Evolutionary Programming Using Mutations Based on the Lévy Probability Distribution

Chang-Yong Lee, and Xin Yao, *Fellow, IEEE*

***Abstract*—This paper studies evolutionary programming with mutations based on the Lévy probability distribution. The Lévy probability distribution has an infinite second moment and is, therefore, more likely to generate an offspring that is farther away from its parent than the commonly employed Gaussian mutation. Such likelihood depends on a parameter α in the Lévy distribution. We propose an evolutionary programming algorithm using adaptive as well as nonadaptive Lévy mutations. The proposed algorithm was applied to multivariate functional optimization. Empirical evidence shows that, in the case of functions having many local optima, the performance of the proposed algorithm was better than that of classical evolutionary programming using Gaussian mutation.**

***Index Terms*—Evolutionary optimization, evolutionary programming, Lévy mutation, Lévy probability distribution, mean-square displacement.**

I. Introduction

CONSIDERABLE work has been devoted to computational methods that are inspired by biological evolution and natural selection. Among them, genetic algorithms (GAs), evolutionary programming (EP), and evolution strategies (ESs) are most prominent. These methods have drawn much attention to the research community in conjunction with the parallel and/or distributed computations. EP [1], in particular, was studied initially as a method for generating artificial intelligence [2].

Even though EP first used finite-state machines as the underlying structure to be evolved [2], it was extended in the mid-1980s to handle essentially arbitrary data structures [1], such as applications involving continuous parameters and combinatorial optimization problems. Since then, EP has been applied successfully for the optimization of real-valued functions [3], [4] and other practical problems [5]. The more elaborate versions of EP incorporating self-adaptation of variances of mutations were introduced in early 1990s [6], [7]. The self-adaptation rules from ESs [9] have also been incorporated into EP [8]. Self-adaptive EP for continuous parameter optimization problems is now a widely accepted method in evolutionary computation. It should be noted that EP for continuous parameter optimization has many similarities with ESs, although their development proceeded independently. In this paper, we focus on EP for continuous parameter optimization, although many of our conclusions are also applicable to other evolutionary algorithms.

There are three major operations in a generic evolutionary algorithm: crossover, selection, and mutation. In contrast to GAs that emphasize crossover, mutation is the main operation in EP. In conventional EP, each parent generates an offspring via Gaussian mutation and better individuals among parents and offspring are selected as parents of the next generation. Schematically, Gaussian mutation in EP is accomplished by adding zero-mean Gaussian random numbers to parents to obtain offspring.

A study on non-Gaussian mutation in EP was initially carried out in mid-1990s [11]. A mutation operation based on the Cauchy distribution was proposed as a "fast evolutionary programming (FEP)" [11], [20]. FEP converges faster to a better solution than conventional EP for many multivariate functions [20]. Unlike the Gaussian probability distribution, the Cauchy probability distribution has *infinite* second moment and, as a result, has a much longer tail. Therefore, Cauchy mutated offspring can be quite different from their parents. It was shown analytically that FEP has some advantages over conventional EP [20].

In this paper, we generalize FEP further by using mutation based on the Lévy probability distribution. With the Lévy probability distribution, one can extend and generalize FEP because the Cauchy probability distribution is a special case of the Lévy probability distribution. The Lévy probability distribution differs from the Gaussian distribution in that the Lévy distribution, like the Cauchy distribution, has an infinite second moment. One of the characteristics of the Lévy distribution is its power law in the tail region. The power law implies that there is no characteristic length scale and this is the milestone of fractal structure. The distribution has drawn much attention from scientific communities in explaining the fractal structure of nature, in areas such as turbulence, polymers, biological systems, and even economics [12]. Moreover, by adjusting the parameter α in the distribution, one can tune the probability density, which in turn yields adjustable variation in the mutation.

The rest of the paper is organized as follows. In Section II, we discuss the characteristics of the Lévy probability distribution and the algorithm for generating random variables having the Lévy probability distribution. In Section III, we analyze mutation operations based on the two different distribution in terms of the mean-square displacement and the number of distinct values obtained by mutation operations. Section IV describes the multidimensional function optimization problems

Manuscript received December 2002; revised May 6, 2003 and June 5, 2003. This work was supported in part by Kongju National University, Korea, and the Royal Society, U.K., 2000–2001.

C.-Y. Lee is with the Department of Industrial Information, Kongju National University, Chungnam 340-802, Korea (e-mail: clee@mail.kongju.ac.kr).

X. Yao is with the School of Computer Science, The University of Birmingham, Edgbaston, Birmingham B15 2TT, U.K. (e-mail: x.yao@cs.bham.ac.uk).

Digital Object Identifier 10.1109/TEVC.2003.816583

using both Gaussian and Lévy mutation operations in EP. This is followed by results and discussions of the optimization experiments for both mutation operations in Section V. The adaptive Lévy mutation in EP is discussed in Section VI. The last section, Section VII, is devoted to conclusions and future studies.

II. Lévy Probability Distribution

As a finite second moment would lead to a characteristic scale and the Gaussian behavior for the probability density through the central limit theorem, P. Lévy, in the 1930s, searched for an exception to it. He discovered a class of probability distributions having an *infinite* second moment and governing the sum of these random variables [13], [22]. An infinite second moment implies the absence of a definite scale, which is the paradigm of fractals.

Consider a process represented by a set $\{Y_i\}$ of identically distributed random variables. If the sum of these random variables has the same probability distribution as individual random variables, then this process is called *stable*. A typical example of the stable process is the Gaussian process. While the Gaussian process has a finite second moment, there is a class of probability distributions of an infinite second moment that also yields a stable process. Such a probability distribution is called the Lévy probability distribution and has the following form [13], [22]:

$$L_{\alpha,\gamma}(y) = \frac{1}{\pi}\int_0^\infty e^{-\gamma q^\alpha}\cos(qy)dq, \qquad y \in R. \tag{1}$$

The distribution is symmetric with respect to $y = 0$ and has two parameters γ and α. γ is the scaling factor satisfying $\gamma > 0$, and α controls the shape of the distribution, requiring $0 < \alpha < 2$. The analytic form of the integral is unknown for general α, but is known for a few cases. In particular, for $\alpha = 1$, the integral can be reduced to the analytic form resulting in the Cauchy distribution. In the limit of $\alpha \rightarrow 2$, the distribution is no longer the Lévy distribution and becomes the Gaussian distribution. The parameter α controls the shape of the distribution in such a way that one can obtain different shapes of probability distribution, especially in the tail region: the smaller is the parameter α, the longer the tail. Fig. 1 compares the shapes of the Lévy distributions of $\alpha = 1.0$ and $\alpha = 1.5$ with the standard Gaussian distribution.

The scaling factor γ can be set to $\gamma = 1$ without loss of generality. To see this, rescale y to $y' = by$ with some constant b. Then, from (1), we get the following relation:

$$L_{\alpha,\gamma}(by) = \frac{1}{b}L_{\alpha,\gamma'}(y) \tag{2}$$

where $\gamma' = \gamma b^{-\alpha}$. In particular, by setting $\gamma' = 1$, the above equation becomes

$$L_{\alpha,\gamma}(y') = \gamma^{\frac{-1}{\alpha}}L_{\alpha,1}(y) \tag{3}$$

implying that γ is nothing but an overall scaling factor. Thus, with the distribution of $\gamma = 1$, we can get the distribution of any other γ. In this study, we fix $\gamma = 1$ and denote $L_{\alpha,1} = L_\alpha$. For large values of y, (1) can be approximated by [13], [22]

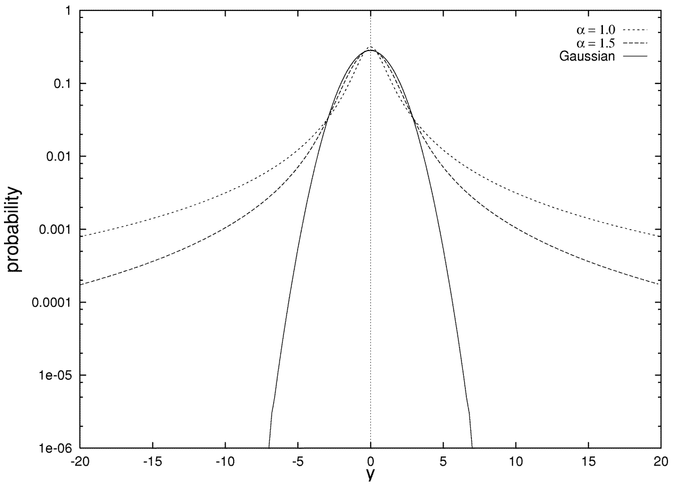


Fig. 1. Comparison of the Lévy probability distributions of $\alpha = 1.0$ and $\alpha = 1.5$ with the standard Gaussian distribution. The vertical coordinate is of log scale and $\gamma = 1$ for all α.

$$L_\alpha(y) \propto y^{-(\alpha+1)}, \qquad |y| \gg 1 \tag{4}$$

which has a power law behavior. Thus, for $0 < \alpha < 2$, this distribution has infinite second moment. Also, (4) does exhibit a scale-free characteristic, meaning that (4) is invariant under the scaling $y \rightarrow by$, $b \in R$, up to a constant factor. This is the reason why the Lévy distribution has been studied widely in conjunction with fractals.

An algorithm for generating Lévy random numbers is needed because the analytic form of the Lévy distribution is not known for general α. Such an algorithm was introduced in [14]. This algorithm utilizes two independent random variables x and y from standard Gaussian distribution and performs a nonlinear transformation

$$v = \frac{x}{|y|^{\frac{1}{\alpha}}}. \tag{5}$$

With this, it was shown that the sum of these variables with an appropriate normalization

$$z_n = \frac{1}{n^{\frac{1}{\alpha}}}\sum_{k=1}^{n} v_k \tag{6}$$

converges to the Lévy probability distribution with larger n. It, however, has a practical disadvantage because it requires generating many (usually, $n = 100$) such random variables in order to get the desired Lévy distribution. To get a more efficient algorithm, the following nonlinear transformation was considered:

$$w = \left\{(k(\alpha) - 1)\, e^{-\frac{|v|}{c(\alpha)}} + 1\right\} v \tag{7}$$

where parameters $k(\alpha)$ and $c(\alpha)$ depend on α and are calculated in [14]. In this way, one can generate random variables of the Lévy probability distribution for $0.8 < \alpha < 1.95$, even for $n = 1$. Fig. 2 shows a distribution of the random variables generated from the above nonlinear transformation with $n = 1$.

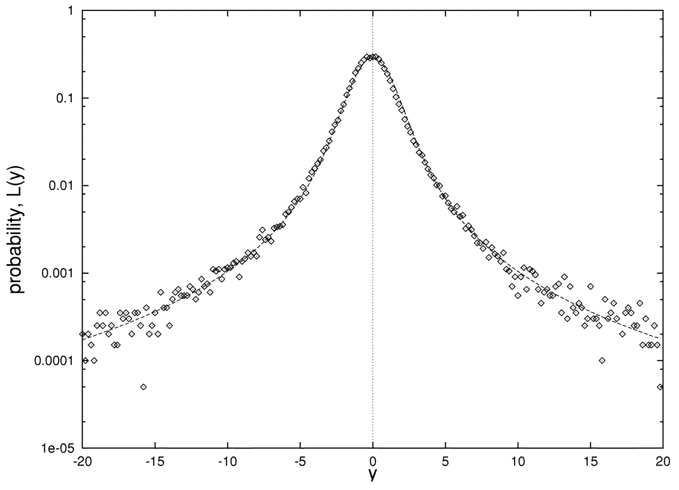


Fig. 2. Lévy probability distribution of $\alpha = 1.5$. Dashed line is obtained from the numerical integration of the Lévy distribution of (1) and scattered diamonds are random numbers generated from (7).

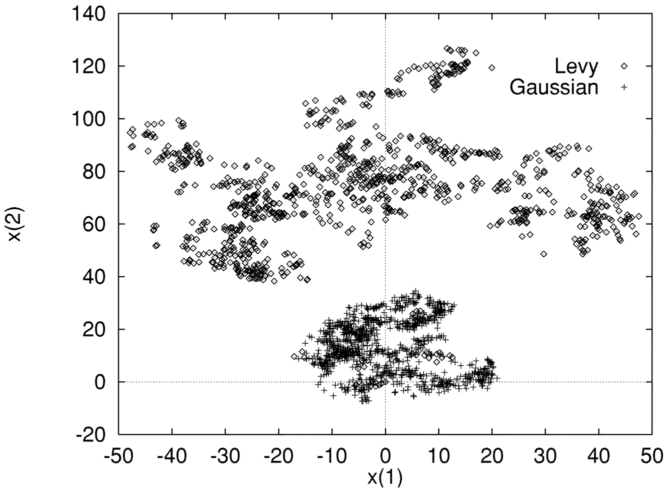


Fig. 3. Plot of the two dimensional variables from the Gaussian and Lévy ($\alpha = 1.5$) mutations. We started from the origin, $x^0(1) = x^0(2) = 0$ and plotted 1000 points with $\beta = 1$.

III. Analysis of Mutation Operations

Before applying the Lévy mutation to EP, we first analyze both Gaussian and Lévy mutations to see how effectively they search the search space.

A. Mean-Square Displacements

The mutation operation in EP can be described schematically as follows. For each individual x in the population, its offspring x' is created as

$$x' \sim x + \beta Y \tag{8}$$

where β represents possible adaptive mutation and Y is a random variable of certain probability distribution.

To see how effectively the mutation operations can explore the search space, we calculate the root mean-square displacements after many mutation operations, indicating the average variation of the offspring from the parents.

Denote x^t to be an individual at the tth generation. For the sake of simplicity, assume that adaptive mutation is turned off. That is, we regard β as a constant. Using the above conventions, an individual at the tth generation can be expressed in terms of an individual at the $(t-1)$th generation

$$x^t = x^{t-1} + \beta Y^{t-1} \tag{9}$$

where Y^{t-1} is a random variable generated at the $(t-1)$th generation. We are interested in the variation of x after t generations, which is given by

$$\Delta(t) \equiv x^t - x^0 = \beta \sum_{k=0}^{t-1} Y^k. \tag{10}$$

In the case of the Gaussian mutation, each Y^k is an identically distributed random variable from the Gaussian distribution with mean 0 and variance 1. Therefore, $\Delta(t)$ has a Gaussian distribution with mean 0 and variance $\beta^2 t$. Thus, the square root of the mean displacement of $\Delta(t)$ is given by

$$\sqrt{\langle \Delta(t)^2 \rangle} = \beta\sqrt{t} \propto \sqrt{t} \tag{11}$$

where $\langle \cdots \rangle$ stands for expectation. This is similar to the Brownian motion in statistical mechanics. From (11), we see that the average variation of an individual after t generations depends on $\sqrt{t}$. That is, on average, the mutated offspring x^t after t generations differs from its original parent x^0 by a factor of $\sqrt{t}$. Thus, in order to get a large varied (or mutated) offspring, we need to carry out successive mutations through many generations.

Now, consider the Lévy mutation. Since the Lévy probability distribution has the following limit property

$$L_\alpha(y) \sim \frac{1}{y^{\alpha+1}}, \qquad \text{when } |y| \gg 1 \tag{12}$$

we have

$$\langle y^2 \rangle \sim \int_{-\infty}^{\infty} \frac{y^2}{y^{\alpha+1}} dx = \infty, \qquad \text{for } 0 < \alpha < 2. \tag{13}$$

Since the Lévy probability distribution is *stable*, the sum of its individual random variables also follows the Lévy probability distribution. That is, the square root of the mean-square displacement (or variation) of $\Delta(t)$ becomes

$$\sqrt{\langle \Delta(t)^2 \rangle} = \infty. \tag{14}$$

It should be noted that, in the case of the Lévy mutation, the mean-square displacement does not depend on the generation t. Even though generating infinite variation is not realistic in practice, the infinite displacement implies that it is possible to get a large variation on the offspring even after just one generation.

The large variation at a single mutation enables the Lévy mutation to search a wider region of the search space globally. To see this more concretely, consider an individual having two components, $x^t(j)$, $j = 1, 2$, as follows:

$$x^{(t)}(1) = x^{(t-1)}(1) + \beta Y_1^{(t-1)} \tag{15}$$

$$x^{(t)}(2) = x^{(t-1)}(2) + \beta Y_2^{(t-1)}. \tag{16}$$

Using both mutation operations for Y_j^k, we plot variables $(x^{(t)}(1), x^{(t)}(2))$, $t = 1, 2, \cdots, 1000$, as shown in Fig. 3. As seen in the figure, the Lévy-mutated variables cover a wider range than those mutated by Gaussian distributions. Large variations of the mutated offspring can help escape from local optima. This is especially true when local optima have deep and wide basins.

TABLE I
BENCHMARK FUNCTIONS USED IN THIS STUDY. N STANDS FOR THE DIMENSION OF THE FUNCTIONS AND S THEIR RANGES. f_1, f_2, AND f_3 ARE UNIMODAL FUNCTIONS, f_4–f_9 ARE FUNCTIONS WITH MANY LOCAL MINIMA, AND f_{10}–f_{14} ARE FUNCTIONS WITH A FEW LOCAL MINIMA ONLY

Test Functions	N	S
$f_1 = \sum_{i=1}^N x_i^2$	30	$(-100, 100)^N$
$f_2 = \sum_{i=1}^N (\sum_{j=1}^i x_j)^2$	30	$(-100, 100)^N$
$f_3 = \sum_{i=1}^{N-1} \{100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\}$	30	$(-30, 30)^N$
$f_4 = -\sum_{i=1}^N x_i \sin(\sqrt{x_i})$	30	$(-500, 500)^N$
$f_5 = \sum_{i=1}^N \{x_i^2 - 10\cos(2\pi x_i) + 10\}$	30	$(-5.12, 5.12)^N$
$f_6 = -20exp\{-0.2\sqrt{\frac{1}{N}\sum_{i=1}^N x_i^2}\} - exp\{\frac{1}{N}\sum_{i=1}^N \cos(2\pi x_i)\} + 20 + e$	30	$(-32, 32)^N$
$f_7 = \frac{1}{4000}\sum_{i=1}^N x_i^2 - \Pi_{i=1}^N \cos(\frac{x_i}{\sqrt{i}}) + 1$	30	$(-600, 600)^N$
$f_8 = \frac{\pi}{N}\{10\sin^2(\pi y_i) + \sum_{i=1}^{N-1}(y_i - 1)^2\{1 + 10\sin^2(\pi y_{i+1})\} + (y_n - 1)^2\}$ $+ \sum_{i=1}^N u(x_i, 10, 100, 4), \quad y_i = 1 + 1/4(x_i + 1)$	30	$(-50, 50)^N$
$f_9 = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{N-1}(x_i - 1)^2\{1 + \sin^2(3\pi x_{i+1})\} + (x_N - 1)^2\{1 + \sin^2(2\pi x_N)\}\}$ $+ \sum_{i=1}^N u(x_i 5, 100, 4)$	30	$(-50, 50)^N$
$f_{10} = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$(-5, 5)^N$
$f_{11} = \{1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\}\times$ $\{30 + (2x_1 - 3x_2)^2)(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\}$	2	$(-2, 2)^N$
$f_{12} = -\sum_{i=1}^5 \{\sum_{j=1}^4 (x_j - a_{ij})^2 + c_i\}^{-1}$	4	$(0, 10)^N$
$f_{13} = -\sum_{i=1}^7 \{\sum_{j=1}^4 (x_j - a_{ij})^2 + c_i\}^{-1}$	4	$(0, 10)^N$
$f_{14} = -\sum_{i=1}^{10} \{\sum_{j=1}^4 (x_j - a_{ij})^2 + c_i\}^{-1}$	4	$(0, 10)^N$

B. Number of Distinct Values Obtained by Mutation Operations

As seen in (10), an offspring of the tth generation can be expressed in terms of the sum of t random numbers. If one regards each generated random number as a random walk, then the tth offspring can be thought of as the result of the t-step random walk. From this one can calculate the expected value of the number of distinct values taken by the t-step random walk for large t. This is important because it indicates how effective the mutation operation is.

Denote D_t to be the number of distinct values obtained by the t-step random walk. In order to get any useful quantities related to D_t, we need to know the probability distribution of D_t, which in general is hard to obtain except for some special cases. It is, however, relatively easy to calculate the expectation $\langle D_t \rangle$ of D_t. For a t-step random walk, the average can be expressed in terms of the sum of contributions from each step. Let P_k be the probability that a new value is obtained at step k. At each step, there are two possible states, either a new value is obtained or not. If a new value is obtained, then the number of distinct values is increased by one, otherwise by zero. Therefore, the average of the total number of distinct values in t steps is given by

$$\langle D_t \rangle = \sum_{k=1}^{t} (1 \cdot P_k + 0 \cdot (1 - P_k)) = \sum_{k=1}^{t} P_k. \tag{17}$$

Note that we express the average number of distinct values as the sum of contributions from each step. If P_k is a constant regardless of k, then D_t follows the well-known binomial distribution.

Now, we calculate $\langle D_t \rangle$ for two cases: finite and infinite second moments. For the Gaussian mutation, which has an *finite* second moment, it can be shown [15] that

$$\langle D_t \rangle \sim \sigma\sqrt{\frac{8t}{\pi}} \propto t^{\frac{1}{2}} \tag{18}$$

where σ is the square root of the second moment. In the case of the Lévy mutation, which has an *infinite* second moment, the expected number of distinct values obtained by a random walk is [16]

$$\begin{aligned} \langle D_t \rangle &\propto t, && \text{for } 0 < \alpha < 1 \\ &\propto \frac{t}{\ln t}, && \text{for } \alpha = 1 \\ &\propto t^{\frac{1}{\alpha}}, && \text{for } 1 < \alpha < 2. \end{aligned} \tag{19}$$

From the above two equations, we can see that for all α, $0 < \alpha < 2$, the random walk from the Lévy mutation yields more distinct values than that from the Gaussian mutation. This in turn implies that one can explore the search space more effectively with the Lévy mutation. Therefore, the Lévy mutation not only can search a wider area of the search space but also generate more distinct values in the search space. The Lévy mutation searches the search space more evenly. It is the infinite second moment of the Lévy distribution that makes all the difference. We will see the effect of the Lévy mutation in the experiments on the function optimization in the following section.

TABLE II

EXPERIMENTAL RESULTS, AVERAGED OVER 50 INDEPENDENT RUNS, FROM LEP AND GEP. THE NUMBERS IN THE PARENTHESES INDICATE STANDARD DEVIATIONS. THE t-TEST VALUES LISTED ARE GEP-LEP

	$\alpha = 0.8$	$\alpha = 1.0$	$\alpha = 1.2$	$\alpha = 1.4$	$\alpha = 1.6$	$\alpha = 1.8$	Gaussian	t-test
f_1	0.030483	0.010173	0.005265	0.003224	0.002614	0.001979	0.000950	-18.87†
	(0.006043)	(0.001650)	(0.000911)	(0.000539)	(0.000428)	(0.000351)	(0.000150)	($\alpha = 1.8$)
f_2	105.378755	172.245618	245.264251	285.972195	332.198891	302.296411	497.120615	3.33†
	(62.316131)	(113.45391)	(176.12988)	(196.91763)	(175.98249)	(156.32334)	(298.66450)	($\alpha = 1.6$)
f_3	102.854049	80.265797	77.714380	72.568030	96.739696	72.343559	98.673779	0.12
	(88.248025)	(89.156805)	(93.602971)	(60.285805)	(77.401102)	(72.916310)	(78.908296)	($\alpha = 1.6$)
f_4	8.3415	-11433.1817	-10695.9293	-10238.3929	-9612.61319	-8927.10660	-7976.67670	9.37†
	(222.62858)	(307.68776)	(404.13107)	(519.08325)	(499.56902)	(493.74672)	(510.36871)	($\alpha = 1.8$)
f_5	42.063987	24.398049	20.507714	20.600218	25.076582	38.266239	63.415617	6.86†
	(15.126775)	(4.729001)	(4.339205)	(4.2007428)	(5.758605)	(8.978101)	(15.660637)	($\alpha = 0.8$)
f_6	0.179721	0.088504	0.060169	0.048737	0.045005	0.974767	8.882487	14.08†
	(0.028128)	(0.009141)	(0.005743)	(0.005138)	(0.006544)	(2.198305)	(3.259676)	($\alpha = 1.8$)
f_7	0.012990	0.012899	0.025948	0.029154	0.038724	0.041181	0.058204	1.49
	(0.011138)	(0.017892)	(0.033769)	(0.029484)	(0.050343)	(0.047762)	(0.063867)	($\alpha = 1.8$)
f_8	0.000279	0.000097	0.000055	0.022922	0.178059	0.301026	0.620834	1.95†
	(0.000057)	(0.000018)	(0.000009)	(0.069665)	(0.343713)	(0.427370)	(1.064294)	($\alpha = 1.8$)
f_9	1.029439	0.004141	0.001394	0.000737	0.000520	0.000460	0.094838	2.8†
	(2.283157)	(0.000703)	(0.000214)	(0.000117)	(0.000063)	(0.000191)	(0.475849)	($\alpha = 1.8$)
f_{10}	-1.028949	-1.028753	-1.029691	-1.029168	-1.029783	-1.029775	-1.030435	1.06
	(0.003247)	(0.007019)	(0.003460)	(0.004669)	(0.002860)	(0.002442)	(0.003212)	($\alpha = 1.6$)
f_{11}	3.189039	3.183000	3.258343	3.193855	3.374878	3.255450	3.211548	0.31
	(0.218324)	(0.264132)	(0.339389)	(0.302817)	(0.463998)	(0.369106)	(0.268470)	($\alpha = 1.4$)
f_{12}	-7.839394	-7.841831	-8.494128	-7.689142	-7.842465	-8.936393	-8.067509	0.004
	(2.934361)	(2.813423)	(2.488407)	(3.110630)	(2.915006)	(2.321307)	(2.791307)	($\alpha = 0.8$)
f_{13}	-9.683556	-9.672865	-8.786782	-9.861692	-9.418368	-9.277473	-8.995198	0.03
	(1.958713)	(2.014291)	(2.764645)	(1.633838)	(2.155953)	(2.599248)	(2.605137)	($\alpha = 0.8$)
f_{14}	-9.705024	-9.932785	-9.754729	-10.198659	-9.202691	-9.369379	-9.277539	0.54
	(2.284323)	(1.856711)	(2.130905)	(1.335125)	(2.704432)	(2.525967)	(2.654420)	($\alpha = 0.8$)

† The t value of 49 degrees of freedom is significant at a 0.05 level of significance by a two-tailed t-test.

IV. FUNCTION OPTIMIZATION USING EP

In this section, we discuss EP algorithms for function optimization using the Gaussian and Lévy mutation operations. For convenience, we will call EP with the Lévy mutation LEP and that with the Gaussian mutation GEP. In optimizing a function $f(\vec{x})$, an EP algorithm aims to find

$$\vec{x}_{\min} \text{ such that } \forall \vec{x}, \qquad f(\vec{x}_{\min}) \leq f(\vec{x}). \tag{20}$$

For GEP, we use the algorithm given in [11].

Step 1) Generate the initial population consisting of μ individuals, each of which can be represented a set of real vectors, $(\vec{x}_i, \vec{\sigma}_i)$, $i = 1, 2, \cdots, \mu$. Each $\vec{x}_i$ and $\vec{\sigma}_i$ has m independent components

$$\vec{x}_i = \{x_i(1), x_i(2), \cdots, x_i(m)\} \tag{21}$$
$$\vec{\sigma}_i = \{\sigma_i(1), \sigma_i(2), \cdots, \sigma_i(m)\}. \tag{22}$$

Step 2) LEP and GEP differ only in this step.

LEP: For each parent $(\vec{x}_i, \vec{\sigma}_i)$, create an offspring $(\vec{x}'_i, \vec{\sigma}'_i)$ as follows: for $j = 1, 2, \cdots, m$

$$\sigma'_i(j) = \sigma_i(j) \exp\{\tau' N(0,1) + \tau N_j(0,1)\} \quad (23)$$

$$x'_i(j) = x_i(j) + \sigma'_i(j) L_j(\alpha) \quad (24)$$

where, $L_j(\alpha)$ is a random number generated anew for each j from the Lévy distribution with the parameter α.

GEP: For each parent $(\vec{x}_i, \vec{\sigma}_i)$, create an offspring $(\vec{x}'_i, \vec{\sigma}'_i)$ as follows: for $j = 1, 2, \cdots, m$

$$\sigma'_i(j) = \sigma_i(j) \exp\{\tau' N(0,1) + \tau N_j(0,1)\} \quad (25)$$

$$x'_i(j) = x_i(j) + \sigma'_i(j) N_j(0,1) \quad (26)$$

where $N(0,1)$ stands for a standard Gaussian random variable fixed for a given i and $N_j(0,1)$ a newly generated Gaussian random variable for each component j. The parameters τ and τ' are defined as [17]

$$\tau = \frac{1}{\sqrt{2\sqrt{\mu}}} \quad \text{and} \quad \tau' = \frac{1}{\sqrt{2\mu}}. \quad (27)$$

Step 3) From μ parents and their μ offspring, calculate their fitness values $f_1, f_2, \cdots, f_{2\mu}$.

Step 4) Define and initialize a "winning function" for every parent and offspring as $w_i = 0$, for $i = 1, 2, \cdots, 2\mu$. For each i, select one fitness function, say $f_j (j \neq i)$ and compare the two fitness functions. If f_i is less than f_j, the winning function for individual i increases by one, $w_i = w_i + 1$. Perform this procedure q times for every parent and offspring.

Step 5) According to the winning function $w_i, i = 1, 2, \cdots, 2\mu$, select μ individuals that have the largest winning values to be the parents for the next generation.

Step 6) Repeat Steps 2–5, until the stopping criteria are satisfied.

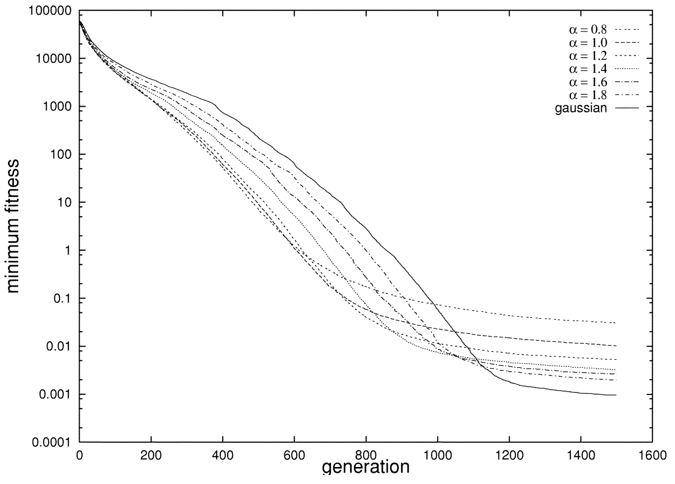
In the above algorithm, the parameter α in LEP remains unchanged during evolution. The effect of different αs will be studied in later sections.

V. Experimental Results and Analysis

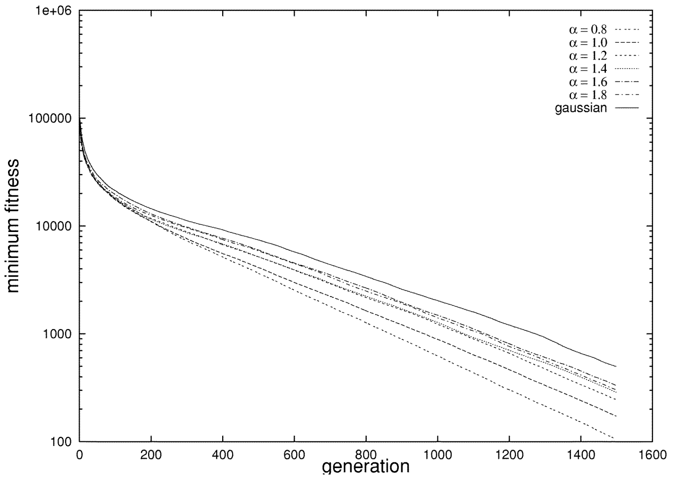
We applied LEP and GEP in Section IV to a set of benchmark optimization problems. Table I shows the benchmark functions and the ranges of the variables that have been used in this study. These were considered in an early study [11]. We divided the functions into three broad classes: functions with no local minima, many local minima, and a few local minima. The following parameters were used in our experiments:

- population size: $\mu = 100$;
- tournament size: $q = 10$;
- initial standard deviation: $\sigma = 3.0$;
- maximum number of generations: 1500 for $f_1, f_2, \cdots, f_9$, 30 for f_{10}, f_{11}, and 100 for f_{12}, f_{13}, f_{14}.

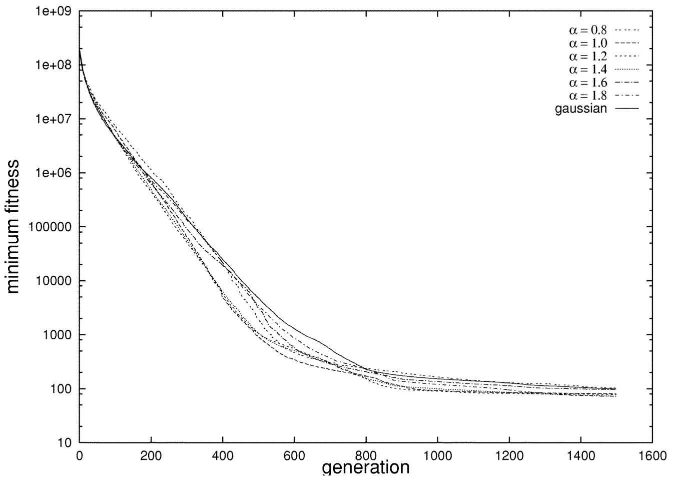
Table II summarizes our experimental results. It is clear from the table that LEP performed no worse than GEP on all benchmark functions except for the sphere function.



(a)



(b)



(c)

Fig. 4. Evolutionary processes of LEP and GEP. The results were averaged over 50 independent trials. (a), (b), and (c) correspond to the results of test functions f_1, f_2, and f_3, respectively.

A. Unimodal Functions

Fig. 4 shows the evolutionary processes, averaged over 50 independent trials, for $f_1 - f_3$. LEP converged faster than GEP for all three functions initially, around 600 to 800 generations. However, the difference between the two becomes unclear when

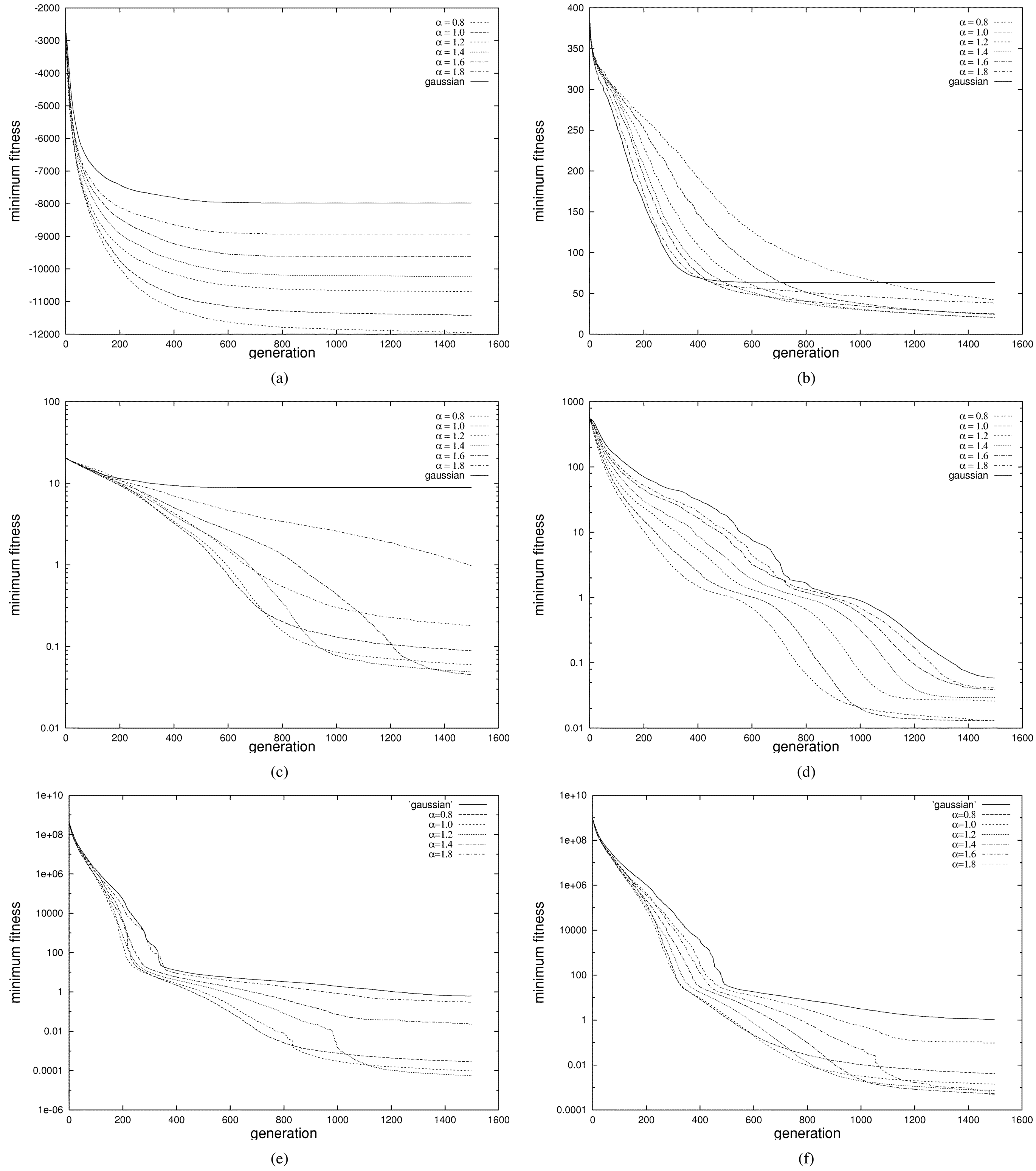
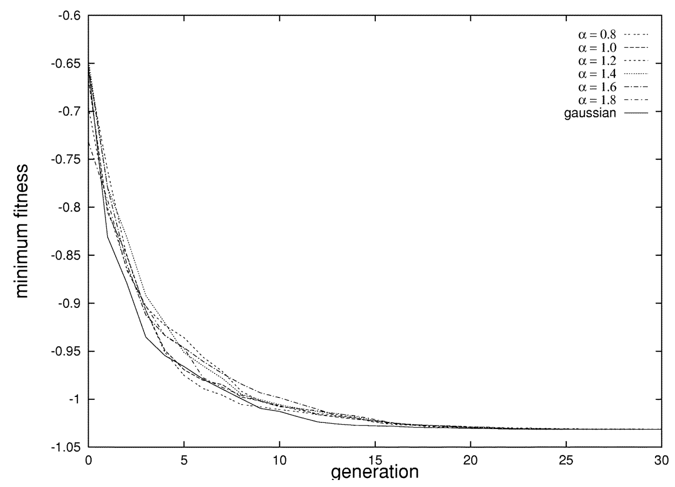


Fig. 5. Evolutionary processes for functions f_4–f_9. The results were averaged over 50 independent trials. (a)–(f) correspond to functions f_4–f_9, respectively.

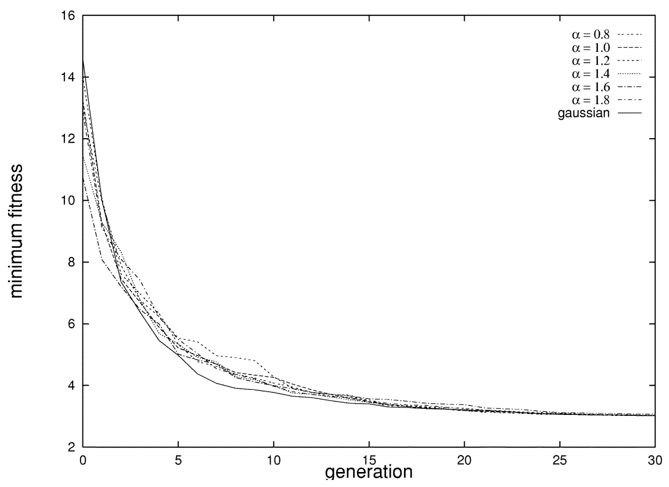
the algorithms were run for a longer time. GEP even overperformed LEP on f_1, the sphere function. Such behaviors are very similar to those observed earlier in the comparison of Gaussian and Cauchy mutation [20]. LEP converged faster initially because of Lévy mutation's long jumps and more distinct values. However, long jumps can be detrimental when the population is close to the global optimum. The offspring generated by long jumps tend to move away from the global optimum, which is what happened for f_1 toward the later stage of evolution.

B. Functions With Many Local Minima

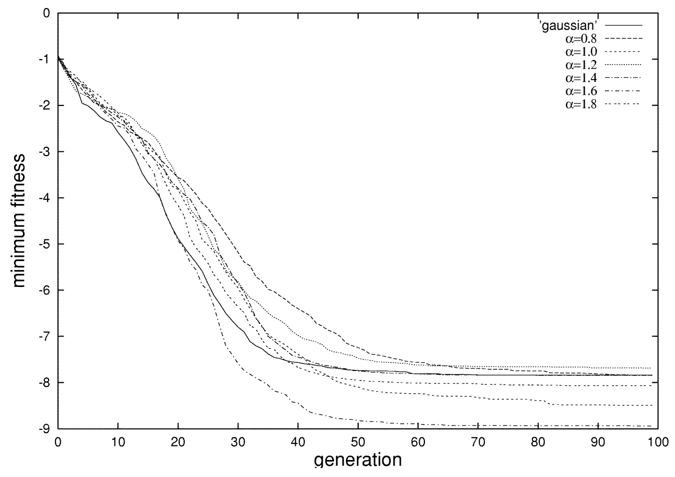
For the functions with many local minima, i.e., $f_5 - f_9$, the experimental results are given in Fig. 5. It is clear that LEP produced



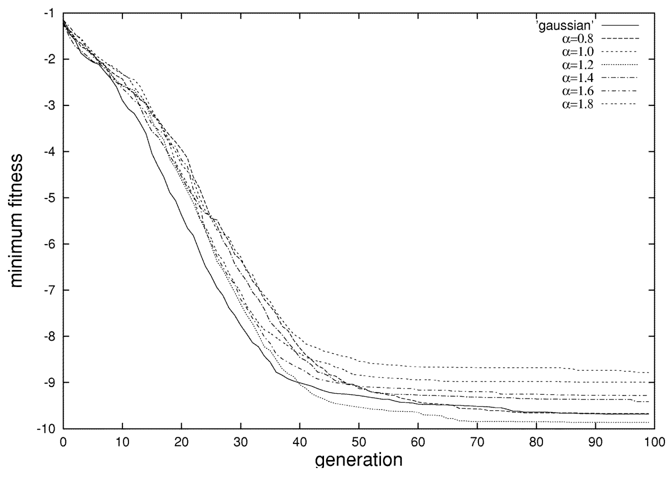
(a)



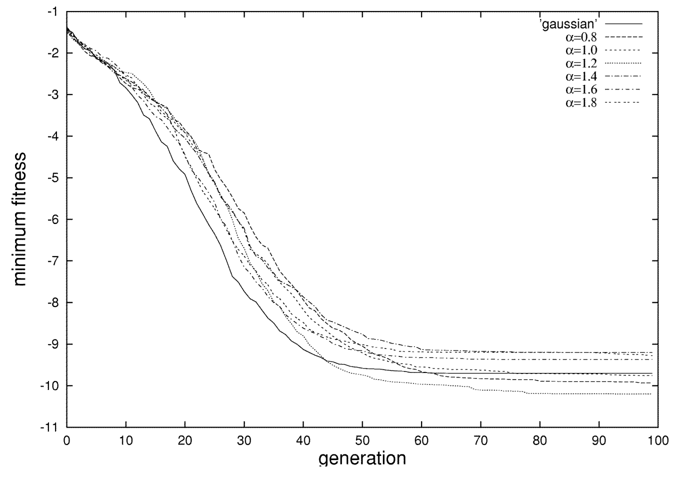
(b)



(c)



(d)



(e)

Fig. 6. Evolutionary processes of LEP and GEP. The results were averaged over 50 independent trials. (a)–(e) correspond to functions f_{10}–f_{14}, respectively.

consistently better results than GEP, both in terms of initial convergence speed and final results. This is not surprising because Lévy mutation's long jumps enabled LEP to move from one local minimum area to another relatively easily in comparison with Gaussian mutation. Lévy mutation's ability to explore the search space more evenly also helped in achieving good performance. One would expect the performance difference between LEP and GEP to be greater as the number of local minima increases.

C. Functions With a Few Local Minima

Fig. 6 shows the results of LEP and GEP on functions with a few local minima, i.e., $f_{10} - f_{14}$. In this case, it is hard to tell the performance difference between LEP and GEP, especially for f_{10} and f_{11}. This is also clearly reflected by the t-test results in Table II. The reasons for such behaviors from LEP and GEP might be related to the average distance between an initial population and the global optimum. In other words, LEP did not outperform GEP significantly because initial populations generated for $f_{10} - f_{14}$ were relatively close to the global minima. Lévy mutation's long jumps did not help in such cases. This is not unlike the situation with FEP, which performed worse than GEP on Shekel functions $(f_{12} - f_{14})$ [20].

VI. Adaptive Lévy Mutations

A. Motivation

The results for $f_{12} - f_{14}$ in Fig. 6 show clearly that different α values in LEP could lead to different final results for LEP. No single α value was the best for all different problems. There is a need to find different α values for different problems.

Although different α values were tested in the previous section, they were fixed during evolution. It is desirable to have different α values not only for different problems but also for different evolutionary search stages for a single problem. Self-adaptation can be introduced into LEP to find an appropriate α value dynamically and autonomously [19]. The idea presented here was inspired by IFEP proposed in [20]. IFEP is an EP algorithm based on mixing different mutation operators. It generates two candidate offspring from each parent, one by Cauchy mutation and the other by Gaussian mutation, and selects the better one as the surviving offspring.

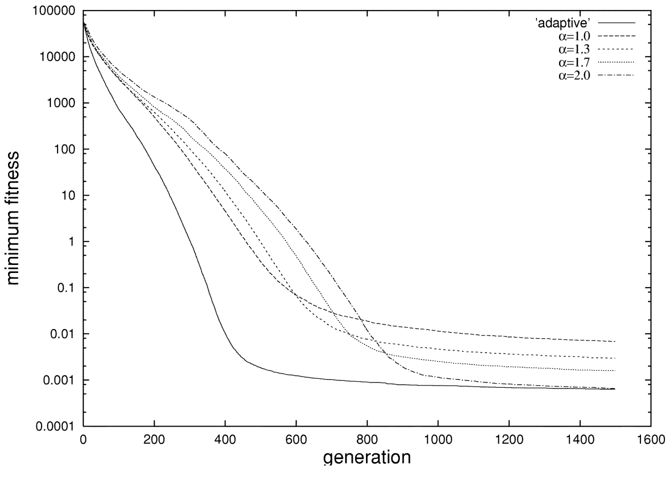
The Lévy distribution provides an ideal opportunity for designing adaptive mutations because of its continuously adjustable parameter α. In this study, we implement the adaptive Lévy mutation by generating several candidate offspring using different α values, and select the best as the survival offspring. To evaluate the performance of adaptive Lévy mutation, we keep the difference between LEP and adaptive LEP (i.e., LEP with adaptive Lévy mutation) as small as possible. The adaptive LEP implemented in this paper is the same as LEP except for the following: we generate four candidate offspring with $\alpha = 1.0$, 1.3, 1.7, and 2.0 from each parent and select the best one as the surviving offspring. This scheme is adaptive because which α to use is not predefined and is determined by evolution.

It is known that Gaussian mutation $(\alpha = 2.0)$ works better for searching a small local neighborhood, whereas Cauchy mutation $(\alpha = 1.0)$ is more effective at exploring a large area of the search space [20]. By adding two additional candidate offspring $(\alpha = 1.3$ and $1.7)$, one is not limited to the two extremes. LEP will be able to determine dynamically which α value to use depending on which search stage it is in, from global exploration to local fine tuning. Similar ideas to this can be found in the study of a complex adaptive system, such as the minority game [21].

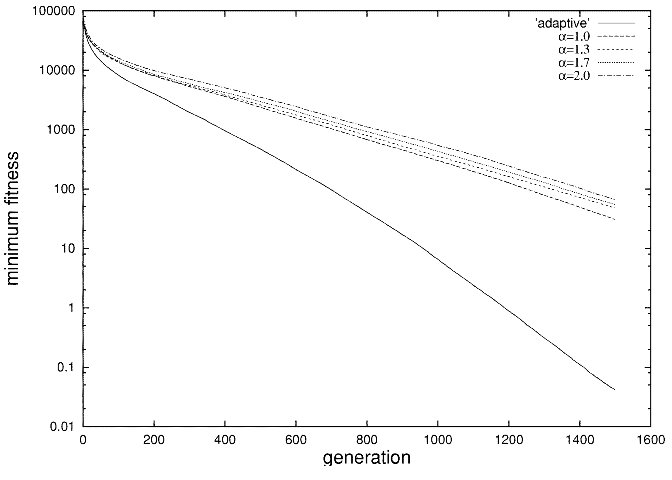
B. Experimental Results and Discussions

We applied the adaptive LEP to the same test functions using the same set of parameters as described in Section V. Since the adaptive LEP uses four different α values, we compared the adaptive LEP with four LEP algorithms with four different α values. For each function, five different sets of experiments were carried out.

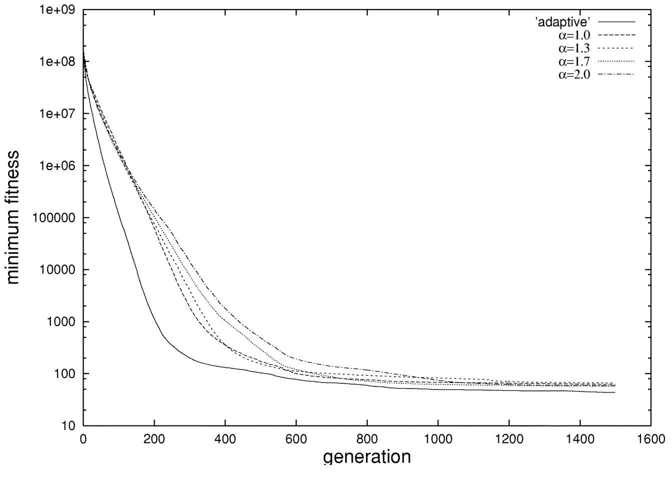
To make the comparison fair in terms of computing time, the population size of the adaptive LEP was reduced to one quarter of that used for LEP with a single fixed α, since each individual



(a)



(b)



(c)

Fig. 7. Evolutionary processes of different LEP on f_1-f_3. The results were averaged over 50 independent runs. (a), (b), and (c) correspond to results for f_1, f_2, and f_3, respectively. The "adaptive" in the legend indicates the result of the adaptive LEP. The others indicate LEP with the corresponding α. The population size for the adaptive LEP was 100 and the others 400.

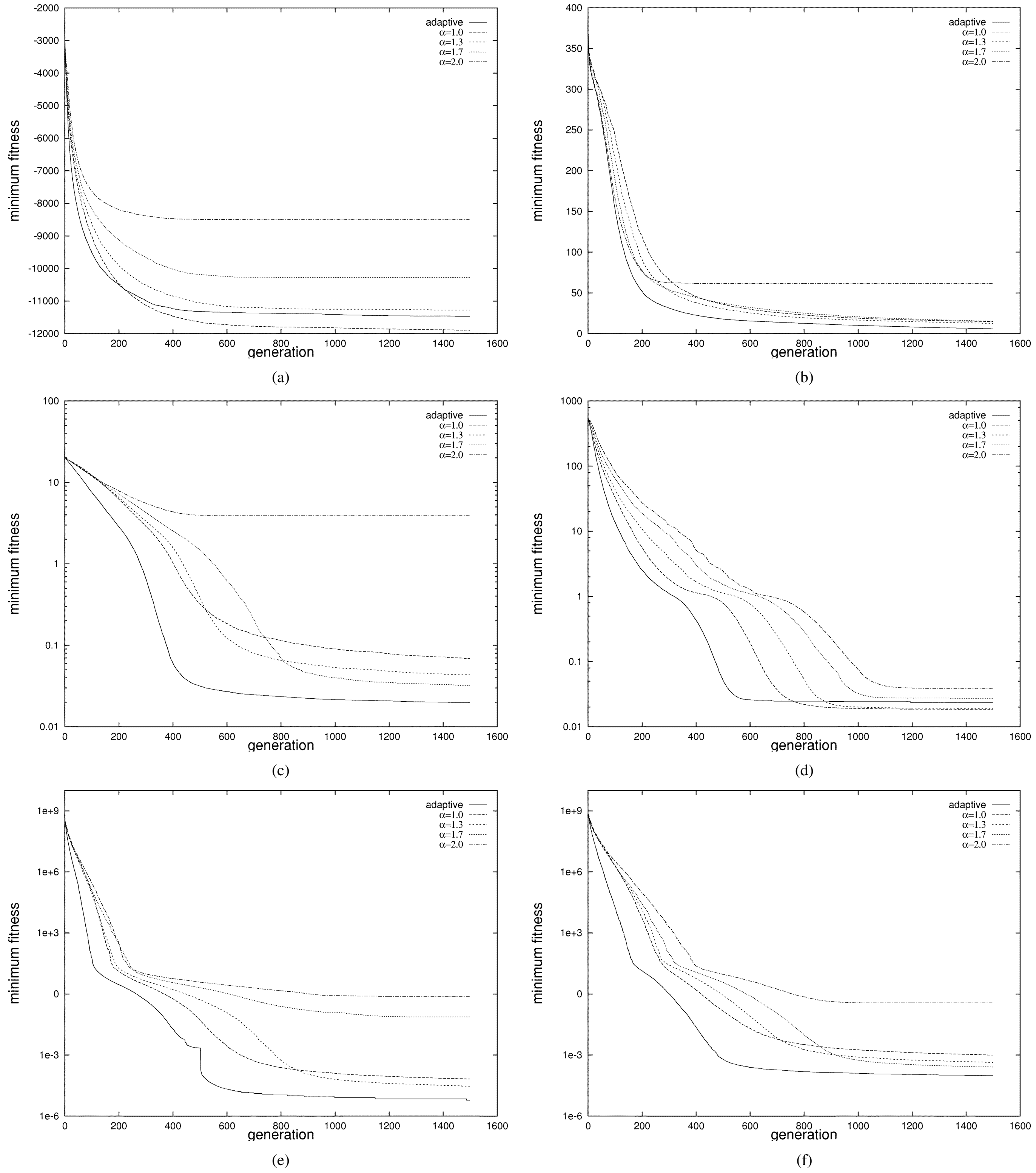


Fig. 8. Evolutionary processes of different LEP algorithms. The results were averaged over 50 independent runs. (a)–(f) correspond to the results for f_4 to f_9, respectively. The "adaptive" in the legend indicates the result of the adaptive LEP. The others indicate LEP with a single fixed α. The population size for the adaptive LEP was 100 and the others 400.

in the adaptive LEP generates four offspring. When the size of the population was the same, we obtained much better results in favor of the adaptive LEP. It is worth pointing out that the adaptive LEP with a quarter of the original population size actually used less computing time, because operations, such as selection, used less time in a small population.

Fig. 7 shows the experimental results of different LEP algorithms on the three unimodal functions. It is clear that the adaptive LEP converged faster than all other nonadaptive LEP and to a better solution.

Fig. 8 shows the results of different LEP algorithms on the six functions with many local minima. The adaptive LEP performed

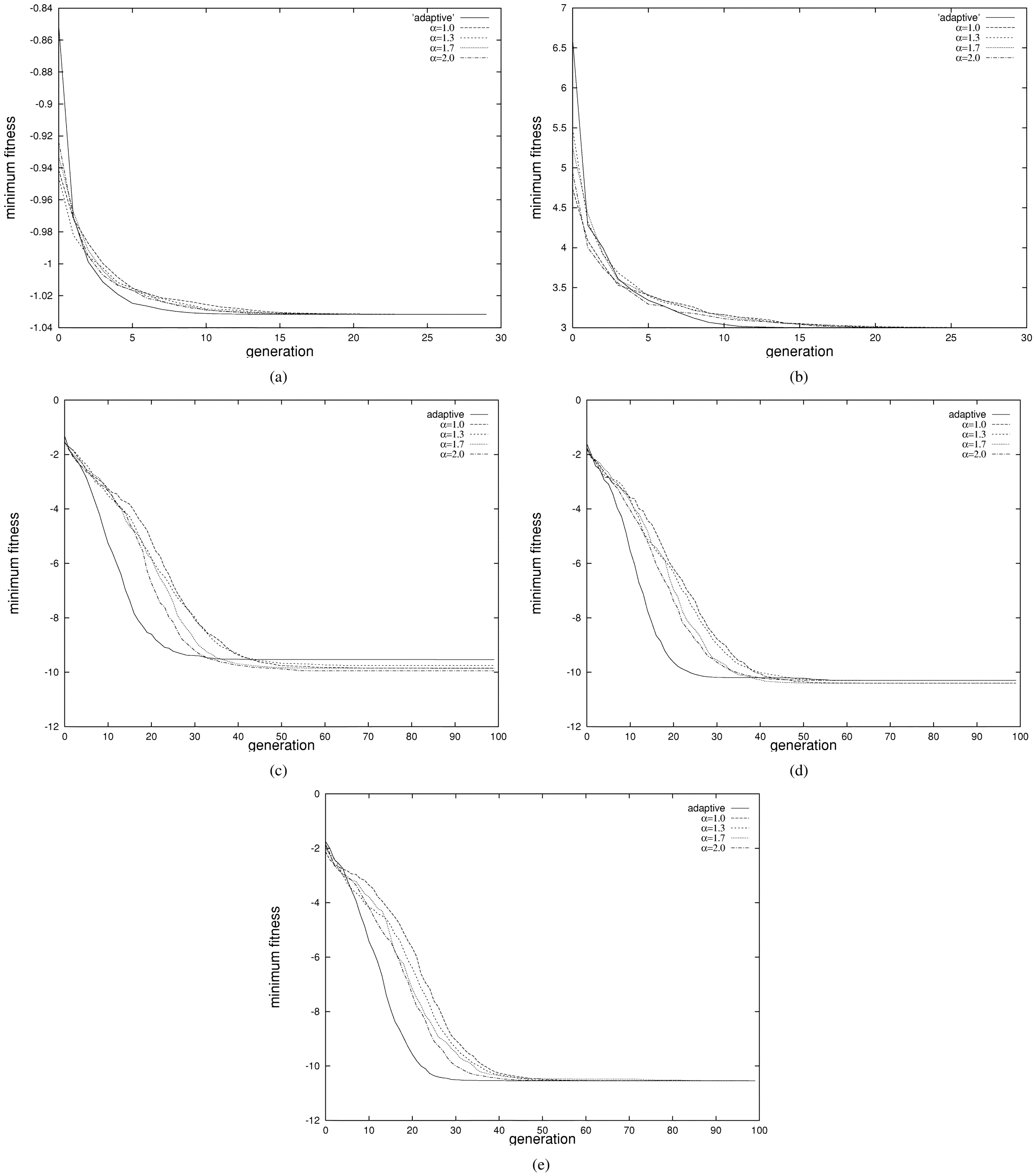


Fig. 9. Evolutionary processes of different LEP algorithms. The results were averaged over 50 independent runs. (a)–(e) correspond to the results for f_{10}–f_{14}, respectively. The "adaptive" in the legend indicates the result of the adaptive LEP. The others indicate LEP with a single fixed α. The population size for the adaptive LEP was 100, and the others 400.

better than others on all functions but f_4 and f_7. Even for f_4 and f_7, the adaptive LEPs initial convergence speed still seemed to be faster than others.

Fig. 9 shows the results of different LEP algorithms on the four functions with a few local minima. In this case, the performance of all LEP algorithms was very similar to each other. There was no statistically significant difference among them although the adaptive LEP appeared to converge faster than others.

Table III summarizes the above results and gives the t-test results, which shows that the adaptive LEP was only outperformed by nonadaptive LEP on a single problem f_4. It also shows that

TABLE III
THE EXPERIMENTAL RESULTS, AVERAGED OVER 50 INDEPENDENT RUNS, OF THE ADAPTIVE LEP AND FOUR NONDAPTIVE LEP ALGORITHMS. "MEAN BEST" INDICATES THE AVERAGE OF THE MINIMUM VALUES OBTAINED AND "STD DEV" STANDS FOR THE STANDARD DEVIATION. "BEST α" INDICATES THE α VALUE THAT LED TO THE BEST NONDAPTIVE LEP

Funct-ions	Number of Generation	Adaptive Mean Best (Std Dev)	Best Lévy Mean Best (Std Dev)	Best α	t-test
f_1	1500	6.32×10^{-4} (7.6×10^{-5})	6.59×10^{-4} (6.4×10^{-5})	2.0	1.9
f_2	1500	0.041850 (0.059696)	30.628906 (22.113122)	1.0	9.68†
f_3	1500	43.40 (31.52)	57.75 (41.60)	1.7	1.9
f_4	1500	-11469.2 (58.2)	-11898.9 (52.2)	1.0	-7.69†
f_5	1500	5.85 (2.07)	12.50 (2.29)	1.3	15.07†
f_6	1500	1.9×10^{-2} (1.0×10^{-3})	3.1×10^{-2} (2.0×10^{-3})	1.7	37.57†
f_7	1500	2.4×10^{-2} (2.8×10^{-2})	1.8×10^{-2} (1.7×10^{-2})	1.0	-1.28
f_8	1500	6.0×10^{-6} (1.0×10^{-6})	3.0×10^{-5} (4.0×10^{-6})	1.3	40.75†
f_9	1500	9.8×10^{-5} (1.2×10^{-5})	2.6×10^{-4} (3.0×10^{-5})	1.7	35.10†
f_{10}	30	-1.031 (0.00)	-1.031 (0.00)	1.0	0.0
f_{11}	30	3.000 (0.000)	3.000 (0.000)	2.0	0.0
f_{12}	100	-9.54 (1.69)	-9.95 (0.99)	2.0	-1.47
f_{13}	100	-10.30 (0.74)	-10.40 (1.0×10^{-4})	2.0	-0.95
f_{14}	100	-10.54 (4.9×10^{-5})	-10.54 (3.1×10^{-3})	1.7	0.0

† The t value of 49 degree of freedom is significant at a 0.05 level of significance by a two-tailed test.

the adaptive LEP was quite robust was able to deal with a wide range of different functions. In general, the adaptive LEP was able to perform at least as good as nonadaptive LEP with a fixed α. This was achieved without introducing any extra parameters and no extra computation time. The results further confirm the importance of mixing variable step sizes (induced by different α values) in an evolutionary algorithm if we do not know a problem well in advance.

In order to see how the adaptive LEP works, we take f_6 as an example for detailed analysis. Similar analysis can be done for other functions. In Fig. 10, we plot the number of successful mutations in a population for four different α values as a function of the number of generations. From Fig. 10 and the corresponding Fig. 8(c), we observe that until the algorithm was close to a solution (up to around 400th generation in Fig. 8(c), non-Gaussian mutations ($\alpha = 1.0, 1.3, 1.7$) occupied about a half of the population. However, once the algorithm is in the vicinity of the solution (after around 400th generation in Fig. 8(c), the Gaussian mutation took over and played a dominant role. In other words, non-Gaussian mutations played a significant role in the early stages of evolution, while the Gaussian mutation filled a major role in the later stages of evolution. This is not unexpected since in the early stages the distance between the current search points and the global minimum was usually large and, thus, non-Gaussian mutations tended to be more effective. As the evolution progressed, the current search point approached the global minimum, and as a result, the Gaussian mutation started to be more effective. A similar behavior was observed previously in IFEP [20].

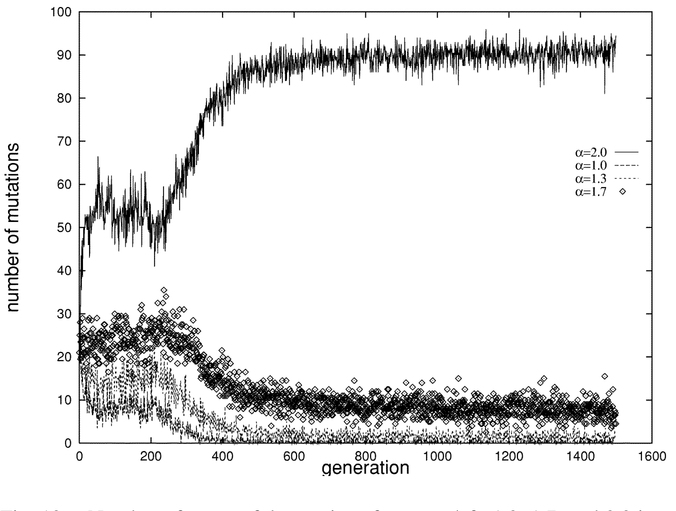


Fig. 10. Number of successful mutations for $\alpha = 1.0, 1.3, 1.7$, and 2.0 in a population when the adaptive LEP was applied to function f_6. The total number of successful mutations should add up to 100%.

VII. CONCLUSION

A new search operator, Lévy mutation, and its adaptive version have been proposed and studied in this paper. Both analytical and experimental work has been carried out to explain why and how Lévy mutation works. In particular, it has been shown that Lévy mutation can lead to a large variation (i.e., step size) and a large number of distinct values in evolutionary search, in comparison with traditional Gaussian mutation. Lévy mutation is also more general and flexible than Cauchy mutation [20] because of the α parameter. In fact, it is the α parameter that enabled us to devise a simple yet effective adaptive version of Lévy mutation. Our experimental results have shown that the adaptive LEP performed well on a number of benchmark functions we tested.

The adaptive and nonadaptive Lévy mutation operators have many aspects for future studies, among which the following three are worth mentioning. First, we have not considered the region of $0 < \alpha < 0.7$. This is mainly due to the fact that a fast algorithm for generating the Lévy random numbers in the $0 < \alpha < 0.7$ region has not been found. However, in order to see whether a further reduction in α would lead to even better results, we need to include this region of the parameter. Second, in applying the adaptive Lévy mutation, we applied the Lévy mutation only to the variation of the variables, not to the strategy parameters. It is interesting to investigate the application of the adaptive Lévy mutation to the strategy parameters. Finally, although we restricted the parameter α to four discrete values, a better scheme might be to let α evolve/adapt continuously during evolution. Such a scheme will be much closer to self-adaptation in evolutionary computation.

ACKNOWLEDGMENT

The authors are grateful to Dr. D. Fogel for acting as the Editor-in-Chief for this submission, and to the anonymous reviewers, anonymous Associate Editor, and Dr. D. Fogel for their useful comments that have helped greatly in improving this paper.

REFERENCES

[1] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. New York: IEEE Press, 1995.
[2] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence Through Simulated Evolution*. New York: Wiley, 1966.
[3] D. B. Fogel and J. W. Atmar, “Comparing genetic operators with Gaussian mutation in simulated evolutionary processes using linear systems,” *Biol. Cybern.*, vol. 63, pp. 111–114, 1990.
[4] D. B. Fogel and L. C. Stayton, “On the effectiveness of crossover in simulated evolutionary optimization,” *Bio. Syst.*, vol. 32, no. 3, pp. 171–182, 1994.
[5] A. V. Sebald and J. Schlenzig, “Minimax design of neural net controllers for highly uncertain plants,” *IEEE Trans. Neural Networks*, vol. 5, pp. 73–82, Jan. 1994.
[6] D. B. Fogel, L. J. Fogel, and W. Atmar, “Meta-evolutionary programming,” in *Proc. 25th Asilomar Conf. Signals, Systems and Computers*, R. Chen, Ed., 1991, pp. 540–545.
[7] D. B. Fogel, “Evolving Artificial Intelligence,” Ph.D. dissertation, Univ. California, San Diego, CA, 1992.
[8] N. Saravanan and D. B. Fogel, “Learning of strategy parameters in evolutionary programming: an empirical study,” in *Proc. 3rd Annual Conf. Evolutionary Programming*, A. Sebald and L. Fogel, Eds. River Edge, NJ: World Scientific, 1994, pp. 269–280.
[9] H.-P. Schwefel, *Numerical Optimization of Computer Models*. Chichester, U.K.: Wiley, 1981.
[10] J. Reed, R. Toombs, and N. Barricelli, “Simulation of biological evolution and machine learning,” *J. Theor. Biol.*, vol. 17, pp. 319–342, 1967.
[11] X. Yao and Y. Liu, “Fast evolutionary programming,” in *Evolutionary Programming V: Proceedings of the 5th Annual Conference on Evolutionary Programming*. Cambridge, MA: MIT Press, 1996.
[12] B. Mandelbrot, *The Fractal Geometry of Nature*. San Francisco, CA: Freeman, 1982.
[13] P. Lévy, *Theorie de l’Addition des Veriables Aleatoires*. Paris, France: Gauthier-Villars, 1937.
[14] R. Mantegna, “Fast, accurate algorithm for numerical simulation of Lévy stable stochastic process,” *Phys. Rev. E*, vol. 49, no. 5, pp. 4677–4683, 1994.
[15] A. Bunde and S. Halvin, Eds., *Fractals in Science*. New York: Springer-Verlag, 1994, ch. 5.
[16] J. Gillis and G. Weiss, “Expected number of distinct sites visited by a random walk with an infinite variance,” *J. Math. Phys.*, vol. 11, no. 4, pp. 1307–1312, 1970.
[17] T. Bäck and H.-P. Schwefel, “An overview of evolutionary algorithms for parameter optimization,” *Evol. Comput.*, vol. 1, pp. 1–23, 1993.
[18] X. Yao, G. Lin, and Y. Liu, “An analysis of evolutionary algorithms based on neighborhood and step size,” in *Proc. 6th Int. Conf. Evolutionary Programming*, 1997, pp. 297–307.
[19] C.-Y. Lee and X. Yao, “Evolutionary algorithm with adaptive Lévy mutations,” in *Proc. CEC2001*, 2001, pp. 568–575.
[20] X. Yao, Y. Liu, and G. Lin, “Evolutionary programming made faster,” *IEEE Trans. Evol. Comput.*, vol. 3, pp. 82–102, July 1999.
[21] D. Challet and Y.-C. Zhang, “Emergence of cooperation and organization in an evolutionary game,” *Physica A*, vol. 246, pp. 407–418, 1997.
[22] B. Gnedenko and A. Kolmogorov, *Limit Distribution for Sums of Independent Random Variables*. Cambridge, MA: Addition-Wesley, 1954.

Chang-Yong Lee received the B.S. degree in computer science and statistics from Seoul National University, Seoul, Korea, in 1983, and the Ph.D. degree from the University of Texas at Austin, in 1995, for research in computational and nuclear physics.

From 1996 to 1998, he was a Senior Researcher at the Basic Research Center, Electronics and Telecommunication Research Institute (ETRI), Korea. In 1998, he joined Kongju National University, Chungnam, South Korea, as an Assistant Professor, where, currently he is an Associate Professor in the Department of Industrial Information. His main research interests include the evolutionary computations, Internet ecology, and bioinformatics.

Dr. Lee will have his biographical profile included in the 7th edition of *Marquis Who’s Who in Science and Engineering*.

Xin Yao (M’91–SM’96–F’03) received the B.Sc. degree from the University of Science and Technology of China (USTC), Hefei, the M.Sc. degree from the North China Institute of Computing Technologies (NCI), Beijing, and the Ph.D. degree in computer science from the USTC, in 1982, 1985, and 1990, respectively, all in computer science.

He is currently a Professor of Computer Science and the Director of the Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), University of Birmingham, U.K., and a Visiting Professor at four other universities in China and Australia. He was a Lecturer, Senior Lecturer, and an Associate Professor at University College, University of New South Wales, the Australian Defence Force Academy (ADFA), Canberra, Australia, between 1992–1999. He held Postdoctoral Fellowships from the Australian National University (ANU), Canberra, and the Commonwealth Scientific and Industrial Research Organization (CSIRO), Melbourne, between 1990 and 1992. His major research interests include evolutionary computation, neural network ensembles, global optimization, computational time complexity, and data mining.

Dr. Yao is the Editor-in-Chief of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, an Associate Editor, and an Editorial Board Member of five other international journals, and the Chair of the IEEE Neural Networks Society Technical Committee on Evolutionary Computation. He is the recipient of the 2001 IEEE Donald G. Fink Prize Paper Award and has given more than 20 invited keynote and plenary speeches at various conferences. He has chaired/co-chaired more than 25 international conferences in evolutionary computation and computational intelligence, including CEC 1999, PPSN VI 2000, CEC 2002, and PPSN 2004.