Available online at www.sciencedirect.com
ScienceDirect
Progress in Natural Science 18 (2008) 345–351
Progress in Natural Science

Special report

A novel evolutionary algorithm for global numerical optimization with continuous variables

Wenhong Zhao [a,b], Wei Wang [a], Yuping Wang [a,*]

[a] School of Computer Science and Technology, Xidian University, Xi'an 710071, China
[b] Faculty of Science, Xidian University, Xi'an 710071, China

Received 29 October 2007; received in revised form 19 November 2007; accepted 19 November 2007

Abstract

Evolutionary algorithms (EAs) are a class of general optimization algorithms which are applicable to functions that are multimodal, non-differentiable, or even discontinuous. In this paper, a novel evolutionary algorithm is proposed to solve global numerical optimization with continuous variables. In order to make the algorithm more robust, the initial population is generated by combining determinate factors with random ones, and a decent scale function is designed to tailor the crossover operator so that it can not only find the decent direction quickly but also keep scanning evenly in the whole feasible space. In addition, to improve the performance of the algorithm, a mutation operator which increases the convergence-rate and ensures the convergence of the proposed algorithm is designed. Then, the global convergence of the presented algorithm is proved in detail. Finally, the presented algorithm is executed to solve 24 benchmark problems, and the results show that the convergence-rate of the proposed algorithm is much faster than that of the compared algorithms.

Keywords: Evolutionary algorithm; Decent scale function; Global numerical optimization; Global convergence

1. Introduction

During the past three decades, global optimization problems have been intensively studied in various areas. A number of algorithms for the global optimization problem arise and all these can be divided roughly into two main classes: the determinate method [1,2] and stochastic modeling method [3,4]. In global optimization problems, algorithms may tend to get stuck in local minima, and the convergence-rates of them are usually very low when there are numerous local optima [5].

Evolutionary algorithm (EA) is a kind of global random search methods based on life evolution mechanisms. It contains Genetic Algorithm (GA) [6], Evolutionary Programming (EP) [7], Evolutionary Strategy (ES) [8] and Genetic Programming (GP) [9]. The main features of the EA are swarm exploration and global performance. EA is suitable for the problems with both discrete variables and continuous variables, and does not need to get exact priori knowledge on the problems. In the existing algorithms, EA has received considerable attention regarding its potential to solve complex global optimization problems. However, low convergence-rate and prematurity are also challenging problems for EA.

The crossover operator and mutation operator are the main components to improve the EA's behavior [7,10]. Improvements have been sought in the optimal crossover rates, mutation rates and a more powerful alternative crossover or mutation [11]. In this paper, to enhance the algorithm, a crossover operator which keeps global search when finding descent directions and a mutation operator which balances global exploration and local search are

* Corresponding author. Tel.: +86 15829630075.
E-mail address: ywang@xidian.edu.cn (Y. Wang).

1002-0071/$ - see front matter © 2007 National Natural Science Foundation of China and Chinese Academy of Sciences. Published by Elsevier Limited and Science in China Press. All rights reserved.
doi:10.1016/j.pnsc.2007.11.008

designed. Furthermore, to speed up the convergence, when producing the initial population, both determinate method and random method are employed. Based on these operations, a novel evolutionary algorithm for global numerical optimization with continuous variables is proposed.

2. Novel evolutionary algorithm for global optimization

Consider the following global optimization problem:

$$\min_{L\leqslant X\leqslant U} f(X) \tag{1}$$

where $X=(x_1, x_2,\ldots,x_N)^T$ is a variable vector in R^N, N is the dimension of the problem, $f(X)$ is the objective function, and $L=(l_1,\ldots,l_n)^T$ and $U=(u_1,\ldots,u_N)^T$ define the feasible solution space. We denote the domain of x_i by $[l_i, u_i]$, and the feasible solution space by $[L, U]$.

2.1. Initial population

Here each individual is taken as a vector of floating-point numbers, with the same length as that of decision variables. The initial population is generated as follows: $(x_1, x_2,\ldots,x_N)^T$ represents a solution to the optimization problem. *pop* individuals, where *pop* is population size, are produced by the following two algorithms. $1/m$ initial population, where m is a pre-specified number, is produced by Algorithm 1; others are generated by Algorithm 2.

Algorithm 1.

Step 1: Produce a random vector ra in $[0, 1]^N$ uniformly. Let $k=1$.
Step 2: Compute $X=L+(U-L)\times k/\lceil pop/m\rceil+(U-L)\cdot ra/\lceil pop/m\rceil$, $k=k+1$.
Step 3: If $k\leqslant\lceil pop/m\rceil$, go to step 2.

Remark 1. Algorithm 1 divides the domain $[L, U]$ into $\lceil pop/m\rceil$ equal parts, and generates one individual in every part uniformly. Thus, some determinate factors are added into the initial population.

Algorithm 2.

Step 1: Produce a random vector ra in $[0, 1]^N$ uniformly. Let $k=1$.
Step 2: Compute $X=L+(U-L)\cdot ra$, $k=k+1$.
Step 3: If $k\leqslant pop-\lceil pop/m\rceil$, go to step 2.

Remark 2. In Algorithm 2, $(pop-\lceil pop/m\rceil)$ individuals are produced randomly in the searching space.

2.2. Crossover operator

In order to find the decent direction quickly, a decent scale function is introduced in this subsection. And according to the relationship between the population and the decent scale function, a crossover operator is constructed.

The decent scale function is defined as $\hat{y}=f(\widetilde{X})-\delta$, where $\widetilde{X}$ denotes the optimal solution in the population of the current generation. In numerical experiments of Section 4, $\delta=\lambda\mid f(\widetilde{X})\mid$, $\lambda=1/10000$.

Definition 1. For problem (1) and points X, $Y\in[L, U]$, X is better than Y if $f(X)<f(Y)$.

Let $X=(x_1, x_2,\ldots,x_N)^T$ and $Y=(y_1, y_2,\ldots,y_N)^T$ be the crossover parents, t the current generation, $E_{n,1}=(1, 1,\ldots,1)^T$ the unit vector, and g_0 a parameter for generating the temporary offspring. The crossover offspring is produced by Algorithm 3.

Algorithm 3.

Step 1: Let $Z_1=(z_{11}, z_{12},\ldots,z_{1N})^T=\alpha_1X+\beta_1Y$, $Z_2=(z_{21}, z_{22},\ldots,z_{2N})^T=\alpha_2X+\beta_2Y$, where α_1, β_1, α_2, β_2 are real numbers such that z_1 and z_2 are in the feasible solution space. Compute $f(Z_1)$, $f(Z_2)$, $f(X)$, $f(Y)$ and add Z_1, Z_2, X, Y to the temporary offspring set. Let $g=1$, $j=4$, where j denotes the size of the temporary offspring set.
Step 2: Denote the better one between X and Y as $V=(v_1, v_2,\ldots,v_N)^T$. For k=1 to 2,

For $1\leqslant i\leqslant N$, denote the line passing through $(v_i, f(V))$ and $(z_{ki}, f(Z_k))$ as L_1, the line presenting $\hat{y}=f(\widetilde{X})-\delta$ as L_2. If L_1 and L_2 intersect at point $p_{int}=(p_{int}.\mathrm{x}, p_{int}.\mathrm{y})$, let $\widetilde{Z}_{j+k,i}=p_{int}.\mathrm{x}$. Otherwise, $\widetilde{Z}_{j+k,i}=v_i$. Let the $(j+k)$th temporary offspring $Z_{j+k}=(\widetilde{Z}_{j+k,1},\ldots,\widetilde{Z}_{j+k,N})^T$ and add Z_{j+k} to the temporary offspring set. $j=j+2$.

Step 3: Select crossover offspring in the temporary offspring set. Denote the temporary offsprings better than $\widetilde{X}$ as $Z_1, Z_2,\ldots,Z_q$.

If $q<2$ and $g<g_0$, let $\hat{y}=\hat{y}-\delta$, $g=g+1$, go to step 2; otherwise, choose the best two individuals in the temporary offspring set as the crossover offspring. Stop.

From the above algorithm, the crossover offsprings are always better than the parents.

2.3. Mutation operator

The non-uniform mutation operator is introduced in Ref. [12] is as follows:

$$x'_k=\begin{cases} x_k+(u_k-x_k)r[1-t/T]^b & \text{if random}(0,1)=0\\ x_k-(x_k-l_k)r[1-t/T]^b & \text{if random}(0,1)=1\end{cases}$$

where $X=\{x_1,\ldots,x_N\}^T$ is the mutation parent, $k=1\sim n, X'=(x'_1,\ldots,x'_N)^T$ is the resulting offspring, t is the current generation, T is the maximal number of generations,r is a random number in [0, 1] uniformly, b is a system parameter which determines the dependence degree of

random disturbance on evolutionary generation t, as a rule, $b = 2$.

Although the non-uniform mutation operator scans in the whole feasible space evenly at the beginning, the exploration becomes localized with the generation increasing. In order to overcome this, the following improvements are made on the non-uniform mutation operator.

(a) Add the following part.

$$x'_k = \begin{cases} x_k + (u_k - x_k)r[t/T]^b & \text{if random}(0,1) = 0 \\ x_k - (x_k - l_k)r[t/T]^b & \text{if random}(0,1) = 1 \end{cases}$$

This added part explores the whole searching space uniformly during the execution of the algorithm. Hence, it overcomes the shortage of the non-uniform mutation operator. By integrating the above two parts, not only the abilities of the global search and local exploration are balanced, but also the diversity of the population increases. As a result, the premature convergence is avoided with high probability.

(b) In order to increase the low convergence-rate and reduce the blindness of the non-uniform mutation, we improve the production of random number r.

Definition 2. Assume that the precision of solution is ε, X is a ε-precision optimum if $\|X - X^*\|_1 \leqslant \varepsilon$, where $\varepsilon > 0$ is small enough, and X^* is the global optimum.

r in the improved mutation is chosen as Algorithm 4.

Algorithm 4.

Step 1: Divide [0,1] into subintervals $[w_1^j, w_2^j], \ldots, [w_{n_j-1}^j, w_{n_j}^j]$ such that $| w_k^j - w_{k-1}^j | < \varepsilon$, $k = 2 \sim n_j$, where $w_1^j = 0, w_{n_j}^j = 1$.
Step 2: For x_j, an arbitrary component of X, produce a random number $r \in [0,1]$ uniformly, if $p_m > r$, choose w_r^j in the set $\{w_1^j, \ldots, w_{n_j}^j\}$ randomly and let $r = w_r^j$. Otherwise, $r = 0$.
Step 3: Repeat step 1 and step 2 N times, a resulting offspring is generated and denoted as $MO = \{mo_1, mo_2, \ldots, mo_N\}^T$.

From above, the improved non-uniform mutation operator is represented as follows.

$$x'_j = \begin{cases} x_j + (u_j - x_j)r[1 - t/T]^b & \text{if } 0 \leqslant rand \leqslant c_1 \\ x_j - (x_j - l_j)r[1 - t/T]^b & \text{if } c_1 \leqslant rand \leqslant c_2 \\ x_j + (u_j - x_j)r[t/T]^b & \text{if } c_2 \leqslant rand \leqslant c_3 \\ x_j - (x_j - l_j)r[t/T]^b & \text{if } c_3 \leqslant rand \leqslant 1 \end{cases}$$

$j = 1 \sim n$, where $X = \{x_1, x_2, \ldots, x_N\}^T$ is the mutation parent, $X' = (x'_1, \ldots, x'_N)$ is the resulting offspring, t is the current convergence generation, T is the maximal number of convergence generations, *rand* is a random number in [0,1] uniformly, b determines how the random disturbance depends on generation t, c_1, c_2, c_3 satisfies $c_1 + c_3 - c_2 = 0.5$.

From the above formulas, the improved non-uniform mutation operator balances the abilities of global exploration and local search and avoids the algorithm being trapped in the local optima.

2.4. The novel evolutionary algorithm (NEA)

Step 0: Initialization
Step 1: Perform Algorithms 1 and 2 to create an initial population, calculate the function values of the population.
Step 2: Find out the best one among the population and denote it as $\widetilde{X}$. Choose *pop* individuals in the population randomly and execute Algorithm 3 to produce the crossover offspring.
Step 3: Execute the mutation operator on the crossover offspring to generate the mutation offspring. Each crossover offspring is selected for mutation with probability p_m.
Step 4: Select the best *pop* individuals in $\widetilde{X}$, the crossover offspring and mutation offspring as the original population of the next iteration.
Step 5: If the stopping criterion is satisfied, stop. Otherwise, return to Step 2.

3. Global convergence

Definition 3. X' is reachable by X through crossover and mutation operations, namely $Prob\{MC(X) = X'\} > 0$, if X' can be generated by X through these operations, where $MC(X)$ represents the individual produced by crossover and mutation operations on X, $Prob\{\cdot\}$ denotes the probability of random event $\{\cdot\}$.

Definition 4. If $Prob\{\|MC(X) - X'\|_1 \leqslant \varepsilon\} > 0$, it is called that X' is reachable with ε-precision by X undergoing crossover and mutation operations.

For the minimize problem $\min_{X \in \Omega} f(X)$, where Ω is the feasible space, Back has proven the following result [13]:

If an evolutionary algorithm meets: (1) For any X' and X, X' is reachable by X through crossover and mutation operations; (2) The population sequence $P(1)$, $P(2)$, …, $P(k)$, … is monotone, that is, $\min\{f(X) | X \in P(k)\} \geqslant \min\{f(X) | X \in P(k+1)\}$ for any k. It converges to the global optimum with probability one.

Next, we prove that if "reachable" is replaced by "reachable with ε-precision" in the condition (1), NEA always converges to the ε-optimal solution.

Theorem 1. *If $\varepsilon > 0$ is small enough and $f(X)$ is continuous in the searching space $[L,U]$, NEA converges to the ε-optimal solution with probability one, namely $Prob\{lim_{k\to\infty}\| P(k) - P^*\|_1 \leqslant \varepsilon\} = 1$, where P^* is the set of the global optimal solutions, and $\{\|P(k)-P^*\|_1 \leqslant \varepsilon\}$ represents that for any $X \in P(k)$, there exists an $X^* \in P^*$ such that $\|X-X^*\|_1 \leqslant \varepsilon$.*

Proof. At first, we prove that for any two points $X' = (x'_1, x'_2, .., x'_N)^T$ and $X=(x_1, x_2, \ldots, x_N)^T \in [L,U]$, X' is reachable with ε-precision by X through crossover and mutation operations, namely $Prob\{\|MC(X)-X'\|_1 \leqslant \varepsilon\} > 0$.

Assume that $\overline{X}$ is an arbitrary point generated by X through crossover, namely $C(X) = \overline{X}$. It only needs to prove that X' is reachable with ε-precision by $\overline{X}$ through mutation, namely $Prob\{\|M(\overline{X}) - X'\|_1 \leqslant \varepsilon\} > 0$. Let $\overline{X} = (\bar{x}_1, \ldots, \bar{x}_N)^T$.

For any $x'_k \in X'$ and $x'_k \neq u_k, l_k$;

(1) If $\bar{x}_k < x'_k < u_k$;
   (a) For $(u_k - \bar{x}_k)[1 - t/T]^b$, there always exist a generation t and $r'_k \in (0,1)$ such that $x'_k = \bar{x}_k + (u_k - \bar{x}_k)[1 - t/T]^b r'_k$;
   (b) For $(u_k - \bar{x}_k)(t/T)^b$, there always exist a generation t and $r'_k \in (0,1)$ such that $x'_k = \bar{x}_k + (u_k - \bar{x}_k)(t/T)^b r'_k$.

   For r'_k, there always exists $r^i_{i_k} \in (0,1)$ satisfying $| r^i_{i_k} - r'_k | < \varepsilon$. Thus, we can construct $x^i_{i_k} = \bar{x}_k + (u_k - \bar{x}_k)(t/T)^b r^i_{i_k}$ or $x^i_{i_k} = \bar{x}_k + (u_k - \bar{x}_k)[1 - t/T]^b r^i_{i_k}$ such that $| x^i_{i_k} - x'_k | < \varepsilon$ holds.

(2) If $l_k < x'_k < \bar{x}_k$;
   (a) For $-(\bar{x}_k - l_k)[1 - t/T]^b$, there always exist a generation t and $r'_k \in (0,1)$ such that $x'_k = \bar{x}_k - (\bar{x}_k - l_k)[1 - t/T]^b r'_k$;
   (b) For $-(\bar{x}_k - l_k)(t/T)^b$, there always exist a generation t and $r'_k \in (0,1)$ such that $x'_k = \bar{x}_k - (\bar{x}_k - l_k)(t/T)^b r'_k$.

   For r'_k there always exists $r^i_{i_k} \in (0,1)$ satisfying $| r^i_{i_k} - r'_k | < \varepsilon$. Thus, we can construct $x^i_{i_k} = \bar{x}_k - (\bar{x}_k - l_k)(t/T)^b r^i_{i_k}$ or $x^i_{i_k} = \bar{x}_k - (\bar{x}_k - l_k)[1 - t/T]^b r^i_{i_k}$ such that $| x^i_{i_k} - x'_k | < \varepsilon$ holds.

For any $x'_k \in X'$, if $x'_k = x_k$, let $x^i_{i_k} = x_k$; if $x'_k = u_k$, let $x^i_{i_k} = u_k$; if $x'_k = l_k$, let $x^i_{i_k} = l_k$. $| x^i_{i_k} - x'_k | = 0 < \varepsilon$ holds for all these conditions.

Thus, for any X', there always exists $\widetilde{X} = (x^i_{i_1}, \ldots, x^i_{i_n})$ such that $\|\widetilde{X} - X'\|_1 \leqslant \varepsilon$. Next, we prove that if $Prob\{M(\overline{X}) = \widetilde{X}\} > 0$, $Prob\{\|M(\overline{X}) - X'\|_1 \leqslant \varepsilon\} > 0$.

Assume the Hamming distance between $\overline{X}$ and $\widetilde{X}$ is h. Without loss of generality, let the last $(n-h)$ components of $\overline{X}$ are the same as those of $\widetilde{X}$. Thus, $Prob\{M(\overline{X}) = \widetilde{X}\} \geqslant p_m^h (1 - p_m)^{n-h} (\frac{1}{2T})^h \prod_{i=1}^{h} \frac{1}{n_i} > 0$.

Therefore, X' is reachable with ε- precision by X through crossover and mutation.

Secondly, for each generation k, NEA always chooses the best *pop* individuals as population $P(k)$. Thus, the population sequence $P(1), \ldots, P(k), \ldots$ is monotone. Note that $f(X)$ is continuous on the domain $[L,U]$. It can be concluded that for $\varepsilon > 0$ small enough and any global optimal solution X^*, $f(X_1) > f(X_2)$ holds if $X_1 \in P^*(\varepsilon)$ and $X_2 \notin P^*(\varepsilon)$. The population can be seen as a Markov chain with two states: 1) the population satisfies $\|P(k)-P^*\|_1 \leqslant \varepsilon$; 2) the population doesn't satisfy $\|P(k)-P^*\|_1 \leqslant \varepsilon$.

From the monotony of the population sequence, State 1 transfers to State 2 with the probability 0, that is, State 1 is an absorbing state. Since arbitrary two points are reachable with ε- precision, State 2 transfers to State 1 with probability $p > 0$. So, State 2 is a transient state. By the Markov chain theory [14], Theorem 1 holds. □

4. Results and comparison

In this section, we apply the proposed algorithm to 24 benchmark functions (f_1–f_{23} are from Ref. [7] and f_{24} from Ref. [17]). Fifty independent runs on each test function were performed and the following results were recorded: the mean number of function evaluations (MNFE), the number of function evaluations (NFE), the mean function value (MFV), and the best function value (BFV). In the experiments, $b = 2$, $p_m = 0.3$, $\lambda = 1/10000$, $m = 1/50$. When solving problems with 30 or more dimensions, $c_1 = 0.35$, $c_2 = 0.7$, $c_3 = 0.85$. Otherwise, $c_1 = 0.5$, $c_2 = 1$, $c_3 = 1$.

The performance of NEA is compared with nine existing algorithms in Tables 1 and 2. In the Tables, N represents the result is not available in the corresponding reference. Each of them is executed to solve some of f_1–f_{24}, and all of the available results for comparison have already been included.

Table 1 shows the performance comparisons among FEP, LEP and NEA. In the experiments, the population size of NEA is 100, NEA stops when the number of function evaluations of the FEP or LEP is reached or the best individual cannot be further improved in successive 200 generations, and g_0 is 2. From Table 1, the following conclusions can be drawn: (1) NEA can find optimal or closer-to-optimal solutions. (2) For the most test functions, NEA can obtain better and closer-to-optimal solutions than FEP and LEP. (3) The mean numbers of function evaluations required by NEA are fewer than FEP and LEP. (4) For the most test functions, the number of convergence generation of NEA is smaller than those of others.

In Table 2, we compare NEA with HPSO1C NMS, HPSO2C NMS, HPSO1D NMS, HPSO2D NMS, SPSO, QPSO, and AQPSO. When we perform NEA, the population size is 30 and $g_0 = 1$. Except for f_1, NEA can give significantly better and closer-to-optimal solutions. At the same time, NEA needs fewer mean numbers of function evaluations than other algorithms. Therefore, NEA not only has a smaller time complexity and higher convergence-rate but also has high stability.

Table 1
Comparisons among NEA, FEP and LEP, where results for FEP and LEP are gained from Ref. [7] and Ref. [15], respectively

Test function	Theoretic best	LEP($\alpha = 1$)		FEP		NEA	
		MNFE	MFV	MNFE	MFV	MNFE	MFV
f_1	0	1.50e + 005	0.010173	1.50e + 005	5.7e − 004	1.50e + 005	1.16e − 019
f_2	0	N	N	2.00e + 005	8.1e − 003	2.00e + 005	1.45e − 014
f_3	0	1.50e + 005	172.245618	5.00e + 005	1.6e − 002	5.00e + 005	1.20e − 010
f_4	0	N	N	5.00e + 005	0.3	5.00e + 005	1.83e − 011
f_5	0	1.50e + 005	80.265797	2.00e + 006	5.06	7.29e + 005	1.64e − 002
f_6	0	N	N	1.50e + 005	577.76	5.00e + 002	0
f_7	0	N	N	3.00e + 005	7.6e − 003	2.00e + 005	2.96e − 004
f_8	−12569.5	1.50e + 005	−11433.1817	9.00e + 005	−12554.5	2.80e + 005	−12569.45
f_9	0	1.50e + 005	24.398049	500000	4.6e + 002	8.79e + 004	0
f_{10}	0	1.50e + 005	0.88504	1.50e + 005	1.8e − 002	1.50e + 005	5.61e − 011
f_{11}	0	1.50e + 005	0.012899	2.00e + 005	1.6e − 002	1.03e + 005	0
f_{12}	0	1.50e + 005	0.000097	1.50e + 005	9.2e − 006	8.04e + 004	4.18e − 007
f_{13}	0	1.50e + 005	0.004141	1.50e + 005	1.6e − 004	1.50e + 005	8.43e − 005
f_{14}	1	N	N	1.00e + 004	1.66	4.50e + 002	0.998
f_{15}	0.0003075	N	N	4.00 e + 005	5.0e − 004	2.95e + 005	3.10e − 004
f_{16}	−1.0316285	3000	−1.028753	1.00e + 004	−1.03	9.90e + 003	−1.03
f_{17}	0.398	N	N	1.00e + 004	0.398	1.00e + 004	0.398
f_{18}	3.00	3000	3.183000	1.00e + 004	3.02	7.55e + 003	3.00
f_{19}	−3.86	N	N	1.00e + 004	−3.86	2.62e + 003	−3.86
f_{20}	−3.32	N	N	2.00e + 004	−3.27	1.71e + 004	−3.25
f_{21}	−10	1.00e + 004	−7.841831	1.00e + 004	−5.52	4.75e + 002	−1.02e + 001
f_{22}	−10	1.00e + 004	−9.672865	1.00e + 004	−8.27	4.75e + 002	−1.04e + 001
f_{23}	−10	1.00e + 004	−9.932785	1.00e + 004	−9.10	4.74e + 002	−1.05e + 001

NEA stops when the number of function evaluations of the FEP or LEP is reached or the best individual cannot be further improved in successive 200 generations.

Table 2
Comparisons among NEA and HPSO1C NMS, HPSO2C NMS, HPSO2C NMS, HPSO2D NMS, SPSO, QPSO AND AQPSO, where the results for HPSO1C NMS, HPSO2C NMS, HPSO1D NMS, HPSO2D NMS are obtained from Ref. [16] and the results for SPSO, QPSO, AQPSO are given in Ref. [17]

	f_1	f_5	f_9	f_{10}	f_{11}	f_{24}
HPSO1CNMS–MFV(1)	6.614e − 010	34.5819	29.3964	0.0948	0.0248	N
HPSO2CNMS–MFV(2)	3.780e − 010	44.5354	33.5891	0.1384	0.0241	N
HPSO1DNMS–MFV(3)	1.161e − 009	28.9776	31.5297	0.3952	0.0215	N
HPSO2DNMS–MFV(4)	1.254e − 009	22.5492	34.4734	1.1578	0.0231	N
SPSO–MFV(5)	2.26e − 010	289.593	37.2796	N	0.01267	4.74e − 005
QPSO–MFV(6)	1.87e − 028	59.0291	22.9594	N	0.1161	3.89e − 004
AQPSO–MFV(7)	5.04e − 006	61.6228	35.2366	N	0.01423	1.53e − 005
NEA–MFV	2.92e − 016	1.37	1.77e − 016	1.17e − 009	1.11e − 017	0
NEA–BFV(8)	3.04e − 019	3.36e − 001	0	2.93e − 010	0	0
Interval of (1, 2, 3, 4)	$[-100,100]^n$	$[-30,30]^n$	$[-5.12,5.12]^n$	$[-32,32]^n$	$[-600,600]^n$	N
Initial interval of (5, 6, 7)	$[50,100]^n$	$[15,30]^n$	$[2.56,5.12]^n$	N	$[300,600]^n$	$[30,100]^n$
NFE of NEA	3.00e+004	3.00e+004	3.00e+004	3.00e+004	2.73e+004	1.87e+003

NFE of HPSO1C NMS, HPSO2C NMS, HPSO1D NMS and HPSO2D NMS are 30000. NFE of SPSO, QPSO and AQPSO are 80000.

Table 3 shows the comparison between NEA and CEA (a variation of NEA). The difference between NEA and CEA is that the initial population is only generated randomly in CEA. For f_1–f_5, f_7–f_8, f_{10}, f_{12}–f_{13}, f_{15}–f_{16}, f_{21}–f_{24}, NEA obtains better solutions than CEA. For f_6, f_9, f_{11}, f_{18}, f_{19}, NEA gives the same optimal solutions and fewer mean numbers of function evaluations than CEA. Only for f_{14}, f_{17}, f_{20}, NEA gives the same mean numbers of function evaluations. In conclusion, combining determinate factors with random ones in the initial population production makes NEA more efficient and robust.

In Table 4, the comparison between NEAM and NUM is shown, where NEAM is an algorithm which only uses the mutation operator in NEA and NUM is an algorithm which only uses the previous non-uniform mutation. It can be seen that, except for f_8, the improved mutation operator gets much better solution and has faster convergence-rate and higher robust.

5. Conclusion

In this paper, we present a novel algorithm, NEA, to solve global numerical optimization problems. It uses a new crossover based on the descent scale function and an improved non-uniform mutation. The experiments indicate that this mutation approach works better than the previous

Table 3
Comparison between NEA and CEA

Test function	NEA			CEA		
	MNFE	MFV	BFV	MNFE	MFV	BFV
f_1	1.50e + 005	1.16e − 019	1.25e − 021	1.50e + 005	7.81e − 018	3.01e − 019
f_2	2.00 e + 005	1.45e − 014	1.13e − 014	2.00 e + 005	4.70e − 014	1.67e − 014
f_3	5.00e + 005	1.20e − 010	1.80e − 013	5.00e + 005	2.45e − 007	5.98e − 009
f_4	5.00e + 005	1.83e − 011	3.28e − 013	5.00e + 005	2.21e − 010	7.40e − 012
f_5	7.29e + 005	1.64e − 002	9.35e − 003	3.82e + 005	2.86e + 001	2.75e + 001
f_6	5.00e + 002	0	0	1.20e + 004	0	0
f_7	2.00e + 005	2.96e − 004	1.93e − 006	2.83e + 005	5.35e − 004	1.30e − 004
f_8	2.80e + 005	−12569.45	−12569.46	8.99e + 005	−9.94e + 003	−1.05e + 004
f_9	8.79e + 004	0	0	1.22e + 005	0	0
f_{10}	1.50e + 005	5.61e − 011	7.89e − 012	1.50e + 005	5.00e − 010	1.82e − 010
f_{11}	1.03e + 005	0	0	1.37e + 005	0	0
f_{12}	8.04e + 004	4.18e − 007	1.40e − 007	1.50e + 005	1.47e − 001	7.22e − 002
f_{13}	1.50e + 005	8.43e − 005	7.01e − 006	1.50e + 005	1.80	1.53
f_{14}	4.50e + 002	0.998	0.998	2.83e + 003	0.998	0.998
f_{15}	2.95e + 005	3.10e − 004	3.07e − 004	3.66e + 005	7.69e − 004	4.03e − 004
f_{16}	9.90e + 003	−1.03	−1.0316	9.90e + 003	−1.03	−1.03
f_{17}	1.00e + 004	0.398	0.398	1.00e + 004	0.398	0.398
f_{18}	7.55e + 003	3.00)	3.00	6.88e + 003	3.00	3.00
f_{19}	2.62e + 003	−3.86	−3.86	1.55e + 003	−3.86	−3.86
f_{20}	1.71e + 004	−3.25	−3.32	1.62e + 004	−3.27	−3.32
f_{21}	4.75e + 002	−1.02e + 001	−1.02e + 001	4.74e + 003	−9.32	−1.00e + 001
f_{22}	4.75e + 002	−1.04e + 001	−1.04e + 001	4.55e + 003	−9.51	−1.03e + 001
f_{23}	4.74e + 002	−1.05e + 001	−1.05e + 001	3.20e + 003	−1.02e + 001	−1.04e + 001
f_{24}	7.71e + 003	0	0	9.87e + 003	1.34e − 002	0

Table 4
Comparison between NEAM and NUM

Test function	NEAM			NUM		
	MNFE	MFV	BFV	MNFE	MFV	BFV
f_1	5.00e + 003	1.84e + 001	2.79	5.00e + 003	3.80e + 004	3.11e + 004
f_2	5.00e + 003	8.47e − 001	3.78e − 001	5.00e + 003	1.74e + 002	7.66e + 001
f_3	5.00e + 003	3.96e + 003	6.15e + 002	5.00e + 003	5.38e + 004	4.51e + 004
f_4	5.00e + 003	6.16	3.38	5.00e + 003	7.48e + 001	6.35e + 001
f_5	5.00e + 003	4.92e + 002	9.31e + 001	5.00e + 003	9.43e + 007	6.28e + 007
f_6	5.00e + 003	1.62e + 001	5.00	5.00e + 003	3.86e + 004	2.58e + 004
f_7	5.00e + 003	2.80e − 002	4.09e − 003	5.00e + 003	4.20e + 001	3.09e + 001
f_8	5.00e + 003	−5.20e + 003	−6.12e + 003	5.00e + 003	−6.46e + 003	−8.09e + 003
f_9	5.00e + 003	9.54	3.47	5.00e + 003	3.08e + 002	2.75e + 002
f_{10}	5.00e + 003	1.71	6.84e − 001	5.00e + 003	1.96e + 001	1.88e + 001
f_{11}	5.00e + 003	1.15	9.91e − 001	5.00e + 003	3.39e + 002	2.54e + 002
f_{12}	5.00e + 003	3.98e − 001	1.87e − 001	5.00e + 003	1.51e + 008	3.63e + 007
f_{13}	5.00e + 003	4.44	2.21	5.00e + 003	3.79e + 008	1.79e + 008
f_{14}	1.12e + 003	0.998	0.998	1.56e + 003	0.998	0.998
f_{15}	5.00e + 003	1.15e − 003	3.11e − 004	5.00e + 003	1.33e − 003	7.80e − 004
f_{16}	5.00e + 003	−1.03	−1.03	5.00e + 003	−9.00e − 001	−1.02
f_{17}	1.22e + 003	3.98e − 001	0.398	3.23e + 003	0.398	0.398
f_{18}	5.00e + 003	3.00	3.00	5.00e + 003	3.00	3.00
f_{19}	2.20e + 003	−3.86	−3.86	1.47e + 003	−3.86	−3.86
f_{20}	5.00e + 003	−3.28	−3.32	5.00e + 003	−3.22	−3.32
f_{21}	3.98e + 003	−9.23	−1.01e + 001	5.00e + 003	−6.69	−9.85
f_{22}	3.29e + 003	−9.11	−1.03e + 001	4.93e + 003	−7.32	−1.03e + 001
f_{23}	2.72e + 003	−9.79	−1.05e + 001	4.92e + 003	−7.42	−1.02e + 001
f_{24}	1.91e + 003	0	0	5.00e + 003	4.19e − 002	1.97e − 003

one. After proving the convergence of NEA, we execute it to solve 24 benchmark functions. The results show that the proposed NEA finds optimal or closer-to-optimal solutions quickly, and has more statistical soundness and faster convergence-rate than the compared algorithms.

Acknowledgment

This work was supported by National Natural Science Foundation of China (Grant Nos. 60374063 and 60672026).

References

[1] Oblow EM. SPT: a stochastic tunneling algorithm for global optimization. J Global Optim 2001;20(2):191–208.
[2] Yang YJ, Shang YL. A new filled function method for unconstrained global optimization. Appl Math Comput 2006;173(11):501–12.
[3] Locatelli M. Simulated annealing algorithms for continuous global optimization: convergence conditions. J Optim Theory Appl 2000;104(1):121–33.
[4] Wang YP. Improving evolutionary algorithms by a new smoothing technique. In: Proceedings of the 5th International Conference on Intelligent Data Engineering and Automated Learning. Exeter, UK, August 25–27, 2004; p. 746–751.
[5] Li C, Wang XH. Gauss–Newton methods for a class of nonsmooth optimization problems. Prog Nat Sci 2000;10(6):470–3.
[6] Leung YW, Wang YP. An orthogonal genetic algorithm with quantization for global numerical optimization. IEEE Trans Evol Comput 2000;4(4):41–53.
[7] Xin Y, Liu Y, Lin GM. Evolutionary programming made faster. IEEE Trans Evol Comput 1999;3(2):82–102.
[8] Olivier F. An evolutionary strategy for global minimization and its Markov chain analysis. IEEE Trans Evol Comput 1998;2(30):77–90.
[9] Hirsh H. Genetic programming. IEEE Intell Syst 2000;15(3):74–84.
[10] Liepins GE, Vose MD. Characterizing crossover operator in genetic algorithms. Ann Math Artif Intell 1992;5(1):27–34.
[11] Spears WM. Recombination parameters, handbook of evolutionary computation. New York: Oxford University Press; 1997.
[12] Zhou M, Sun SD. Genetic algorithms: theory and applications. Beijing: National Defence Industry Press; 1999.
[13] Back T. Evolutionary algorithms in theory and practice. New York: Oxford University Press; 1996.
[14] Allen AO. Probability statistics and queuing theory with computer science applications. 2nd ed. Boston, MA: Academic Press; 1990.
[15] Lee CY, Yao X. Evolutionary programming using mutations based on the levy probability distribution. IEEE Trans Evol Comput 2004;8(1):1–13.
[16] Gimmler J, Stutzle T, Exner TE. Hybrid particle swarm optimization: an examination of the influence of iterative improvement algorithms on performance. IEEE Trans Evol Comput 2004;8(1):1–13.
[17] Xu WB, Sun J. Adaptive parameter selection of quantum-behaved particle swarm optimization on global level. IEEE Trans Evol Comput 2004;8(1):1–13.