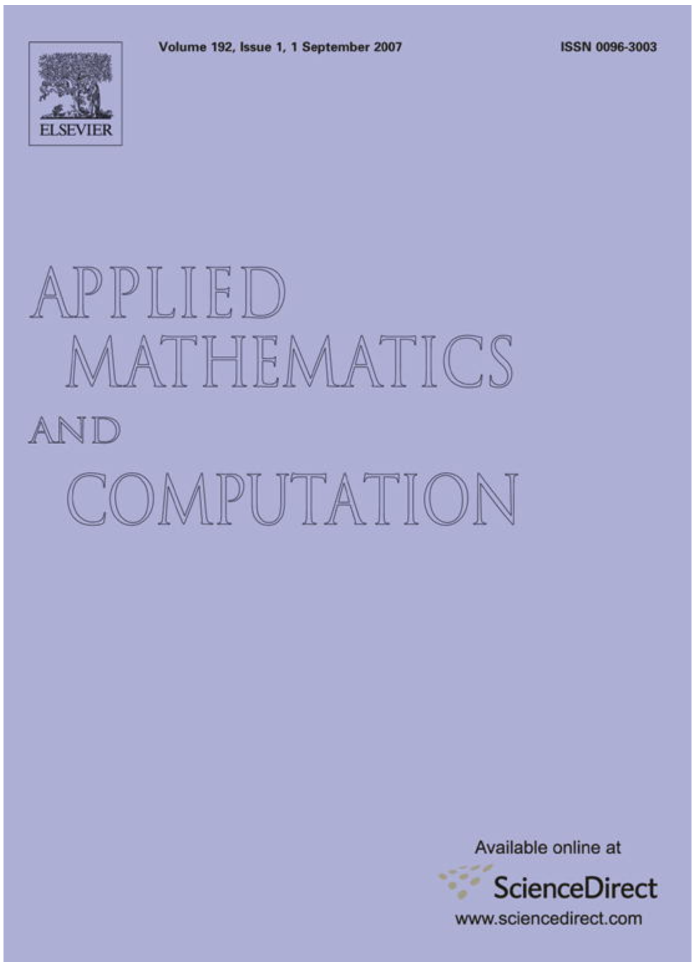

Volume 192, Issue 1, 1 September 2007
ISSN 0096-3003
ELSEVIER
APPLIED MATHEMATICS AND COMPUTATION
Available online at
ScienceDirect
www.sciencedirect.com

Available online at www.sciencedirect.com

Applied Mathematics and Computation 192 (2007) 1–11

APPLIED MATHEMATICS AND COMPUTATION

www.elsevier.com/locate/amc

Evolutionary programming based on non-uniform mutation

Xinchao Zhao [a,b,*], Xiao-Shan Gao [b], Ze-Chun Hu [c]

[a] *School of Sciences, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[b] *KLMM, Institute of Systems Science, AMSS, Academia Sinica, Beijing 100080, China*
[c] *Department of Mathematics, Nanjing University, Nanjing 210093, China*

Abstract

A new evolutionary programming using non-uniform mutation instead of Gaussian, Cauchy and Lévy mutations is proposed. Evolutionary programming with non-uniform mutation (NEP) has the merits of searching the space uniformly at the early stage and very locally at the later stage during the programming. For a suite of 14 benchmark problems, NEP outperforms the improved evolutionary programming using mutation based on Lévy probability distribution (ILEP) for multimodal functions with many local minima while being comparable to ILEP in performance for unimodal and multimodal functions with only a few minima. The detailed theoretical analysis of the executing process of NEP and the expected step size on non-uniform mutation are given.
© 2006 Elsevier Inc. All rights reserved.

Keywords: Evolutionary programming; Non-uniform mutation; Executing process

1. Introduction

Inspired by biological evolution and natural selection, intelligent computation algorithms are proposed to provide powerful tools for solving many difficult problems. Genetic algorithms (GAs) [1–5], evolution strategies (ESs) [6,7], and evolutionary programming (EP) [8,9] are especially noticeable among them. In GAs, the crossover operator plays the major role and mutation is always seen as an assistant operator. In ESs and EP, however, mutation has been considered as the main operator. GAs usually adopt a high crossover probability and a low mutation probability, while ESs and EP apply mutation to every individual. One, two, multipoint, or uniform crossover and uniform mutation [4,10] are often used in GAs. Besides Gaussian mutation [8,9], self-adaptation mutations [11,12], self-adaptation rules from ESs [13], Cauchy [14] and Lévy probability distribution [15] mutations have also been incorporated into EP. These new operators can greatly enhance the performance of the algorithms on some specific problems. About some recent progress in algorithms, please refer to [16–20].

[*] Corresponding author. Address: School of Sciences, Beijing University of Posts and Telecommunications, Beijing 100876, China.
E-mail addresses: zhaoxc@www.sci.bupt.cn, xczhao@mmrc.iss.ac.cn (X. Zhao), xgao@mmrc.iss.ac.cn (X.-S. Gao), huzc@nju.edu.cn (Z.-C. Hu).

0096-3003/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2006.06.107

In this paper, we focus on EP. In classical EP (CEP), each parent generates an offspring via Gaussian mutation, and better individuals among parents and offspring become the parents of the next generation. In [14], a "fast evolutionary programming (FEP)" using mutation based on Cauchy distribution was proposed. FEP converges faster to a better solution than CEP for a number of multimodal functions with many local minima [14]. We know that Cauchy probability distribution is a special case of Lévy probability distribution. Lee and Yao [15] generalized FEP and proposed an EP by using mutation based on Lévy probability distribution (LEP).

Yao and co-workers [14,15] claimed that "higher probability of making longer jumps" is a key point that makes FEP and LEP perform better than CEP in certain cases, and that "longer jumps" are detrimental if the current point is already very close to the global optimum. Inspired by this comment, we borrow non-uniform mutation operators from [10] in order to consider a new EP based on this kind of operators (NEP). Non-uniform mutation operators have the feature of searching the space uniformly at the early stage and very locally at the later stage.

Theoretical analysis on how NEP works is given based on probability theory. First, the expected step size of non-uniform mutation is calculated. Its derivative with respect to the generation variable t is less than zero, which implies the monotonously decreasing exploring region with the progress of the algorithm. Second, we obtain a quantitative description of the step size through analyzing the step size equation. Theoretical analysis also supports the fact that NEP is not sensitive to the search space size [21].

The rest of the paper is organized as follows: In Section 2, NEP algorithm is presented. In Section 3, comprehensive experiments are done to compare NEP with the recently proposed ILEP [15]. In Section 4, theoretical analysis on the executing process of NEP is given. An adaptive NEP algorithm is proposed in Section 5 and conclusions are reached in the last section.

2. Non-uniform evolutionary programming (NEP)

In this section, we introduce an evolutionary programming algorithm based on a non-uniform mutation operator.

2.1. Non-uniform mutation operator

Michalewicz proposed a dynamical non-uniform mutation operator [10] to reduce the disadvantage of random mutation in EAs. This operator is defined as follows.

For each individual X_i^t in a population of t-th generation, create an offspring X_i^{t+1} through a non-uniform mutation as follows: if $X_i^t = \{x_1, \dots, x_k, \dots, x_m\}$ is a chromosome (t is the generation number) and the element x_k is selected for this mutation, the result is a vector $X_i^{t+1} = \{x_1, \dots, x_k', \dots, x_m\}$, where

$$x_k' = \begin{cases} x_k + \Delta(t, \mathrm{UB} - x_k) & \text{if a random } \xi \text{ is } 0, \\ x_k - \Delta(t, x_k - \mathrm{LB}) & \text{if a random } \xi \text{ is } 1 \end{cases} \tag{1}$$

and LB and UB are the lower and upper bounds of the variable x_k, respectively. The function $\Delta(t,y)$ returns a value in the range $[0,y]$ such that $\Delta(t,y)$ approaches to zero as t increases. This property allows this operator to search the space uniformly at early stages (when t is small), and very locally at later stages. We use the following function:

$$\Delta(t,y) = y \cdot \left(1 - r^{\left(1-\frac{t}{T}\right)^b}\right), \tag{2}$$

where r is a uniform random number from $[0,1]$, T is the maximal generation number, and b is a system parameter determining the degree of non-uniformity (see Section 5.1).

2.2. Simple crossover in NEP

Let m be the dimension of a given problem and the components of chromosomes be float numbers. Choose a pair of individuals:

$$\mathbf{X} = (x_1, \ldots, \mathbf{x}_{\mathbf{pos_1}}, \ldots, \mathbf{x}_{\mathbf{pos_2}}, \ldots, x_m),$$
$$\mathbf{Y} = (y_1 \ldots, \mathbf{y}_{\mathbf{pos_1}}, \ldots, \mathbf{y}_{\mathbf{pos_2}}, \ldots, y_m),$$

and randomly generate a decimal r. If r is less than the crossover probability, two-point crossover is applied to them as follows: Generate two random integers pos_1, pos_2 in the interval $[1, m]$. The components of two individuals between the numbers pos_1 and pos_2 will be exchanged and thus the new individuals are generated as follows:

$$\mathbf{X}' = (x_1, \ldots, \mathbf{y}_{\mathbf{pos_1}}, \ldots, \mathbf{y}_{\mathbf{pos_2}}, \ldots, x_m),$$
$$\mathbf{Y}' = (y_1, \ldots, \mathbf{x}_{\mathbf{pos_1}}, \ldots, \mathbf{x}_{\mathbf{pos_2}}, \ldots, y_m).$$

2.3. NEP algorithm

For a function $f(\mathbf{X})$, NEP algorithm will find an

$$\mathbf{X}^* \text{ such that } f(\mathbf{X}^*) \leqslant f(\mathbf{X}) \quad \text{for all } \mathbf{X} \text{ in the searching space.} \tag{3}$$

The NEP algorithm adopts real encoding, two-point crossover and non-uniform mutation. The procedure of NEP is given as follows:

Procedure of NEP

(1) Generate the initial population consisting of n (without loss of generality, we assume that n is an even number) individuals, each of which $\mathbf{X_i^0}$, has m independent components, $\mathbf{X}_i^0 = (x_1, x_2, \ldots, x_m)$.
(2) Evaluate each individual based on the objective function, $f(\mathbf{X}_i^0)$.
(3) Apply the simple crossover to the current population.
(4) For each parental individual $\mathbf{X_i^t} = (x_1^t, \ldots, x_m^t)$, randomly generate a decimal r, if $r < pm$, then
  (4.1) for each component x_j^t construct $x_j^{t'}$ using Eqs. (1) and (2) (new individual is denoted by $\mathbf{X}_i^{t'}$)
  (4.1.1) If $f(\mathbf{X}_i^{t'}) \leqslant f(\mathbf{X}_i^t)$, $\mathbf{X}_i^{t'}$ replaces $\mathbf{X}_i^t$ as the current individual.
(5) Conduct n times pairwise comparison over the offspring population. For each comparison, q individuals are chosen uniformly at random from the offspring population. The fittest individual is put into the next generation.
(6) Repeat steps (3)–(5), until the stopping criteria are satisfied.

End of Procedure

In the procedure above, $pc \in [0, 1]$ is the crossover probability and $pm \in (0, 1]$ is the mutation probability. The parameter b in Eq. (2) remains unchanged during the evolution.

3. Experiments and analysis

In our algorithm we let the sum of the crossover probability (pc) and the mutation probability (pm) be **1** to ensure that NEP generates the same number of offspring compared with other EPs in terms of probabilistic sense. Precisely, we let pc be 0.4 and pm 0.6.[1] Following [10], we let the parameter b in Eq. (2) remain 5 throughout the evolution. To make the comparison fair, the population size of NEP is reduced proportionally according to the problem dimensions. For example, for a four dimensional function, if the population size of LEP is 100 then the population size of NEP will be 25.

The Cauchy probability distribution (for FEP [14]) is a special case of the Lévy probability distribution (for LEP [15]). The improved LEP(ILEP) performs at least as well as the non-adaptive LEP with a fixed parameter α [15]. Here, we just compare NEP with ILEP.

[1] We tried several combinations of pc and pm with $(pc,pm) = (0.7, 0.3)$, $(0.6, 0.4)$, $(0.5, 0.5)$, $(0.4, 0.6)$ and $(0.3, 0.7)$, and other parameters unchanged. The corresponding algorithm are comparable and we let $(pc,pm) = (0.4, 0.6)$ without any special reasons.

3.1. Benchmark functions and parameters setting

In the following we use 14 benchmarks as in ILEP [15]. Next we spell them out as three groups, where n is the dimension, D is the searching space and $f_{\min}$ means the minimum value of the function.

(i) *High-dimension unimodal functions*

Benchmark functions	n	D	$f_{\min}$
$f_1 = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]^n$	0
$f_2 = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$	30	$[-100, 100]^n$	0
$f_3 = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$	30	$(-30, 30)^n$	0

(ii) *High-dimension multimodal functions with many local minima*

Benchmark functions	n	D	$f_{\min}$
$f_4 = -\sum_{i=1}^{n} \left(x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)\right)$ $-12\,569.48$	30	$[-500, 500]^n$	
$f_5 = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	30	$[-5.12, 5.12]^5$	0
$f_6 = -20\exp\left[-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right]$ $-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	30	$[-32, 32]^n$	0
$f_7 = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	$[-600, 600]^n$	0
$f_8 = \frac{\pi}{n}\left\{10\sin^2(\pi y_i) + \sum_{i=1}^{n-1}(y_i - 1)^2 \cdot \left[1 + 10\sin^2(\pi y_{i+1})\right]\right.$ $\left. + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4), y_i = 1 + \frac{1}{4}(x_i + 1)$	30	$[-50, 50]^n$	0
$f_9 = 0.1\left\{10\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + 10\sin^2(3\pi x_{i+1})]\right.$ $\left. + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30	$[-50, 50]^n$	0

where in the definitions of benchmark functions f_8 and f_9,

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

(iii) *Low-dimension functions with only a few local minima.*

Benchmark functions	n	D	$f_{\min}$
$f_{10} = \left(4 - 2.1x_1^2 + \frac{1}{3}x_1^4\right)x_1^2 + x_1x_2 + \left(-4 + 4x_2^2\right)x_2^2$	2	$[-5, 5]^n$	-1.0316
$f_{11} = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$ $[30 + 2(x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	$[-2, 2]^n$	3
$f_{12} = -\sum_{j=1}^{5}\left[\sum_{i=1}^{4}(x_i - a_{ij})^2 + c_j\right]^{-1}$	4	$[0, 10]^n$	-10.1532
$f_{13} = -\sum_{j=1}^{7}\left[\sum_{i=1}^{4}(x_i - a_{ij})^2 + c_j\right]^{-1}$	4	$[0, 10]^n$	-10.40282
$f_{14} = -\sum_{j=1}^{10}\left[\sum_{i=1}^{4}(x_i - a_{ij})^2 + c_j\right]^{-1}$	4	$[0, 10]^n$	-10.53628

Some properties of several benchmarks are listed below (see [22]).

- f_3 is a continuous and unimodal function, with the optimum located in a steep parabolic valley with a flat bottom and the non-linear interactions between the variables, i.e., it is *non-separable* [23]. These features make the search direction have to continually change to reach the optimum. Experiments show that it is even more difficult than some multimodal benchmarks.
- The difficulty of f_2 concerns the fact that searching along the coordinate axes only gives a poor rate of convergence since its gradient is not oriented along the axes. It presents similar difficulties as f_3, but its valley is narrower.
- f_7 is difficult to optimize because it is non-separable and the search algorithm has to climb a hill to reach the next valley [23].

The maximal evolutionary generation numbers are the same as the adaptive LEP (1500 for functions $f_1,\ldots,f_9$, 30 for f_{10}, f_{11} and 100 for $f_{12},\ldots,f_{14}$). Due to the population size of the adaptive LEP being 100 and each individual generates four offsprings at every generation, the population size of the adaptive LEP is equivalent to 400. Consequently, we set the population size of NEP to be the largest integer less than $\frac{400}{\text{Dimensions of Problem}}$ which are 13 for functions $f_1,\ldots,f_9$, 200 for f_{10}, f_{11} and 100 for f_{12}, f_{13}, f_{14}. A computational precision of **50** digits after point is used in NEP. So a result *being 0* means that it is *less that* 10^{-50} in NEP and vice versa.

3.2. Performance comparison and analysis

Each result represented in Figs. 1–3 and Table 1 is the average for 50 independent runs.

The best and average fitness versus generations is shown in Figs. 1–3. Due to the small population size (i.e., 13) of high-dimensional functions, the plots of the average and best fitness nearly overlap each other.

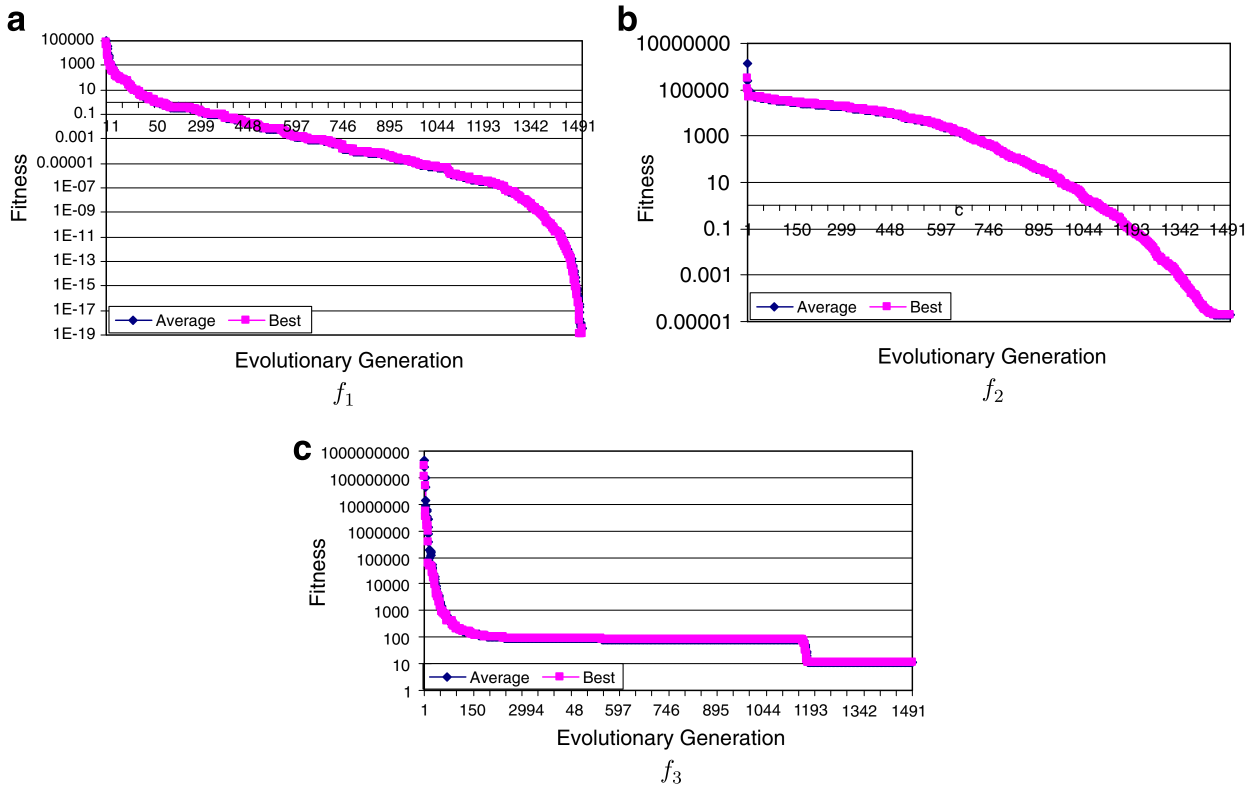
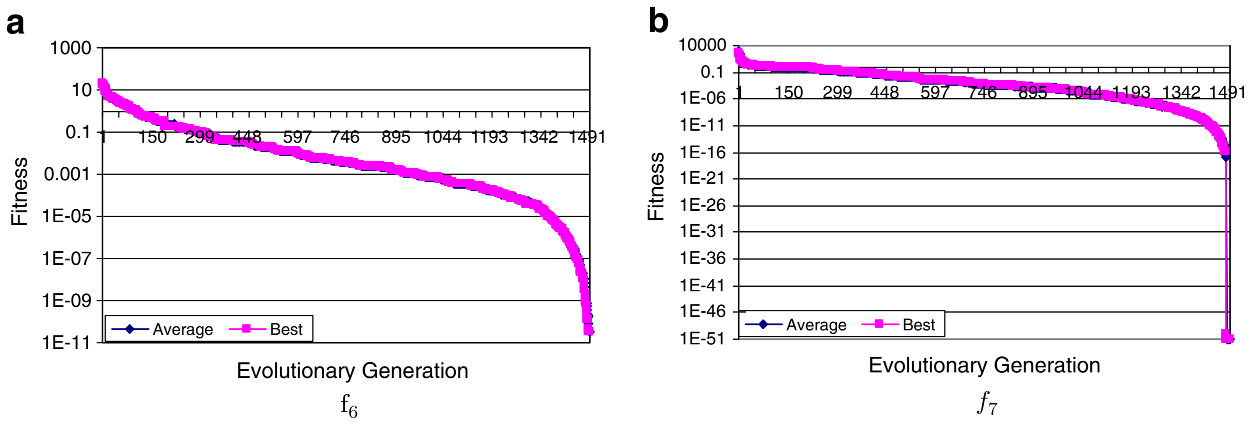


Fig. 1. Best and average fitness of unimodal functions versus evolutionary generation: (a) f_1; (b) f_2 and (c) f_3.



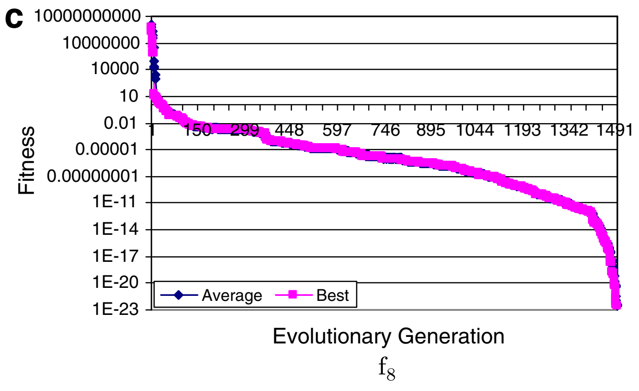


Fig. 2. Best and average fitness of multimodal functions versus evolutionary generation: (a) f_6; (b) f_7 and (c) f_8.

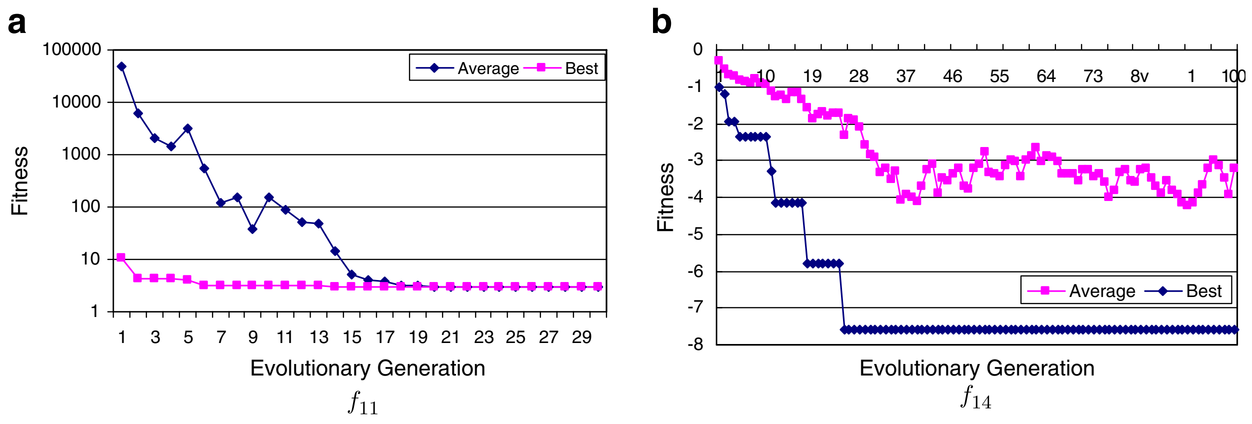


Fig. 3. Best and average fitness of low-dimensional functions versus evolutionary generation: (a) f_{11} and (b) f_{14}.

Fig. 2 shows that NEP can find the potential area quickly at the initial stage of the algorithm for high-dimensional multimodal functions. The results approach to 0.1–0.001 after about 200 generations. At the middle stage of the algorithm, a smooth behavior is indicated when it maybe patiently locates the position of the global optimum. A fast convergent speed appears at the later stage of the algorithm, which illuminates the *fine-tuning ability* of non-uniform mutation operator. For the function f_7, the algorithm reached the position of the minimal solution at about the last ten generations. These results roughly show that NEP has the merits of *long jumping (initial stage)* and *fine-tuning (later stage)* abilities. For unimodal functions, NEP shows similar behaviors as the multimodal functions observed from Fig. 1 except the function f_3.

Function f_3 is *non-separable* [23] whose result is the worst one compared with the results of other functions for NEP. Even though, it still outperforms ILEP in terms of solution quality and robustness (see Table 1).

Table 1
Performance comparison between NEP and ILEP

f	Population size		Mean Best		Std Dev	
	NEP	ILEP	NEP	ILEP	NEP	ILEP
1	13	100	3.66e−19	6.32e−4	1.65e−18	7.6e−5
2	13	100	1.62e−5	4.18e−2	8.34e−6	5.97e−2
3	13	100	1.35e1	4.34e1	7.91	3.15e1
4	13	100	−12 569.49	−11 469.2	5.15e−12	5.82e1
5	13	100	5.68e−14	5.85e0	0	2.07
6	13	100	1.07e−11	1.9e−2	2.93e−11	1.0e−3
7	13	100	9.94e−3	2.4e−2	9.39e−3	2.8e−2
8	13	100	2.02e−21	6.0e−6	3.86e−21	1.0e−6
9	13	100	1.93e−20	9.8e−5	4.25e−19	1.2e−5
10	200	100	−1.032	−1.031	2.37e−11	0.00
11	200	100	3.008	3.000	2.22e−2	0.000
12	100	100	−6.71	−9.54	2.68	1.69
13	100	100	−7.82	−10.30	2.76	0.74
14	100	100	−7.53	−10.54	2.77	4.9e−5

"Mean Best" indicates the average of the minimal values obtained at every run and "Std Dev" stands for the standard deviation.

Table 1 tells us that NEP is only outperformed by ILEP on functions $f_{12} - f_{14}$, which are low-dimensional multimodal functions with only a few local minima. As Schnier and Yao [24] analyzed that they are rather deceptive. For the high-dimension unimodal functions $f_1 - f_3$, and high-dimension multimodal functions with many local minima $f_4 - f_9$, NEP is better in terms of convergent ability and robustness.

It is worthy of noticing that NEP nearly finds the global optima for all the high-dimension multimodal functions except for the function f_7, the standard deviation of whose result just reaches 3 digits after the decimal point. The experimental results further confirm the difficulty of the function f_7. The standard deviation with respect to the function f_5 is equal to zero, that is, all the 50 runs reach the optimum.

4. Theoretical analysis on the executing process of NEP

In Section 3, experiments show that NEP is fast and robust for most benchmark functions tested. In this section, we try to give the underling reasons by conducting theoretical analysis.

Suppose that $X = \{x_1, \ldots, x_k, \ldots, x_m\}$ is a real-coded chromosome in the population and the component x_k (whose lower and upper bounds are L_k, U_k) is selected for variation through some mutation operation and x'_k is obtained. A decimal fraction r is uniformly and randomly generated in $[0, 1]$.

4.1. Preliminary analysis on NEP

As shown in [14,15], *Gaussian* mutation is much localized than *Cauchy* or *Lévy* mutation, and FEP [14] and LEP [15] have higher probabilities of making longer jumps than CEP. However, as they analyzed, "longer jumps" are not always beneficial. If the current search position is near to the small neighborhood of the global optimum, the "longer jumps" are detrimental.

For NEP, let ξ be a *Bernoulli* random variable in Eq. (1) with

$$P(\xi = 0) = P(\xi = 1) = \frac{1}{2}.$$

Let $\eta = -2\xi + 1$. Then η is a random variable with values in $\{1, -1\}$ and

$$P(\eta = 1) = P(\eta = -1) = \frac{1}{2}.$$

Thus, by Eqs. (1) and (2), we have

$$\begin{aligned} x_k' &= x_k + \frac{1+\eta}{2}\Delta(t, U_k - x_k) - \frac{1-\eta}{2}\Delta(t, x_k - L_k) \\ &= x_k + \frac{1+\eta}{2}(U_k - x_k)(1 - r^{(1-\frac{t}{T})^b}) + \frac{1-\eta}{2}(x_k - L_k)(1 - r^{(1-\frac{t}{T})^b}). \end{aligned} \tag{4}$$

Then the expected step size of the non-uniform mutation is

$$\begin{aligned} E|x_k' - x_k| &= P(\eta = 1)\cdot(U_k - x_k)\cdot E\left(1 - r^{\left(1-\frac{t}{T}\right)^b}\right) + P(\eta = -1)\cdot(x_k - L_k)\cdot E\left(1 - r^{\left(1-\frac{t}{T}\right)^b}\right) \\ &= \frac{U_k - L_k}{2}E\left(1 - r^{\left(1-\frac{t}{T}\right)^b}\right) = \frac{U_k - L_k}{2}\int_0^1 \left(1 - r^{\left(1-\frac{t}{T}\right)^b}\right) = \frac{U_k - L_k}{2}\left(1 - \frac{1}{1 + \left(1 - \frac{t}{T}\right)^b}\right). \end{aligned} \tag{5}$$

It is like neither the *Gaussian* mutation which locally searches nor the *Cauchy* or *Lévy* mutation which makes long jumps from the beginning to the end of algorithms. The expected step size of the non-uniform mutation in an interval depends on the generation t. Let $f(t) := E|x_k' - x_k|$, then we know that $f(t)$ is a decreasing function of t, because

$$\frac{\partial f(t)}{\partial t} = \frac{U_k - L_k}{2}\cdot\frac{\partial\left(1 - \frac{1}{1+(1-\frac{t}{T})^b}\right)}{\partial t} = -\left(\frac{U_k - L_k}{2}\cdot\frac{\frac{b}{T}\cdot(1-\frac{t}{T})^{b-1}}{(1 + (1-\frac{t}{T})^b)^2}\right) < 0. \tag{6}$$

Hence, the jumping size of the non-uniform mutation monotonously decreases with the progress of the algorithm. This roughly shows that the exploration region of NEP is smaller and smaller when t increases.

4.2. Detailed analysis on NEP

From Eq. (4), we have the following *jumping equation*:

$$x_k' = \begin{cases} x_k + (U_k - x_k)\cdot\left(1 - r^{\left(1-\frac{t}{T}\right)^b}\right) & \text{if } \eta = 1, \\ x_k - (x_k - L_k)\cdot\left(1 - r^{\left(1-\frac{t}{T}\right)^b}\right) & \text{if } \eta = -1. \end{cases} \tag{7}$$

Without loss of generality, we assume that x_k jumps to its right side and obtain x_k'. Then we have

$$x_k' = x_k + (U_k - x_k)\cdot\left(1 - r^{\left(1-\frac{t}{T}\right)^b}\right). \tag{8}$$

At the initial stage of the algorithm, i.e., $t \ll T$, $\frac{t}{T} \approx 0$, and it follows from Eq. (8) that (noting that b is a constant)

$$x_k' \approx x_k + (U_k - x_k)\cdot(1 - r). \tag{9}$$

So the algorithm nearly covers all the right side space regardless of the current position.

Similarly, if x_k jumps to its left side, we have that

$$x_k' \approx x_k - (x_k - L_k)\cdot(1 - r) \tag{10}$$

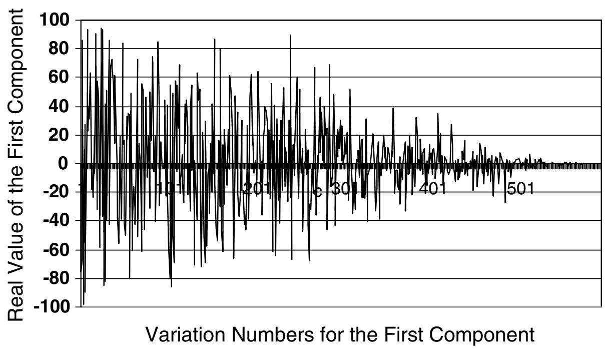
and the algorithm nearly covers all the left side space regardless of the current position.

In virtue of Eqs. (9) and (10), and the fact that r is a uniform random number in $[0, 1]$, we can clearly say that the search engine of NEP nearly explores the whole space at the initial stage of the algorithm.

If t is close to T (later stage of the algorithm), i.e., $\frac{t}{T} \approx 1$, then we have that

$$\left(1 - \frac{t}{T}\right)^b \approx 0 \;\Rightarrow\; r^{(1-\frac{t}{T})^b} \approx 1 \;\Rightarrow\; 1 - r^{(1-\frac{t}{T})^b} \approx 0,$$

regardless of r, where "$\Rightarrow$" means "implies". That is to say, when t is close to T, the term of $\left(1 - r^{\left(1-\frac{t}{T}\right)^b}\right)$ in Eq. (7) is "infinitely" approaching to 0 from the positive direction. Denote the small positive number $\left(1 - r^{\left(1-\frac{t}{T}\right)^b}\right)$ by $\epsilon(t)$, then



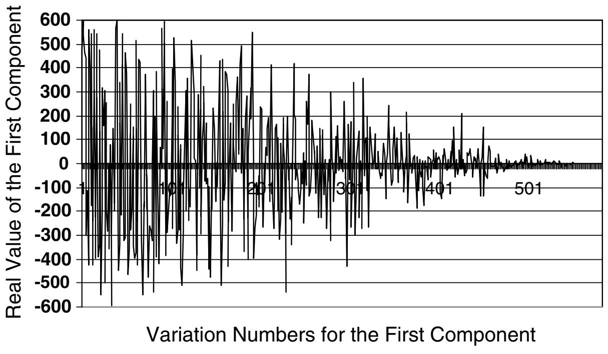


Fig. 4. Mutation numbers versus the real value for the first component of functions f_1 and f_7. Abscissa axis stands for the mutation numbers and vertical coordinates is the value of the first component value.

$$x'_k = \begin{cases} x_k + \epsilon(t) \cdot (U_k - x_k) & \text{if } \eta = 1, \\ x_k - \epsilon(t) \cdot (x_k - L_k) & \text{if } \eta = -1. \end{cases} \tag{11}$$

It follows from Eq. (11) that at later stage of the algorithm (i.e., t is close to T), the search engine of NEP exploits the neighborhood of the current solution very locally.

Now, we may say that the non-uniform mutation has the feature of searching the whole space initially and very locally at later stage of algorithms.

4.3. Experimental results

In order to illustrate our analysis about the feature of non-uniform mutation, we consider two functions f_1, f_7 (just investigate their first components) chosen from Section 3.1 to see how they are varying during the algorithm. Function f_1 is a typical unimodal sphere and f_7 is a typical multimodal benchmark. The aim of this group of experiment is not to consider the performance of algorithm, but to examine how the mutation step varies. We suppose that mutation probability $pm = 1$, crossover probability $pc = 0$, population size is 1, the maximal evolutionary generation is 600 and without greedy selection (see step (4.1.1) and step (5) in "Procedure of NEP" in Section 2.3).

Fig. 4 illustrates the real value for the first component of functions versus the mutation numbers. These figures clearly show the feature of non-uniform mutation. At the initial stage, variables "fly" to scan the whole search space. At the middle stage, the exploring space has a decreasing trend. However, it is *not* absolutely monotonic. Sometimes, NEP has another long jump leaving the current solution to find a new promising area. At the later stage, they just exploit the neighborhood of the global optima $\left(x_i^* = 0\right)$. This experiment strongly supports the theoretical analysis above about the feature of the non-uniform mutation.

5. An adaptive NEP

5.1. The Influence of the parameter b

Besides t in Eq. (5), there is another parameter b that we have been treating it as a constant. However, it is better to treat b as a parameter. In fact, b determines the degree of non-uniformity [10]. Next we will analyze this parameter to show how it affects the search step. For a given t in Eq. (5), let

$$g(b) := \frac{U_k - L_k}{2} \times \left(1 - \frac{1}{1 + (1 - \frac{t}{T})^b}\right) = \frac{U_k - L_k}{2} \times \frac{1}{1 + \left(1 - \frac{t}{T}\right)^{-b}},$$

then we have

$$\frac{\partial g(b)}{\partial b} = \frac{U_k - L_k}{2} \times \frac{-1}{(1 + (1 - \frac{t}{T})^{-b})^2} \times \left(1 - \frac{t}{T}\right)^{-b} \times \ln\left(1 - \frac{t}{T}\right) = \frac{U_k - L_k}{2} \times \frac{\ln(\frac{T}{T-t}) \times \left(1 - \frac{t}{T}\right)^{-b}}{\left(1 + \left(1 - \frac{t}{T}\right)^{-b}\right)^2} > 0. \tag{12}$$

Table 2
Comparison between ANEP, NEP and ILEP for the last three benchmarks

Benchmark	Mean Best			Std Dev		
	ILEP	NEP	ANEP	ILEP	NEP	ANEP
f_{12}	−9.54	−6.71	−7.29	1.69	2.68	2.86
f_{13}	−10.30	−7.82	−8.59	0.74	2.76	2.49
f_{14}	−10.54	−7.53	−8.76	4.9e−5	2.77	2.40

"Mean Best" indicates the average of the minimum values obtained at every run and "Std Dev" is the standard deviation.

It follows from Eq. (12) that the expected step size of the non-uniform mutation increases i.e., the average speed of decreasing of the step size becomes smaller when b increases. That is, different b means different non-uniformity. Based on this analysis, an adaptive NEP (ANEP) is proposed. It differs from NEP only in step (4) of the algorithm described in Section 2.3 In ANEP, three candidate offspring are generated with $b = 2, 5, 30$ from the same parent and select the best one as the surviving offspring. This scheme is adaptive because the value of b to be used is not predefined and is determined by the evolution.

5.2. Experiments and discussion

Only three benchmarks f_{12}, f_{13}, f_{14} from Section 3.1 were used to investigate the performance of ANEP because the performance of NEP is poor for these three benchmarks and good for others. Furthermore, there is no statistical difference between the performance of NEP and ANEP for other benchmarks (and so the results are omitted here). Since ANEP uses three different values for b, we let the population size of ANEP (67) be one third of NEP (200) and all the other parameters remain the same as in Section 3. The computing results are averaged over 50 trials and are listed in Table 2.

Although the performance of ANEP is improved based on the experimental results and the theoretical analysis on the expected search step, it is still poorer than ILEP [15] for these benchmarks. As "No Free Lunch Theorem" [25] states, there is no one optimization algorithm performs best for all problems.

By the way, ANEP finds the optimal minima about 25 times in 50 trials for these three benchmarks.

6. Conclusion

In this paper, a new EP algorithm is proposed based on a non-uniform mutation operator. Comparisons with the newly proposed ILEP based on Lévy probability mutation, NEP is faster and more robust for most multimodal benchmark functions tested.

Detailed theoretical analysis on its working schemes of NEP is presented. NEP searches the space uniformly at the early stage of the algorithm and very locally at the later stage. The probabilistic gradual decreasing jump length makes the algorithm exploring smaller and smaller regions with the progress of the algorithm. At the later stage, the algorithm just exploits the neighborhood of the current solution so as to exactly locate the global optimum.

Since the performance of NEP based on the mutation used in this paper is poorer than that of ILEP on some low-dimensional functions with a few minima, further research should be made in the future.

Acknowledgements

The first author thanks Prof. Xin Yao, Dr. Jun He, Jin Li, Xiaoli Li, Yong Xu and Dan Chen in the School of Computer Science, the University of Birmingham, UK, for their helps and suggestions. Partially supported by a National Key Basic Research Project of China and by a USA NSF grant CCR-0201253.

References

[1] A.S. Fraser, Simulation of genetic system by automatic digital computers. I. Introduction, Austral. J. Biol. Sci. 10 (1957) 484–491.

[2] H.J. Bremerman, Optimization through evolution and recombination, in: M.C. Yovits, G.T. Jacobi, G.D. Goldstine (Eds.), Self-organizing Systems, Spartan Books, Washington, DC, 1962, pp. 93–106.
[3] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, 1975.
[4] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.
[5] T. Kuo, S.Y. Huwang, A genetic algorithm with disruptive selection, IEEE Trans. Syst. Man Cybern. 26 (2) (1996) 299–307.
[6] H.P. Schwefel, Numerical Optimization of Computer Models, Wiley, New York, 1977.
[7] X. Yao, Y. Liu, Fast evolution strategies, Contr. Cybern. 26 (3) (1997) 467–496.
[8] L.J. Fogel, A.J. Owens, M.J. Walsh, Artificial Intelligence through Simulated Evolution, Wiley, New York, 1966.
[9] D.B. Fogel, Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, IEEE Press, New York, 1995.
[10] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, third ed., Springer, New York, 1996.
[11] D.B. Fogel, L.J. Fogel, J.W. Atmar, Meta-evolutionary Programming, in: R. Chen (Ed.), Proc. 25th Asilomer Conf. Sigals, Systems and Computers, 1991, pp. 540–545.
[12] D.B. Fogel, Evolving artificial intelligence, Ph.D. Dissertation, University of California, 1992.
[13] N. Saravanan, D.B. Fogel, Learning of strategy parameters in evolutionary programming: an empirical study, in: A. Sebald, L. Fogel (Eds.), Proc. 3rd Annual Conf. Evolutionary Programming, 1994, pp. 269–280.
[14] X. Yao, Y. Liu, G.M. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput. 3 (2) (1999) 82–102.
[15] C.Y. Lee, X. Yao, Evolutionary programming using mutations based on the levy probability distribution, IEEE Trans. Evol. Comput. 8 (1) (2004) 1–13.
[16] X.C. Zhao, A greedy genetic algorithm for unconstrained global optimization, J. Syst. Sci. Complex. 18 (1) (2005) 102–110.
[17] X.C. Zhao, H.L. Long, Multiple bit encoding-based search algorithms, in: Proc. 2005 IEEE Congress on Evolutionary Computation, IEEE Press, New York, 2005, pp. 1996–2001.
[18] H. Coskun, K. Fevzi, A heuristic approach for finding the global minimum: Adaptive random search technique, Appl. Math. Comput. 173 (2) (2006) 1323–1333.
[19] Y. Liang, L. Kwong-Sak, Evolution strategies with exclusion-based selection operators and a Fourier series auxiliary function, Appl. Math. Comput. 174 (2) (2006) 1080–1109.
[20] L. Wang, F. Tang, H. Wu, Hybrid genetic algorithm based on quantum computing for numerical optimization and parameter estimation, Appl. Math. Comput. 171 (2) (2005) 1141–1156.
[21] X.C. Zhao, X.S. Gao, A micro-evolutionary programming for optimization of continuous space, in: A. Tiwari, R. Roy (Eds.), PPSN VIII Workshop on Challenges in Real World Optimization using Evolutionary Computing, 2004, pp. 17–23.
[22] F. Herrera, M. Lozano, Gradual distributed real-coded genetic algorithms, IEEE Trans. Evol. Comput. 4 (1) (2000) 43–63.
[23] D. Whitley, S. Rana, J. Dzubera, E. Mathias, Evaluating evolutionary algorithms, Artif. Intell. 85 (1996) 245–276.
[24] T. Schnier, X. Yao, Using multiple representations in evolutionary algorithms, in: Proc. 2000 IEEE Congress on Evolutionary Computation, IEEE Press, New York, 2000, pp. 479–486.
[25] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82.