Available online at www.sciencedirect.com
ScienceDirect
Progress in Natural Science 18 (2008) 481–486
Progress in Natural Science
www.elsevier.com/locate/pnsc

Special report

An experimental study on the self-adaption mechanism used by evolutionary programming

Jingsong He [a,b,*]

[a] Department of Electronic Science and Technology, University of Science and Technology of China, Hefei 230027, China
[b] Nature Inspired Computation and Application Laboratory, University of Science and Technology of China, Hefei 230027, China

Received 29 October 2007; received in revised form 22 November 2007; accepted 23 November 2007

Abstract

Evolutionary programming (EP) is one of the most important methods for numerical optimization. Its main technique is the combination of mutations and the self-adaption mechanism. In the past years, studies on EP focused on how to improve the efficiency of mutations with different probability distributions, and few of them touched on the question that the self-adaption mechanism did not work sometimes, but simply followed the original suggestion. So far, no experimental results have shown why this is a question. This paper firstly gives a primary analysis on the behavior of the self-adaption mechanism, and then presents experimental evidences to show why its adaptive ability is doubtful.

Keywords: Evolutionary programming; Cauchy mutation; Self-adaption mechanism

1. Introduction

As a historical and veteran member of the evolutionary computation (EC) family, evolutionary programming (EP) [1,2] has been widely studied and used in the field of numerical optimization in the past years [3–12]. In general, optimization by evolutionary programming is mainly with two mechanisms: mutation and self-adaption [7,8], where mutations are usually generated by probability-based technique. According to the probability distribution used by EP, the classical approach of EP with Gaussian mutations was called CEP [7,9], and EP with Cauchy mutations was called fast evolutionary programming (FEP) [7,9]. Although the only difference between CEP and FEP is the form of the used probability distribution, this difference showed an inhomogeneous viewpoint on the search step size: Cauchy mutation is good at making longer jumps and thus performs significantly better than CEP on a number of multimodal problems, while Gaussian mutation is good at making relatively small jumps and thus performs better when individuals are near to the global optimum. It seems that Cauchy mutation and Gaussian mutation is a pair of paradoxical notion in evolution.

Although FEP outperforms CEP on many multimodal functions because of the advantage of long jumps generated by Cauchy mutation, it has the problem of generating relatively short jumps. One reason is because FEP usually uses an invariable scale parameter $t = 1$ in Cauchy mutation. This kind of setting becomes an obstacle for FEP to reach better performances on various kinds of problems: Yao et al. [7] carried out a series of empirical and theoretical studies, and found that the optimal value of the scale parameter t in the Cauchy mutation of FEP is problem dependent, but is extremely difficult to find the optimal t for a given problem. In order to reduce the impact of the setting of t-parameter, the improved FEP (i.e. IFEP) [7]

* Address: Department of Electronic Science and Technology, University of Science and Technology of China, Hefei 230027, China.
E-mail address: hjss@ustc.edu.cn

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.2007.11.013

was proposed, in which the Gaussian and Cauchy mutations were mixed thus the problem of how to deal with the setting of t-parameter in Cauchy mutation was avoided. However, the t-parameter problem remains of significant value, because it is not an independent problem in Cauchy mutation. This directs to a crucial question about the capability of the self-adaption mechanism.

In essence, the key of CEP is "Gaussian mutation + self-adaption mechanism", and the key of FEP is "Cauchy mutation + self-adaption mechanism". It is apparent that if the self-adaption mechanism was versatile, it should have the ability of adapting out any size of strategy parameter that EP needed. Based on this assumption, few different results between the optimization performance of CEP and FEP as well as the results between FEP using different t-parameters exist. Unfortunately, the fact is on the contrary: the self-adaption mechanism can neither adapt out long jumps to help CEP in escaping local minima, nor adapt out small jumps to help FEP in doing local search. This fact implies that the self-adaption may be disabled. For future development, it is important to understand the capability of the self-adaption.

To have some primary cognitions in this paper, we firstly give a statistical analysis on the behavior of the self-adaption mechanism. And then, we view the behavior of what role the t-parameter in Cauchy mutation plays together with the self-adaption mechanism. Following a set of experimental studies further, we find that the self-adaption mechanism is doubtful.

2. A brief description of evolutionary programming

In evolutionary programming (EP), each individual is taken as a pair of real-valued vector $(\boldsymbol{x}_i, \eta_i)$, $\forall i = 1, 2, \ldots, \mu$, where $\boldsymbol{x}_i$'s are objective vectors variables and η_i's are standard deviations for mutations. For each current individual (X_i, η_i), the generating method and the self-adaption equation are [7,8]

$$X_i'(j) = X_i(j) + \eta_i(j) N_j(0,1) \tag{1}$$

$$\eta_i'(j) = \eta_i(j) \exp[\tau' N(0,1) + \tau N_j(0,1)] \tag{2}$$

where $X_i(j)$, $X_i'(j)$, $\eta_i(j)$, and $\eta_i'(j)$ denote the jth component of the vectors $\boldsymbol{x}_i$, $\boldsymbol{x}_i'$, η_i, and η_i', respectively; $N(0,1)$ denotes a normally distributed one-dimensional random number with mean zero and standard deviation one; $N_j(0,1)$ indicates that the random number is generated anew for each value of j. The factors τ and τ' are commonly set to be $(2n^{0.5})^{-0.5}$ and $(2n)^{-0.5}$ [7,8], respectively. Gehlhaar and Fogel's study [6] showed that swapping the order of (1) and (2) may improve EP's performance.

Different from the classical evolutionary programming (CEP) which uses (1) and (2) above, the idea of FEP is to introduce Cauchy mutation into (1) to substitute Gaussian mutation. The one-dimensional Cauchy density function centered at origin is defined as

$$f_t(x) = \frac{1}{\pi} \frac{t}{t^2 + x^2}, \quad -\infty < t < +\infty \tag{3}$$

where $t > 0$ is a scale parameter. The update equation of (1) is replaced by

$$X_i'(j) = X_i(j) + \eta_i(j)\delta_j \tag{4}$$

where δ_j is a Cauchy random variable with the scale parameter $t = 1$ and is generated anew for each j. Analytical results [7] showed that Gaussian mutation is more suitable for a localized search, while Cauchy mutation is good at making longer jumps for escaping local minima, thus suitable for global search.

3. Analysis on the behavior of the self-adaption mechanism

According to the self-adaption mechanism described in Section 2, it is clear that whatever the individuals do, and whether an individual is survived are decided by a certain selection mechanism, and those initial standard deviations, i.e. $(\eta_i i \in \{1, 2, \ldots, \mu\})$, are always changed by the self-adaption mechanism and succeeded as an attached label with those survived offsprings surviving to the end of the evolutionary procedure. Mathematically, a serial states of $\eta_i(t)$ from the start to the end of EP exist at least, where t denotes the time step of evolution, and $t = 0$ denotes its initial value. Thereby, the iterative expression of η_i takes

$$\eta_i(t+1) = \eta_i(t) \exp[\tau'\alpha(t) + \tau\beta(t)] \tag{5}$$

where $\alpha(t)$ and $\beta(t)$ denote Gaussian random number as $N(0,1)$ and $N_j(0,1)$ generated at the t time step respectively. Hence,

$$\begin{aligned}\eta_i(k) &= \eta_i(1) \prod_{t=1}^{k} \exp\left[\tau'\alpha(t) + \tau\beta(t)\right] \\ &= \eta_i(1) \exp\left[\sum_{t=1}^{k} \tau'\alpha(t) + \tau\beta(t)\right] \\ &= \eta_i(1) \exp\left[\tau' \sum_{t=1}^{k} \alpha(t) + \tau \sum_{t=1}^{k} \beta(t)\right]\end{aligned}$$

Since $\alpha(t)$ and $\beta(t)$ are Gaussian random numbers with mean zero and standard deviation one, thereby

$$\lim_{k\to\infty} \sum_{k=1}^{\infty} \alpha(t) = 0, \quad \lim_{k\to\infty} \sum_{k=1}^{\infty} \beta(t) = 0$$

Thus

$$\lim_{k\to\infty} \eta_i(k) = \eta_i(1)$$

The above analytical result implies that the self-adaption mechanism of Eq. (2) is sensitive to the initial value of η: starting from the initial strategy parameter, and ending at the same value. At each time step, the value of $\eta(t)$ can be thought as a newly started point, and then the self-adaption mechanism will be repeated later. Thus, it is possible to observe the behavior of the self-adaption mechanism with different mutations. It is apparent that simulation

experiment are useful to find the possible fault of the self-adaption mechanism, because only one evidence is qualified enough.

First, we observe the capability of the self-adaption mechanism with the Gaussian mutations. Two illustrative records are shown in Fig. 1, where the initial value of η is set to be 3 according to the suggestion of Ref. [7]. We can see that the shott-time frequency of the change of η is relatively low. That is, the self-adaption mechanism does not ensure of generating the expected values for Gaussian mutation in time.

The second simulation experiment is to observe the behavior of the self-adaption mechanism with Cauchy mutation. Since a Cauchy random number δ can be represented as $\delta = (\delta/\alpha) \cdot \alpha$ and α is a Gaussian random number, denote $\psi = \delta/\alpha$, then the one-dimensional form of Eq. (3) can be written as

$$X'_i(j) = X_i(j) + \psi_j \cdot \eta_i(j) \cdot \alpha_j \tag{6}$$

where Gaussian random number α_j is generated independently anew for each j. Thus, Cauchy mutation and Gaussian mutation have the similar form in expression: when $\psi \equiv 1$, Eq. (6) denotes one-dimensional Gaussian mutation, and when $\psi_j = \delta_j/\alpha_j$, Eq. (6) denotes one-dimensional Cauchy mutation. That is, ψ can be thought as the weight of η. Therefore, observation on the changed state of $\psi \cdot \eta(t)$ can be used to distinguish the difference between the self-adaption mechanism with Cauchy and Gaussian mutations.

Two illustrative records are given in Fig. 2, where the initial value of η is the same as the former experimental observation. It is apparent that the short-time frequency of the change of Cauchy mutations is relatively higher than that of Gaussian mutations. By the comparative results of Ref. [7] between FEP and CEP, it seems that the short-time frequency of the change of η affects the optimization performance directly, because the difference between FEP and CEP is only in ψ: CEP takes $\psi = 1$, while FEP takes $\psi_j = \delta_j/\alpha_j$, where α_j denotes a Gaussian random number generated for the jth variance.

Both the update equations of mutations used by CEP and FEP, and the above simulation experiments showed that the only difference between Gaussian mutation and Cauchy mutation is the focus on the weight ψ. From this point of view, the reason why FEP outperforms CEP on many benchmark functions is reasonable: the self-adaption mechanism cannot work rapidly, while Cauchy mutations, which can be thought as a weighted Gaussian mutation, is helpful in making the change of the strategy parameter much faster. Therefore, the self-adaption mechanism is not perfect, and Cauchy mutation improves the performance of the self-adaption mechanism indirectly.

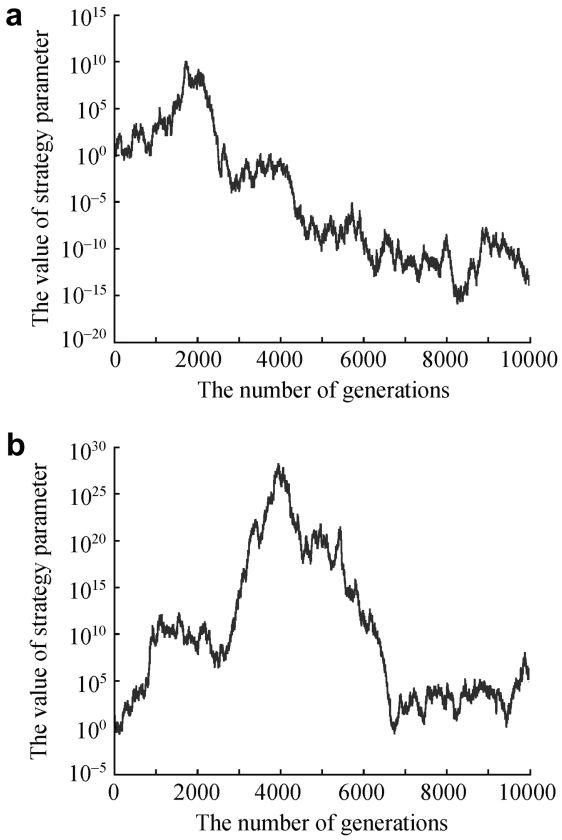

Fig. 1. Two examples (a) and (b) of the simulated results of self-adaption for Gaussian mutations with the initial value of $\eta(1) = 3$ and the size of dimension of 30. All the simulated results have the same characteristic of the short-time frequency.

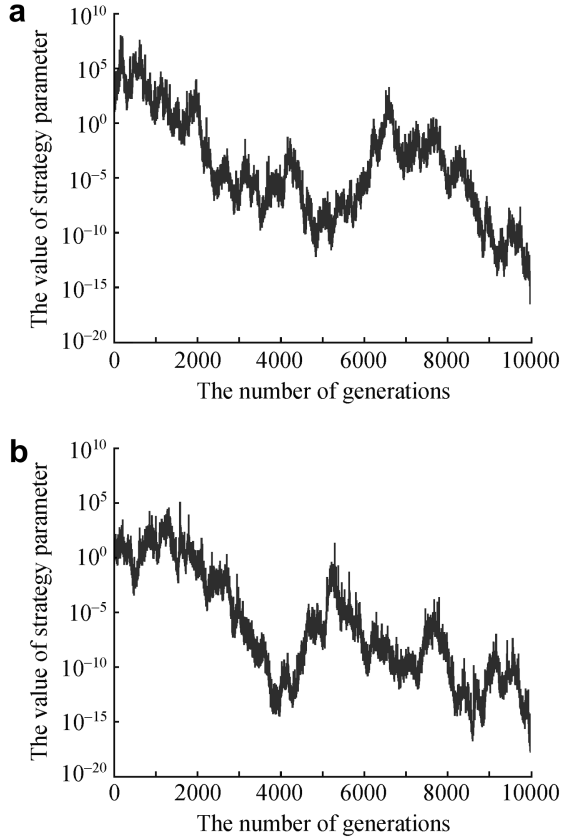

Fig. 2. Two examples (a) and (b) of the simulated results of self-adaption for Cauchy mutations with the initial value of $\eta(1) = 3$ and the size of dimension of 30. All the simulated results have the same characteristic of the short-time frequency.

4. Experimental evidences

Although Cauchy mutation can make EP faster than Gaussian mutation does, it was reported [7] that the setting of t-parameter of Cauchy mutation affects FEP's performance seriously, and it seems that there is no orderliness to set the t-parameter efficiently. According to the viewpoint and analytical results in Section 3, the ability of the self-adaption mechanism is doubtful still: is it because that the self-adaption mechanism does not always work properly?

It can be imaged easily that if the t-parameter problem in Cauchy mutation is mainly related to the ability of the self-adaption mechanism, it will be very difficult to find its setting rule, because the self-adaption mechanism does not expect outputting a serial of ordered strategy parameters in nature. In order to have an investigation about this question, here we revise Eq. (4) (i.e. the Cauchy mutation) as

$$X_i'(j) = X_i(j) + F \cdot \eta_i(j) \cdot \delta_j \tag{7}$$

where F is a uniform random number bounded in $(0, 1]$. The meaning of this kind of revision are as follows.

(i) The scaling factor F, as the weight of δ_j, can be regarded as the result of different setting of t-parameter of Cauchy distribution.
(ii) Since FEP uses an invariable scale parameter $t = 1$ in Cauchy mutation, and the averaged search step size of the Cauchy mutations is longer than that of the Gaussian mutations, the effect of scaling factor F, which is bounded in $(0, 1)$, is to randomly make Cauchy jumps relatively small.
(iii) If the introducing of the scaling factor F can make improvements for FEP, it means that the self-adaption mechanism at least losses some ability that the scaling factor F takes.

For simplicity, here, the FEP using the random weighted Cauchy mutation Eq. (7) is called the weighted FEP (WFEP).

No free lunch theorems [13] have shown that under certain assumptions no single search algorithms is best on average for all problems. Surely, here we do not expect that the random set of the scaling factor F performs better on all problems than FEP does using the constant value $F = 1$. Our purpose is to verify a question that if WFEP outperforms FEP on a number of benchmark functions significantly. We denote that the self-adaption mechanism still has some faults with FEP.

The benchmark functions for experiments are listed in the Appendix, which are used to confirm the guess above. f_1 and f_2 are unimodal functions, f_3–f_6 are multimodal functions with many local minima, and f_7–f_9 are multimodal functions with a few local minima. In the experiments, the population size $\mu = 100$, the initial $\eta = 3$, and the opponents $q = 10$ for WFEP and FEP are the same. Experimental results are summarized in Table 1, where all the results are averaged on 50 runs. We can see that WFEP performs significantly better than FEP on these problems. The benefits made by the introduced factor F show that the demerits of FEP comes mainly from the self-adaption mechanism.

More detailed recorders of the averaged optimization performance on f_1, f_3, and f_5 are shown in Figs. 3–5. In the early period of evolution, the self-adaption mechanism cannot make strategy parameters relatively large for WFEP, thus WFEP shows relatively slow convergence speed than FEP. While in the later time of evolution, the self-adaption mechanism cannot make strategy parameters

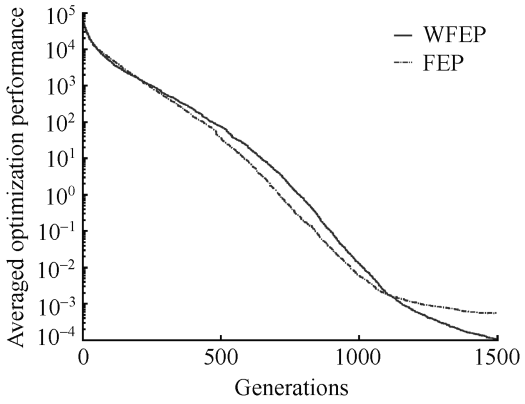

Fig. 3. Averaged optimization performance of WFEP and FEP ($t = 1$) on benchmark function f_1.

Table 1
Comparison between the standard FEP and the random weighted FEP (WFEP)

Function	Gen	WFEP mean best	FEP mean best	WFEP-FEP t-test[a]
f_1	1500	1.03e−4	5.7e−4	−24.625
f_2	5000	5.10e−3	0.3	−4.1704
f_3	1500	5.80e−3	1.8e−2	−34.928
f_4	1500	2.87e−6	9.2e−6	−10.517
f_5	1500	3.36e−5	1.6e−4	−10.517
f_6	4000	3.075e−4	5.0e−4	−4.2537
f_7	100	−9.34	−5.52	−11.297
f_8	100	−9.11	−5.52	−7.2610
f_9	100	−10.53	−6.57	−8.9176

[a] All results have been averaged over 50 runs, where the value of t with 49 degrees of freedom is significant at $\alpha = 0.05$ by a two-tailed test.

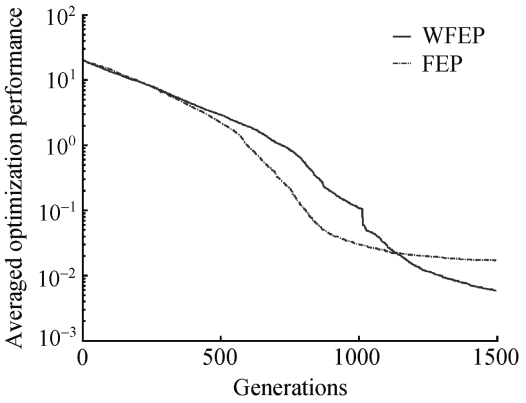

Fig. 4. Averaged optimization performance of WFEP and FEP ($t = 1$) on benchmark function f_3.

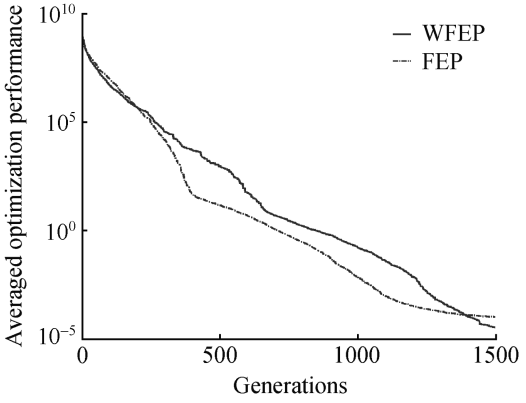

Fig. 5. Averaged optimization performance of WFEP and FEP ($t = 1$) on benchmark function f_5.

relatively small for FEP, thus FEP shows relatively slow convergence speed than WFEP. That is, the self-adaption mechanism cannot provide an appropriate strategy parameter at an appropriate time.

5. Conclusions

Evolutionary programming is one of the most important methods in the field of evolutionary computation. There are a number of traditional and new applications superior to EP's performance [3–5,14–16], and some new innovations of other optimization algorithms [17,18], such as differential evolution (DE) [19] and particle swarm optimization (PSO) [20], are also stimulated by EP's idea, specially by the FEP which uses Cauchy mutation.

Most studies on EP in the past years focused on how to do mutation with different probability distributions. However, few attention was paid on the self-adaption mechanism. This paper presents a primary investigation in experiments.

The analytical results can be summarized briefly as follows: The search step size is important to EP's performance, while the ability of the self-adaption mechanism is doubtful. Different from the explanation in Ref. [7], which cannot explain why the t-parameter in Cauchy mutation is a problem, this paper presents another new interpretation of the self-adaption mechanism. This result may be useful to improve EP's performance in the future.

Acknowledgement

This work was supported by the National Natural Science Foundation of China under Grant No. 60573170.

Appendix. Benchmark functions

Sphere model:

$$f_1(x) = \sum_{i=1}^{30} x_i^2, \quad -100 \leqslant x_i \leqslant 100, \quad \min(f_1(x)) = 0$$

Schwefel's problem 2.21:

$$f_2(x) = \max_i\{|x_i|, 1 \leqslant i \leqslant 30\}, \quad -100 \leqslant x_i \leqslant 100, \quad \min(f_2(x)) = 0$$

Ackley's function:

$$f_3(x) = -20\exp\left(-0.2\sqrt{\frac{1}{30}\sum_{i=1}^{30} x_i^2}\right) - \exp\left(\frac{1}{30}\sum_{i=1}^{30}\cos 2\pi x_i\right) + 20 + e, \quad -32 \leqslant x_i \leqslant 32, \quad \min(f_3(x)) = 0$$

Generalized penalized function:

$$f_4(x) = \frac{\pi}{30}\{10\sin^2(\pi y_1) + \sum_{i=1}^{29}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{30} u(x_i, 10, 100, 4), \quad -50 \leqslant x_i \leqslant 50, \quad \min(f_4(x)) = 0$$

$$f_5(x) = 0.1\{10\sin^2(\pi 3x_1) + \sum_{i=1}^{29}(x_i - 1)^2[1 + 10\sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + \sin^2(2\pi x_{30})]\} + \sum_{i=1}^{30} u(x_i, 5, 100, 4), \quad -50 \leqslant x_i \leqslant 50, \quad \min(f_5(x)) = 0$$

where

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

$$y_i = 1 + \frac{1}{4}(x_i + 1)$$

Table 2
Kowalik's function

i	a_i	b_i^{-1}
1	0.1957	0.25
2	0.1947	0.5
3	0.1735	1
4	0.1600	2
5	0.0844	4
6	0.0627	6
7	0.0456	8
8	0.0342	10
9	0.0323	12
10	0.0235	14
11	0.0246	16

Table 3
Shekel function

i	a_{i1}	a_{i2}	a_{i3}	a_{i4}	c_i
1	4	4	4	4	0.1
2	1	1	1	1	0.2
3	8	8	8	8	0.2
4	6	6	6	6	0.4
5	3	7	3	7	0.4
6	2	9	2	9	0.6
7	5	5	3	3	0.3
8	8	1	8	1	0.7
9	6	2	6	2	0.5
10	7	3.6	7	3.6	0.5

Kowalik's function:

$$f_6(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right],$$
$$-5 \leqslant x_i \leqslant 5, \quad \min(f_6(x)) \approx 0.0003075$$

The coefficients are defined in Table 2.

Shekel's family:

$$f(x) = -\sum_{i=1}^{m}[(x - a_i)(x - a_i)^T + c_i]^{-1}$$

with $m = 5$, 7, and 10 for f_7, f_8, and f_9, respectively, $0 \leqslant x_j \leqslant 10$. These functions have five, seven, and ten local minima for f_7, f_8, and f_9, respectively. $x_{\text{local\_opt}} = 1/c_i$ for $1 \leqslant i \leqslant m$. The coefficients are defined in Table 3.

References

[1] Fogel LJ, Owens AJ, Walsh MJ. Artificial intelligence through simulated evolution. New York: Wiley; 1966.

[2] Eiben AE, Smith JE. Introduction to evolutionary computing. Berlin: Springer; 2003.

[3] Fogel DB. System identification through simulated evolution: a machine learning approach to modeling. Needham Heights, MA: Ginn; 1991.

[4] Fogel DB. Evolving artificial intelligence. Ph.D. dissertation, University of California, San Diego, CA, 1992.

[5] Fogel DB. Applying evolutionary programming to selected traveling salesman problems. Cybern Syst 1993;24:27–36.

[6] Gehlharr DK, Fogel DB. Tuning evolutionary programming for conformationally flexible molecular docking. In: Evolutionary programming V: proc. of the fifth annual conference on evolutionary programming. Cambridge, MA: MIT Press; 1996. p. 419–29.

[7] Yao X, Liu Y, Lin G. Evolutionary programming made faster. IEEE Trans Evol Comput 1999;2(2):82–102.

[8] Bäck T, Schwefel HP. An overview of evolutionary algorithms for parameter optimization. Evol Comput 1993;1(1):1–23.

[9] Lee CY, Yao X. Evolutionary programming using the mutations based on Lévy probability distribution. IEEE Trans Evol Comput 2004;8(5):456–70.

[10] Schnier T, Yao X. Using multiple representations in evolutionary algorithms. In: Proc. of the 5th international conference on evolvable systems: from biology to hardware. Lecture notes in computer science, vol. 2606. Berlin: Springer; 2003. p. 35–46.

[11] Liu Y. Operator adaption in evolutionary programming. In: Advances in computation and intelligence. Lecture notes in computer science, vol. 4683. Berlin: Springer; 2007. p. 90–9.

[12] He JS, Yang ZY, Yao X. Hybridisation of evolutionary programming and machine learning with k-nearest neighbor estimation. In: 2007 IEEE Congress on Evolutionary Computation (CEC2007). Singapore, September 25–28, 2007. p. 1693–700.

[13] Wolpert DH, Macready WG. No free lunch theorems for search. IEEE Trans Evol Comput 1997;1:67–82.

[14] Li B, Lin J, Yao X. A novel evolutionary algorithm for determining unified creep damage constitutive equations. Int J Mech Sci 2002;44:987–1002.

[15] Tang J, He JS, Huang L. A new optimization engine for the LSF vector quantization. In: Eighth ACIS international conference on software engineering, artificial intelligence, networking, and parallel/distributed computing, vol. 3. QingDao, China, July 30–Aug. 1, 2007. p. 466–71.

[16] He JS, Yao X, Tang J. Towards intrinsic evolvable hardware for predictive lossless image compression. In: Simulated evolution and learning. Lecture notes in computer science, vol. 4247. Berlin: Springer; 2006. p. 632–9.

[17] He JS, Yang ZY, Yao X. Hybridisation of particle swarm optimization and fast evolutionary programming. In: Simulated evolution and learning. Lecture notes in computer science, vol. 4247. Berlin: Springer; 2006. p. 392–9.

[18] Yang ZY, He JS, Yao X. Making a difference to differential evolution. In: Advances in metaheuristics for hard optimization. Berlin: Springer; 2007. p. 415–32.

[19] Storn R, Price K. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 1997;11:341–59.

[20] Kennedy J, Eberhart RC. Particle swarm optimization. In: proceedings of the 1995 IEEE international conference on neural networks, vol. 4. Perth, Australia, December 1995. p. 1942–48.