation of real-coded random initial population, evaluation of individual and population fitness, selection of bad elements according to power-law probability distribution, generation of new population based on uniform random mutation, and updating the population by accepting the new population unconditionally. The experimental results on 10 benchmark test functions with the dimension $N = 30$ have shown that IRPEO is competitive or even better than the recently reported various genetic algorithm (GA) versions with different mutation operations in terms of simplicity, effectiveness, and efficiency. Furthermore, the superiority of IRPEO to other evolutionary algorithms such as original population-based EO, particle swarm optimization (PSO), and the hybrid PSO-EO is also demonstrated by the experimental results on some benchmark functions.

1. Introduction

It has been widely recognized that a variety of real-world complex engineering optimization problems can be formulated as continuous unconstrained optimization problems [1]. On the other hand, these benchmark functions of unconstrained optimization problems have been often used to evaluate the performances of various evolutionary optimization algorithms [2], for example, genetic algorithm (GA) and its modified versions. This paper focuses on another novel evolutionary algorithm called extremal optimization (EO) for continuous unconstrained optimization problems.

Originally inspired by far-from-equilibrium dynamics of self-organized criticality (SOC) [3, 4], EO provides a novel insight into optimization domain because it merely selects against the bad instead of favoring the good randomly or according to a power-law distribution [5, 6]. From the perspectives of evolutionary computation, EO is much simpler than other popular evolutionary algorithms, such as genetic algorithms (GA), because it has only selection and mutation operations and less adjustable parameters [7, 8]. As a consequence, the basic EO algorithm and its modified versions have been successfully applied to a variety of benchmark and real-world engineering optimization problems, such as graph partitioning [9], graph coloring [10], travelling salesman problem [11, 12], maximum satisfiability (MAX-SAT) problem [13, 14], and steel production scheduling [15]. The more comprehensive introduction concerning EO is referred to in the surveys [16, 17].

However, the applications of EO in continuous optimization problems are relatively rare [18–23]. Sousa and Ramos [19] presented generalized EO (GEO) for continuous optimization problems, where each variable is encoded a binary bit. Furthermore, the GEO has been successfully

applied to complex heat pipe design [20]. However, the constraints have been incorporated simply into the GEO algorithm by setting a high fitness value for a minimization problem when the solution is infeasible. In [21], a modified algorithm called population-based EO (PEO) has been proposed for solving constrained optimization problems. The main advantage of PEO is the iterated optimization based on the basic EO algorithm starting from an initial population that consists of a set of individuals. Additionally, Chen et al. [22] have proposed a hybrid algorithm called PSO-EO by combining the exploration ability of particle swarm optimization (PSO) with the exploitation ability of EO. The effectiveness of PSO-EO has been demonstrated by the experimental results on 6 benchmark test functions. Another similar hybrid algorithm for continuous problem is based on improved shuffled frog-leaping algorithm and EO [23]. Following this line of PEO, this paper extends the basic idea of PEO to continuous unconstrained optimization problems and presents an improved real-coded population-based EO (IRPEO) algorithm. The key operations of IRPEO include generation of real-coded random initial population, evaluation of individual and population fitness, selection of bad elements according to power-law probability distribution, generation of new population based on uniform random mutation, and updating the population by accepting the new population unconditionally. Compared with GEO, the proposed algorithm adopts real-coded representation of the solutions and population-based iterated optimization based on a set of solutions. The superiority of IRPEO to various GA [24] algorithms with different mutation operations is demonstrated by the experimental results on 10 continuous unconstrained optimization benchmark test functions. Furthermore, the experimental results on these benchmark functions have also shown that the proposed IRPEO provides better performance than other evolutionary algorithms such as PSO, original population-based EO, and the hybrid PSO-EO algorithm [22].

The rest of this paper is organized as follows. The preliminaries on continuous optimization problems and EO are introduced in Section 2. Then, Section 3 presents the proposed IRPEO algorithm. Furthermore, the experimental results on the benchmark test functions are given to demonstrate the effectiveness of IRPEO in Section 4. Finally, the conclusion and the open issues of this paper are given in Section 5.

2. Preliminaries

2.1. Continuous Optimization Problems. An unconstrained continuous optimization problem [1] is generally defined as the following form:

$$\begin{aligned} &\min \text{(or } \max) \quad f(\mathbf{X}), \quad \mathbf{X} = (x_1, x_2, \ldots, x_n) \\ &\text{st.} \quad \mathbf{X}_{\min} \le \mathbf{X} \le \mathbf{X}_{\max}, \end{aligned} \tag{1}$$

where $f(\mathbf{X})$ is the objective function under the decision variables $\mathbf{X}, \mathbf{X}_{\min} = (X_{\min}(1), X_{\min}(2), \ldots, X_{\min}(n))$ and $\mathbf{X}_{\max} = (X_{\max}(1), X_{\max}(2), \ldots, X_{\max}(n))$ are the vector of minimum and maximum of decision variables, respectively. In other words, $X_{\min}(i) \le x_i \le X_{\max}(i), i = 1, 2, \ldots, n$, where $X_{\min}(i)$ and $X_{\max}(i)$ are the minimum and maximum of the decision variable x_i, respectively.

2.2. EO. In general, the τ-EO [5, 6] algorithm and its modified versions consist of the following basic operations, such as initialization of a random solution, evaluation of global fitness and local fitness, selection of some bad local variables based on power-law probability distribution, mutation for the selected variables and generation of a new solution, and updating the solution by accepting the new solution unconditionally [7]. The flowchart of the τ-EO for a minimization optimization problem is presented in Figure 1.

Remark 1. According to the seminal work [5, 6], the global fitness $C(S)$ of a solution S for an optimization problem with n optimized variables should be decomposed into n equivalent degrees of freedom, that is, the local fitness λ_i. Furthermore, Liu et al. [12] give consistency and equivalence conditions between global fitness and local fitness.

Remark 2. The power-law based probability selection [25] is described as follows:

$$P(k) = \frac{k^{-\tau}}{\sum_{s=1}^{n} s^{-\tau}}, \quad (1 \le k \le n), \tag{2}$$

where $P(k)$ is the probability of the kth rank variable (or element) selected from the n variables (or elements) for mutation and τ is a positive parameter controlling the power-law probability.

3. The Proposed Algorithm for Continuous Unconstrained Optimization Problems

In this section, we propose an improved real-coded population-based EO (IRPEO) algorithm for continuous unconstrained optimization problems. The basic idea behind the IRPEO is the population-based iterated optimization consisting of the following operations: generation of real-coded random initial population, evaluation of individual and population fitness, selection of bad elements according to power-law probability distribution, generation of new population based on uniform random mutation, and updating the population by accepting the new population unconditionally. The proposed algorithm is described in the following steps.

Input. They are a continuous unconstrained optimization problem and the control parameters of the IRPEO, including the control parameter τ of power-law probability distribution, the size of population SP, and maximum number of iteration $I_{\max}$.

Output. They are the best solution S_{best} and the corresponding fitness f_{best}.

Step 1. Generate an initial population $\mathbf{P_I} = \{S_1, S_2, \ldots, S_n\}$ that consists of a set of solutions, where each solution $S_i = (x_1, x_2, \ldots, x_n)$ is generated as a set of real-coded values

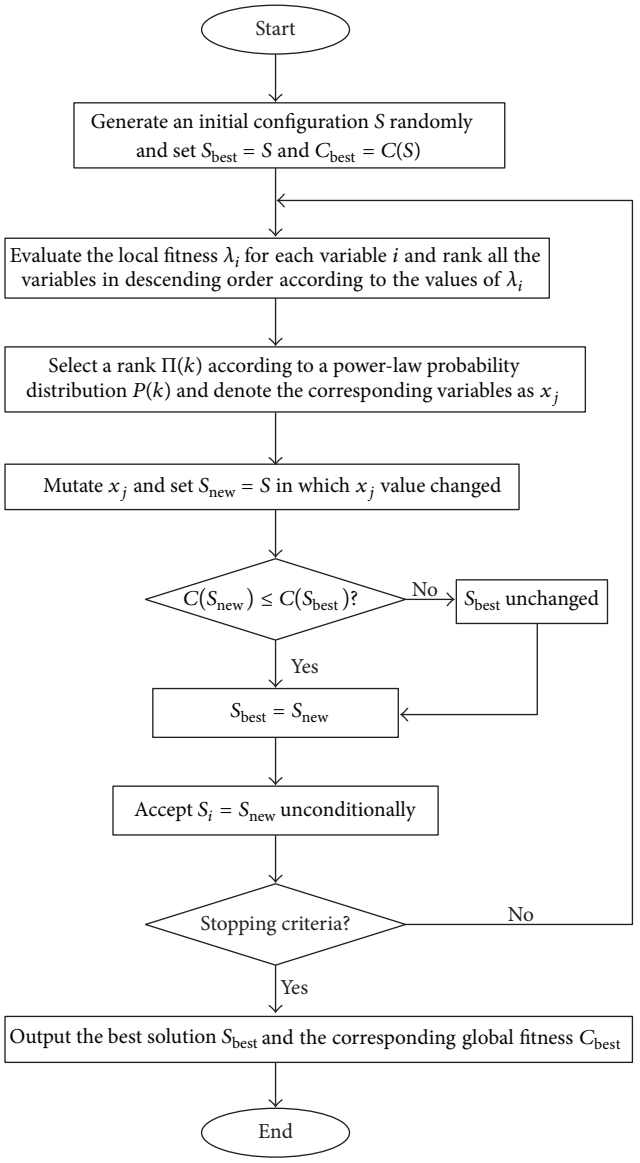


FIGURE 1: The flowchart of τ-EO algorithm for a minimization problem.

subject to the given domains randomly by the equations, and set $\mathbf{P} = \mathbf{P_I}$:

$$S_i = \mathbf{X}_{\min} + \left(\mathbf{X}_{\max} - \mathbf{X}_{\min}\right) \cdot \mathbf{R}\left(0, 1\right). \tag{3}$$

More specifically, each variable x_j in S_i is rewritten as follows:

$$S_i\left(x_j\right) = X_{\min}\left(j\right) + \left[X_{\max}\left(j\right) - X_{\min}\left(j\right)\right] \cdot \text{rand}\left(0, 1\right), \quad j = 1, 2, \ldots, n, \tag{4}$$

where rand$(0, 1)$ is a random number during 0 and 1.

Step 2. Evaluate the fitness f_i of each solution S_i according to the objective function of the continuous problem to be optimized and the fitness F of the population $\mathbf{P}$ according to the following equation:

$$F = \frac{1}{f}. \tag{5}$$

Step 3. Rank the values of $\{f_i\}$; that is, find a permutation Π_1 of the labels i such that $f_{\Pi 1(1)} \ge f_{\Pi 1(2)} \ge \cdots \ge f_{\Pi 1(SP)}$ for minimization problems ($f_{\Pi 1(1)} \le f_{\Pi 1(2)} \le \cdots \le f_{\Pi 1(SP)}$ for maximization problem) and obtain the best solution $S_{best} = S_{\Pi 1(SP)}$ and $f_{best} = f_{\Pi_1(SP)}$.

Step 4. Select a bad solution $S_{\Pi 1(i)}$ based on power-law probability distribution $P(k)$ defined as (2) and generate a new solution S_{Ni} by adopting uniform random mutation. To be more precise, generate a random number $r_{i,j}$ during 0 and 1 firstly and then the jth element of ith new solution; that is, $S_{Ni}\left(x_j\right)$ is obtained by (6). Denote the new population as $\mathbf{P_N} = \{S_{N1}, S_{N2}, \ldots, S_{Nn}\}$, where we replace $S_{\Pi 1(i)}$ by the new solution S_{Ni} and keep the other solutions unchanged:

$$S_{Ni}\left(x_j\right) = \begin{cases} \dfrac{X_{\max}\left(j\right) + X_{\min}\left(j\right)}{2} + \dfrac{X_{\max}\left(j\right) - X_{\min}\left(j\right)}{2} \\ \quad \cdot \left(\text{rand}\left(0, 1\right) - 0.5\right), \\ \qquad\qquad \text{if } P\left(j\right) < r_{i,j} \le P\left(j+1\right) \\ S_i\left(x_j\right), \qquad \text{otherwise}, \end{cases} \quad j = 1, 2, \ldots, n. \tag{6}$$

Step 5. Evaluate the fitness f_{Ni} of each solution S_{Ni} according to the objective function of the continuous problem to be optimized and rank the values of $\{f_{Ni}\}$; that is, find a permutation Π_2 of the labels i such that $f_{N\Pi 2(1)} \ge f_{N\Pi 2(2)} \ge \cdots \ge f_{N\Pi 2(SP)}$ for minimization problems (or $f_{N\Pi 2(1)} \ge f_{N\Pi 2(2)} \ge \cdots \ge f_{N\Pi 2(SP)}$ for maximization problems).

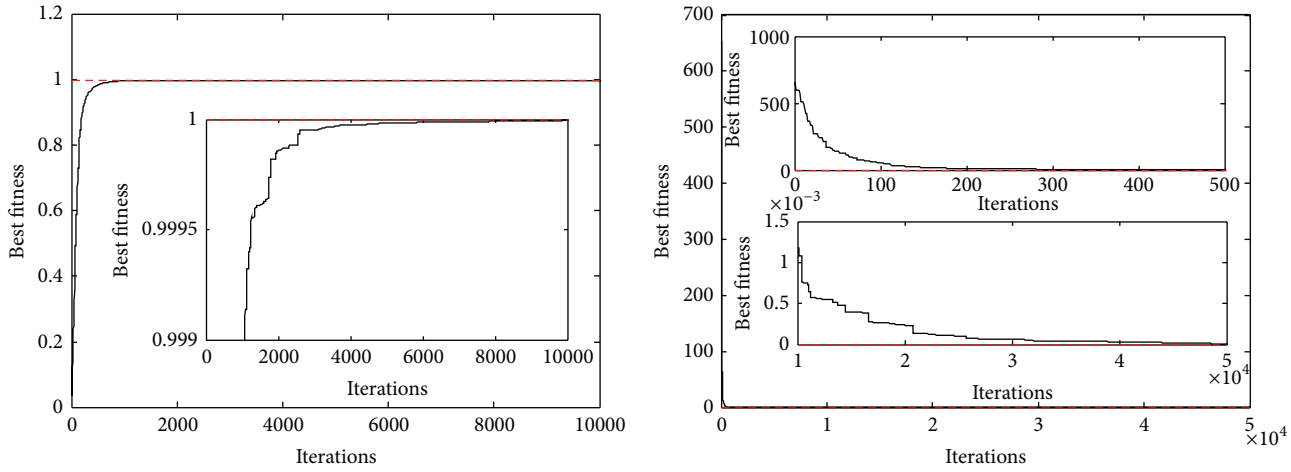
Step 6. If $f_{best} > f_{N\Pi 2(SP)}$ for minimization problems (if $f_{best} < f_{N\Pi 2(SP)}$ for maximization problems), then $S_{best} = S_{N\Pi 2(SP)}$ and $f_{best} = f_{N\Pi 2(SP)}$.

Step 7. Accept $\mathbf{P} = \mathbf{P_N}$ unconditionally.

Step 8. Repeat Step 2 to Step 7 until the stopping criteria; for example, the maximum number of iteration $I_{\max}$ is satisfied.

Step 9. Output the best solution S_{best} and the corresponding fitness f_{best}.

From the above description of the proposed algorithm, it is obvious that the parameters used in IRPEO algorithm including the size of population (SP), the maximum number of iterations ($I_{\max}$), and power-law coefficient τ play critical roles in controlling the performances of IRPEO. From the perspectives of algorithm design, IRPEO is simpler than other reported popular algorithms, for example, GA [24], PSO [22], PEO [22], and PSO-EO [22], because IRPEO has less parameters to be tuned in the practical experiments. More details concerning the parameters used in different evolutionary algorithms and the effects of these parameters SP,



(a) f_3 (b) f_4

FIGURE 2: The optimization process of the proposed EO for test functions.

$I_{\max}$, and τ on the performance of IRPEO will be discussed in the next section.

The optimization dynamics of the proposed algorithm for the benchmark test functions [24] are illustrated in Figure 2. The test function f_3 is a maximum optimization problem with the global optimum 1.000 while f_4 is a minimum optimization problem with the global optimum 0.000. Obviously, IRPEO can all find a potential optimal region fast and converge fast to the optimum (red dashed) for minimum and maximum optimization problems.

4. Experimental Results

4.1. Experimental Results. To demonstrate the superiority of the proposed IRPEO algorithm, 10 benchmark functions [24] including f_1 to f_{10} with the dimension $N = 30$ and 2 benchmark functions [22] including f_{11}, f_{12} with the dimension $N = 10, 30$, respectively, shown in Table 1 are chosen as test functions. These test functions include unimodal and multimodal functions. The performances of these algorithms for each benchmark test function are measured by the statistical results including best fitness f_B, the average fitness f_A, the worst fitness f_W, and the standard deviation (SD) under 30 independent runs. It should be noted that all the experiments have been implemented by MATLAB software on a 2.50 GHz PC with processor i5-3210 M and 2 GB RAM.

The control parameters used in IRPEO for the following experiments are set as SP = 50, $I_{\max}$ = 10000–50000, and τ = 1.04–1.06. The comparative performances of IRPEO with the recently reported GA with different mutation operations including GA-ADM, GA-RM, GA-PLM, GA-NUM, GA-MNUM, and GA-PM [24] for the above test functions are shown in Table 2. Clearly, the proposed RPEO outperforms these reported modified GA versions [24] and PEO [22] for test functions f_1–f_6 in terms of f_A and SD, so the comprehensive performance of IRPEO ranks first among these evolutionary algorithms. For the test functions f_7–f_{10}, the performance of IRPEO is only worse than GA-ADM and GA-MNUM while being better than the other four GA versions. For the same random mutation (RM) operation, the proposed IRPEO provides better performance than GA-RM [24] for all these test functions. Furthermore, the IRPEO is much simpler than these GA versions because IRPEO has only selection and mutation operations with fewer adjustable parameters to tune. In this sense, the proposed IRPEO is competitive or even better than the recently reported various GA versions with different mutation operations.

Table 3 gives the comparative performances of IRPEO with other reported evolutionary algorithms [22] including PSO-EO, PSO, PEO, and GA for the test functions f_{11} and f_{12}. It is evident that the proposed IRPEO is also superior to these algorithms in terms of f_B, f_A, f_W, and SD.

4.2. Parameters versus Performance. The parameters used in these tested evolutionary algorithms in the last subsection are shown in Table 4. It is clear that the proposed IRPEO is simpler than other reported popular algorithms, for example, GA [24], PSO [22], PEO [22], and PSO-EO [22], because IRPEO has fewer parameters to be tuned in the practical experiments.

The effects of parameters including SP, $I_{\max}$, and τ on the performance of IRPEO for test function f_2 are illustrated in Figures 3, 4, and 5, respectively. It should be noted that the performance of IRPEO is measured by the error between the statistical results (f_B, f_A, and f_W) under 10 independent runs and the optimum of the test function. Generally, the performance is improved as the values of SP and $I_{\max}$ increase, but its improvement is not distinct when these values reach to some constants. Additionally, the satisfied performance is obtained when τ ranges 1.02 to 1.06. In fact, the effects of these parameters on the performance of IRPEO for other test functions are obtained by similar analysis method.

Table 1: Benchmark test functions [22, 24].

Test functions	Search space	N	Global optimum
$f_1 = -20\exp\left(-0.02\sqrt{\frac{1}{N}\sum_{i=1}^{N}x_i^2}\right) - \exp\left(\frac{1}{N}\sum_{i=1}^{N}\cos(2\pi x_i)\right) + 20 + e$	$[-30, 30]^N$	30	0 (min)
$f_2 = 0.1\sum_{i=1}^{N}\cos(5\pi x_i) - \sum_{i=1}^{N}x_i^2$	$[-1, 1]^N$	30	$0.1N$ (max)
$f_3 = \exp\left(-0.5\sum_{i=1}^{N}x_i^2\right)$	$[-1, 1]^N$	30	1 (max)
$f_4 = 1 + \frac{1}{4000}\sum_{i=1}^{N}x_i^2 - \prod_{i=1}^{N}\cos\left(\frac{x_i}{\sqrt{i}}\right)$	$[-600, 600]^N$	30	0 (min)
$f_5 = 10N + \sum_{i=1}^{N}\left[x_i^2 - 10\cos(2\pi x_i)\right]$	$[-5.12, 5.12]^N$	30	0 (min)
$f_6 = \sum_{i=1}^{N-1}\left[100\left(x_{i+1} - x_i^2\right)^2 - (x_i - 1)^2\right]$	$[-30, 30]^N$	30	0 (min)
$f_7 = \sum_{i=1}^{N}x_i^2 + \left(\sum_{i=1}^{N}\frac{i}{2}x_i\right)^2 + \left(\sum_{i=1}^{N}\frac{i}{2}x_i\right)^4$	$[-5.12, 5.12]^N$	30	0 (min)
$f_8 = \sum_{i=1}^{N}x_i^2$	$[-5.12, 5.12]^N$	30	0 (min)
$f_9 = \sum_{i=1}^{N}ix_i^2$	$[-5.12, 5.12]^N$	30	0 (min)
$f_{10} = \sum_{i=1}^{N}\lvert x_i\rvert + \prod_{i=1}^{N}\lvert x_i\rvert$	$[-10, 10]^N$	30	0 (min)
$f_{11} = -\sum_{i=1}^{N}\sin(x_i)\sin^{2m}\left(\frac{ix_i^2}{\pi}\right)$	$[0, \pi]^N$	10	−9.66 (min)
$f_{12} = -\sum_{i=1}^{N}x_i\sin\left(\sqrt{x_i}\right)$	$[-500, 500]^N$	30	−12569.487 (min)

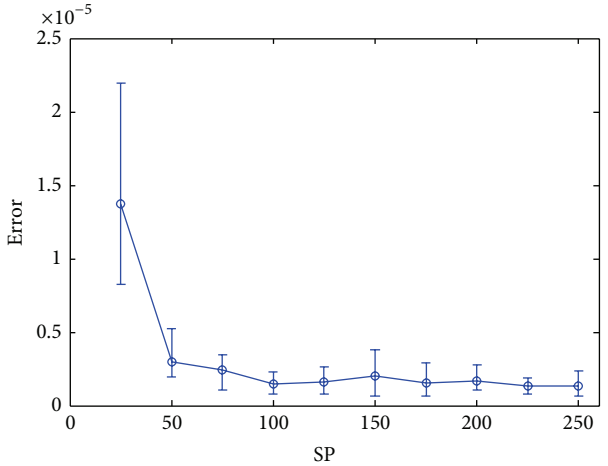


Figure 3: The effect of parameter SP on the performance of IRPEO for test function f_2 when $I_{\max} = 50000$ and $\tau = 1.05$.

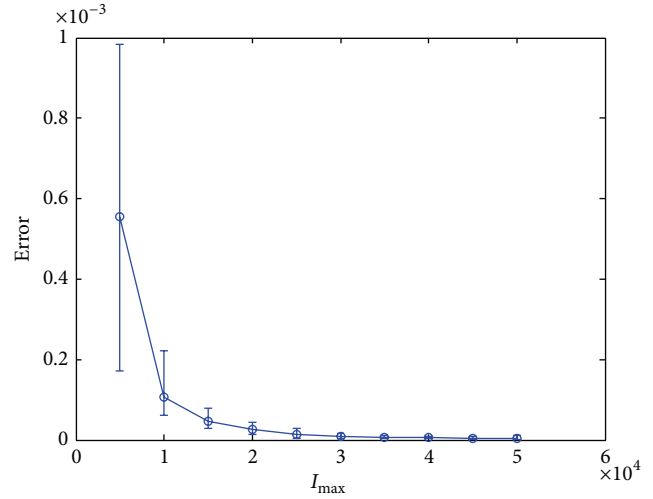


Figure 4: The effect of parameter $I_{\max}$ on the performance of IRPEO for test function f_2 when SP = 50 and $\tau = 1.05$.

Remark 3. On the basis of the above analysis, the performance of IRPEO shown in Tables 2 and 3 is to be improved by tuning the parameters carefully, for example, increasing the values of SP and $I_{\max}$ appropriately and choosing optimal value of τ by trial and error.

Table 2: Comparative performances of IRPEO with the GAs [24] and PEO [22] for test functions.

Problems	Optimum	Algorithm	f_B	f_A (rank)	f_W	SD (rank)	Final rank
f_1	0 (min)	**IRPEO**	**1.150E − 04**	**2.908E − 04 (1)**	**5.150E − 04**	**2.250E − 04 (1)**	**1**
		GA-ADM [24]	/	$3.279E-01$ (7)	/	$2.737E-02$ (6)	7
		GA-RM [24]	/	$1.424E-03$ (4)	/	$7.777E-04$ (3)	4
		GA-PLM [24]	/	$3.471E-04$ (2)	/	$1.173E-04$ (2)	2
		GA-NUM [24]	/	$5.908E-04$ (3)	/	$8.682E-04$ (4)	3
		GA-MNUM [24]	/	$1.816E-01$ (6)	/	$6.085E-02$ (7)	6
		GA-PM [24]	/	$4.459E-01$ (8)	/	$2.104E-01$ (8)	8
		PEO [22]	$9.0E-02$	$1.1E-01$ (5)	$1.2E-01$	$8.40E-03$ (5)	5
f_2	3 (max)	**IRPEO**	**3.00000**	**3.00000(1)**	**3.00000**	**1.397E − 07 (1)**	**1**
		GA-ADM [24]	/	3.00000 (1)	/	$4.714E-07$ (2)	2
		GA-RM [24]	/	2.99995 (4)	/	$2.627E-05$ (4)	4
		GA-PLM [24]	/	2.99998 (3)	/	$9.229E-06$ (3)	3
		GA-NUM [24]	/	2.99985 (5)	/	$1.355E-04$ (5)	5
		GA-MNUM [24]	/	2.98028 (7)	/	$5.024E-02$ (7)	7
		GA-PM [24]	/	2.99358 (6)	/	$5.855E-03$ (6)	6
f_3	1 (max)	**IRPEO**	**1.000000**	**1.000000(1)**	**1.000000**	**0.000E + 00 (1)**	**1**
		GA-ADM [24]	/	1.000000 (1)	/	$6.000E-08$ (2)	2
		GA-RM [24]	/	0.999994 (5)	/	$2.387E-06$ (4)	5
		GA-PLM [24]	/	0.999996 (4)	/	$1.120E-06$ (3)	4
		GA-NUM [24]	/	0.999986 (6)	/	$1.355E-04$ (5)	6
		GA-MNUM [24]	/	0.999997 (3)	/	$5.024E-02$ (7)	3
		GA-PM [24]	/	0.999560 (7)	/	$5.855E-03$ (6)	7
f_4	0 (min)	**IRPEO**	**1.042E − 05 (1)**	**1.920E − 05 (1)**	**3.252E − 05**	**6.231E − 06 (1)**	**1**
		GA-ADM [24]	/	$4.432E-03$ (2)	/	$7.872E-03$ (5)	2
		GA-RM [24]	/	$1.362E-02$ (4)	/	$1.582E-02$ (7)	4
		GA-PLM [24]	/	$5.736E-03$ (3)	/	$7.968E-03$ (6)	3
		GA-NUM [24]	/	$2.447E-02$ (5)	/	$9.881E-06$ (3)	5
		GA-MNUM [24]	/	$3.655E-02$ (6)	/	$1.724E-06$ (2)	6
		GA-PM [24]	/	$1.021E+00$ (7)	/	$3.657E-04$ (4)	7
f_5	0 (min)	**IRPEO**	**2.468E − 05**	**6.189E − 05 (1)**	**1.065E − 04**	**2.400E − 04 (1)**	**1**
		GA-ADM [24]	/	$1.360E+01$ (7)	/	$2.482E+00$ (6)	7
		GA-RM [24]	/	$1.296E-02$ (3)	/	$1.062E-02$ (3)	3
		GA-PLM [24]	/	$4.310E-03$ (2)	/	$2.389E-03$ (2)	2
		GA-NUM [24]	/	$4.432E-02$ (4)	/	$3.624E-02$ (4)	4
		GA-MNUM [24]	/	$1.801E+01$ (8)	/	$7.725E+00$ (8)	8
		GA-PM [24]	/	$8.233E+00$ (6)	/	$6.066E+00$ (7)	6
		PEO [22]	$1.850E+00$	$2.140E+00$ (5)	$2.470E+00$	$2.500E-01$ (5)	5
f_6	0 (min)	**IRPEO**	**5.360E − 02**	**8.64E − 01 (1)**	**8.64E − 01 (1)**	**1.273E − 01 (1)**	**1**
		GA-ADM [24]	/	$6.584E+00$ (4)	/	$4.512E+00$ (4)	4
		GA-RM [24]	/	$6.981E+01$ (7)	/	$4.075E+01$ (6)	7
		GA-PLM [24]	/	$5.126E+01$ (5)	/	$3.432E+01$ (5)	5
		GA-NUM [24]	/	$6.670E+01$ (6)	/	$4.945E+01$ (7)	6
		GA-MNUM [24]	/	$4.032E+00$ (3)	/	$4.002E+00$ (3)	3
		GA-PM [24]	/	$4.043E+03$ (8)	/	$1.617E+04$ (8)	8
		PEO [22]	$9.30E+00$	$9.42E+00$ (2)	$9.63E+00$	$1.30E-01$ (2)	2

Table 2: Continued.

Problems	Optimum	Algorithm	f_B	f_A (rank)	f_W	SD (rank)	Final rank
		IRPEO	$1.880E-01$	$2.960E-01$ (3)	$5.590E-01$	$9.945E-02$ (3)	3
		GA-ADM [24]	/	$0.000E+00$ (1)	/	$0.000E+00$ (1)	1
		GA-RM [24]	/	$4.052E+00$ (5)	/	$5.574E+00$ (5)	5
f_7	0 (min)	GA-PLM [24]	/	$3.076E-01$ (4)	/	$3.246E-01$ (4)	4
		GA-NUM [24]	/	$4.877E+01$ (6)	/	$3.077E+01$ (6)	6
		GA-MNUM [24]	/	$0.000E+00$ (1)	/	$0.000E+00$ (1)	1
		GA-PM [24]	/	$7.091E+01$ (7)	/	$3.038E+01$ (7)	7
		IRPEO	$1.240E-07$	$3.130E-07$ (3)	$5.370E-07$	$1.212E-07$ (3)	3
		GA-ADM [24]	/	$0.000E+00$ (1)	/	$0.000E+00$ (1)	1
		GA-RM [24]	/	$3.310E-06$ (6)	/	$1.811E-06$ (5)	6
f_8	0 (min)	GA-PLM [24]	/	$5.618E-07$ (4)	/	$3.460E-07$ (4)	4
		GA-NUM [24]	/	$1.128E-06$ (5)	/	$1.701E-06$ (6)	5
		GA-MNUM [24]	/	$0.000E+00$ (1)	/	$0.000E+00$ (1)	1
		GA-PM [24]	/	$1.195E-02$ (7)	/	$9.260E-03$ (7)	7
		IRPEO	$2.440E-06$	$4.590E-06$ (3)	$7.580E-06$	$1.745E-06$ (3)	3
		GA-ADM [24]	/	$0.000E+00$ (1)	/	$0.000E+00$ (1)	1
		GA-RM [24]	/	$7.572E-05$ (6)	/	$1.083E-04$ (6)	6
f_9	0 (min)	GA-PLM [24]	/	$5.361E-06$ (4)	/	$5.511E-06$ (4)	4
		GA-NUM [24]	/	$2.504E-05$ (5)	/	$4.956E-05$ (5)	5
		GA-MNUM [24]	/	$0.000E+00$ (1)	/	$0.000E+00$ (1)	1
		GA-PM [24]	/	$4.649E-01$ (7)	/	$3.203E-01$ (7)	7
		IRPEO	$1.022E-04$	$2.167E-04$ (3)	$4.232E-04$	$2.030E-04$ (3)	3
		GA-ADM [24]	/	$0.000E+00$ (1)	/	$0.000E+00$ (1)	1
		GA-RM [24]	/	$1.771E-03$ (5)	/	$6.387E-03$ (5)	5
f_{10}	0 (min)	GA-PLM [24]	/	$3.358E-04$ (4)	/	$7.331E-04$ (4)	4
		GA-NUM [24]	/	$7.410E-03$ (6)	/	$3.535E-02$ (6)	6
		GA-MNUM [24]	/	$0.000E+00$ (1)	/	$0.000E+00$ (1)	1
		GA-PM [24]	/	$4.600E-02$ (7)	/	$1.101E-01$ (7)	7

Table 3: Comparative performances of IRPEO with PEO-EO, PSO, PEO, and GA [22] for test functions.

Problems	Optimum	Algorithm	f_B (rank)	f_A (rank)	f_W (rank)	SD (rank)	Final rank
		IRPEO	**−9.66(1)**	**−9.66(1)**	**−9.66(1)**	**5.767E − 05 (1)**	**1**
		PEO-EO [22]	−9.66 (1)	−9.66 (1)	−9.66 (1)	$2.15E-03$ (2)	2
f_{11}	−9.66 (min)	PSO [22]	−9.66 (1)	−9.52 (5)	−9.06 (5)	0.17 (5)	5
		GA [22]	−9.66 (1)	−9.62 (3)	−9.50 (3)	0.06 (4)	3
		PEO [22]	−9.61 (5)	−9.55 (4)	−9.50 (3)	0.03 (3)	4
		IRPEO	**−12569.5(1)**	**−12569.5(1)**	**−12569.4(1)**	**7.629E − 03 (1)**	**1**
		PEO-EO [22]	−12569.5 (1)	−12568.0 (2)	−12562.6 (2)	2.01 (2)	2
f_{12}	−12569.5 (min)	PSO [22]	−9577.7 (5)	−10139.3 (4)	−11026.2 (4)	625.7 (5)	4
		GA [22]	−9549.3 (4)	−8846.0 (5)	−8404.5 (5)	481.0 (4)	5
		PEO [22]	−12214.2 (3)	−12083.3 (3)	−11977.3 (3)	90.3 (3)	3

5. Conclusion

In this paper, an improved real-coded population-based EO method (IRPEO) has been proposed to solve continuous unconstrained optimization problems. The proposed IRPEO is population-based iterated optimization consisting of the following operations: generation of real-coded random initial population, evaluation of individual and population fitness, selection of bad elements according to power-law probability distribution, generation of new population based on uniform random mutation, and updating the population by accepting the new population unconditionally.

Table 4: Parameters used in different evolutionary algorithms.

Algorithm	Number of parameters	Parameters
GA [24]	At least 4	Population size (SP), maximum number of iterations (I_{max}), the probability of crossover, the parameter used in mutation operations
PSO [22]	At least 6	SP, I_{max}, inertia weight factor w_{max}, w_{min}, acceleration parameter c_1, c_2
PEO [22]	At least 4	SP, I_{max}, dynamical multi-α (i.e., parameter used in Lévy mutation)
PSO-EO [22]	8	SP, I_{max}, w_{max}, w_{min}, c_1, c_2, TC and TG used in the hybrid GC mutation
IRPEO	3	SP, I_{max}, power-law coefficient τ

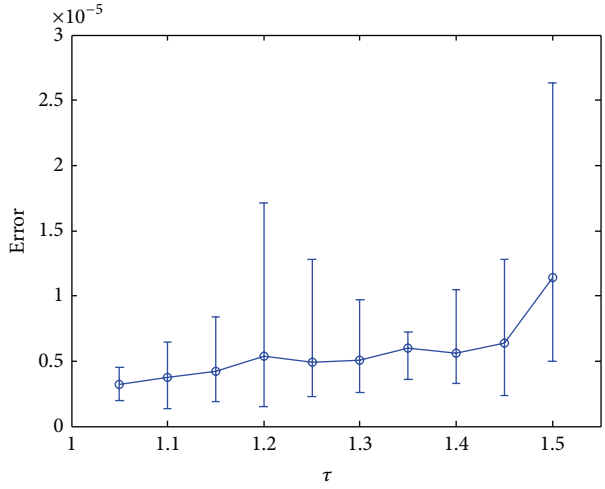


Figure 5: The effect of parameter τ on the performance of IRPEO for test function f_2 when $I_{max} = 50000$ and SP = 50.

The experimental results on 10 continuous unconstrained optimization benchmark test functions have shown that the average performance of IRPEO is competitive or even better than that of various GAs [24] with different mutation operations and the original PEO algorithm [22]. In addition, the superiority of IRPEO to other evolutionary algorithms such as PSO, original PEO, and the hybrid PSO-EO algorithm [22] is also demonstrated by the experimental results on these benchmark functions. Furthermore, the effect of the adjustable parameters used in IRPEO on its performance is discussed in this work. In fact, the IRPEO algorithm is also enhanced by tuning the parameters carefully. However, more experiments for benchmark test functions and real-world engineering optimization problems will be done to further validate the superiority of IRPEO to other optimization algorithms. Moreover, the extension of IRPEO algorithm to constrained optimization problems is another future research work.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors gratefully acknowledge the helpful comments and suggestions of the reviewers and editor, which have improved the presentation. This work is partially supported by the National Natural Science Foundation of China (no. 51207112), Program of "Xinmiao" (Potential) Talents in Zhejiang Province (no. 2012R424044), and Training Programs of Innovation and Entrepreneurship for Undergraduates in Wenzhou University (no. DC2012060).

References

[1] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, New York, NY, USA, 1999.

[2] T. Bäck and H. P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary Computation*, vol. 1, no. 1, pp. 1–23, 1993.

[3] P. Bak, C. Tang, and K. Wiesenfeld, "Self-organized criticality," *Physical Review Letters*, vol. 59, pp. 381–384, 1987.

[4] P. Bak and K. Sneppen, "Punctuated equilibrium and criticality in a simple model of evolution," *Physical Review Letters*, vol. 71, no. 24, pp. 4083–4086, 1993.

[5] S. Boettcher and A. G. Percus, "Nature's way of optimizing," *Artificial Intelligence*, vol. 119, pp. 275–286, 2000.

[6] S. Boettcher and A. G. Percus, "Optimization with extremal dynamics," *Physical Review Letters*, vol. 86, no. 23, pp. 5211–5214, 2001.

[7] Y.-Z. Lu, M.-R. Chen, and Y.-W. Chen, "Studies on extremal optimization and its applications in solving real world optimization problems," in *Proceedings of the IEEE Symposium on Foundations of Computational Intelligence (FOCI '07)*, pp. 162–168, Honolulu, Hawaii, USA, April 2007.

[8] S. Boettcher, "Extremal optimization: heuristics via coevolutionary avalanches," *Computing in Science & Engineering,*, vol. 2, no. 6, pp. 75–82, 2000.

[9] S. Boettcher and A. G. Percus, "Extremal optimization for graph partitioning," *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, vol. 64, no. 2, Article ID 026114, 2001.

[10] S. Boettcher and A. G. Percus, "Extremal optimization at the phase transition for the three-coloring problem," *Physical Review E*, vol. 69, Article ID 066703, 2004.

[11] Y.-W. Chen, Y.-Z. Lu, and P. Chen, "Optimization with extremal dynamics for the traveling salesman problem," *Physica A: Statistical Mechanics and its Applications*, vol. 385, no. 1, pp. 115–123, 2007.

[12] J. M. Liu, Y. W. Chen, G. K. Yang, and Y. Z. Lu, "Self-organized combinatorial optimization," *Expert Systems With Applications*, vol. 38, pp. 10532–10540, 2011.

[13] M. B. Menaï and M. Batouche, "An effective heuristic algorithm for the maximum satisfiability problem," *Applied Intelligence*, vol. 24, pp. 227–239, 2006.

[14] G. Q. Zeng, Y. Z. Lu, and W. J. Mao, "Modified extremal optimization for the maximum satisfiability problem," *Journal of Zhejiang University: Science C (Computers & Electronics)*, vol. 12, no. 7, pp. 589–596, 2011.

[15] Y. W. Chen, Y. Z. Lu, and G. K. Yang, "Hybrid evolutionary algorithm with marriage of genetic algorithm and extremal optimization for production scheduling," *International Journal of Advanced Manufacturing Technology*, vol. 36, no. 9-10, pp. 959–968, 2008.

[16] S. Boettcher, "Evolutionary dynamics of extremal optimization," *Lecture Notes on Computer Science*, vol. 5581, pp. 1–14, 2009.

[17] G. Q. Zeng and Y. Z. Lu, "Survey on computational complexity with phase transitions and extremal optimization," in *Proceedings of the 48th IEEE Conference on Decision and Control held jointly with 28th Chinese Control Conference*, pp. 4352–4359, 2009.

[18] T. Zhou, W.-J. Bai, L.-J. Cheng, and B.-H. Wang, "Continuous extremal optimization for Lennard-Jones clusters," *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, vol. 72, no. 1, Article ID 016702, 2005.

[19] F. L. Sousa and F. M. Ramos, "Function optimization using extremal dynamics," in *Proceedings of the 4th International Conference on Inverse Problems in Engineering*, pp. 1–5, 2002.

[20] F. L. de Sousa, F. M. Ramos, P. Paglione, and R. M. Girardi, "New stochastic algorithm for design optimization," *AIAA Journal*, vol. 41, no. 9, pp. 1808–1818, 2003.

[21] M. R. Chen, Y. Z. Lu, and G. K. Yang, "Population-based extremal optimization with adaptive Lévy mutation for constained optimization," in *Computational Intelligence and Security*, vol. 4456 of *Lecture Notes in Computer Science*, pp. 144–155, 2007.

[22] M.-R. Chen, X. Li, X. Zhang, and Y.-Z. Lu, "A novel particle swarm optimizer hybridized with extremal optimization," *Applied Soft Computing Journal*, vol. 10, no. 2, pp. 367–373, 2010.

[23] X. Li, J. Luo, M.-R. Chen, and N. Wang, "An improved shuffled frog-leaping algorithm with extremal optimisation for continuous optimisation," *Information Sciences*, vol. 192, pp. 143–151, 2012.

[24] P. H. Tang and M. H. Tseng, "Adaptive directed mutation for real-coded genetic algorithms," *Applied Soft Computing*, vol. 13, pp. 600–614, 2013.

[25] G.-Q. Zeng, Y.-Z. Lu, W.-J. Mao, and J. Chu, "Study on probability distributions for evolution in modified extremal optimization," *Physica A: Statistical Mechanics and its Applications*, vol. 389, no. 9, pp. 1922–1930, 2010.