Brief Papers

Combining Mutation Operators in Evolutionary Programming

Kumar Chellapilla

Abstract— **Traditional investigations with evolutionary programming (EP) for continuous parameter optimization problems have used a single mutation operator with a parameterized probability density function (pdf), typically a Gaussian. Using a variety of mutation operators that can be combined during evolution to generate pdf's of varying shapes could hold the potential for producing better solutions with less computational effort. In view of this, a linear combination of Gaussian and Cauchy mutations is proposed. Simulations indicate that both the adaptive and nonadaptive versions of this operator are capable of producing solutions that are statistically as good as, or better, than those produced when using Gaussian or Cauchy mutations alone.**

Index Terms— **Cauchy mutation, evolutionary programming, Gaussian mutation, variation operators.**

I. Introduction

EVOLUTIONARY algorithms, such as evolutionary programming (EP), evolution strategies (ES), and genetic algorithms (GA's), operate on a population of candidate solutions and rely on a set of variation operators to generate new offspring. Selection is used to probabilistically promote better solutions to the next generation and eliminate less-fit solutions. Conventional implementations of EP [1] and ES [2] for continuous parameter optimization use Gaussian mutations to generate offspring.

Recently, Cauchy mutations have been proposed for use with EP and ES [4], [5], inspired by fast simulated annealing [3]. The lognormal self-adaptation scheme [6], [7] was extended for evolving scale parameters for these Cauchy mutations. Empirical studies showed that on the tested multimodal functions with many local minima, EP using Cauchy mutations outperformed EP using Gaussian mutations, whereas on multimodal functions with few local minima the differences were not statistically significant [6], [7]. The fatter tails of the Cauchy distribution generate a greater probability of taking large steps, which could help in escaping local optima, and this has been offered as a possible explanation for its enhanced performance [6]. These studies, however, limited mutations to a single class of parameterized probability density functions (pdf's), namely either Gaussian or Cauchy.

In view of the fact that Cauchy mutations might be more useful in escaping local optima while Gaussian mutations provide relatively faster local convergence on convex functions [8], mutation strategies consisting of combinations of these two operators that can exploit their desirable properties have been proposed [9]–[11]. Two mutation operators are offered here that consist of linear combinations of the Gaussian and Cauchy mutation operators. Their performance both in terms of the quality of the solutions produced and the rate of optimization is empirically investigated on a suite of well-known test functions.

II. Background

Evolutionary algorithms address the problem of global optimization (minimization or maximization) in the presence of multiple local optima. A global minimization problem can be formalized as a pair (S, f), where $S \subseteq R^n$ is a bounded set on R^n and f: $S \rightarrow R$ is an n-dimensional real-valued function. The problem is to find a point $\boldsymbol{x}_{\min} \in S$ such that $f(\boldsymbol{x}_{\min})$ is a global minimum on S. More specifically, it is required to find an $\boldsymbol{x}_{\min} \in S$ such that

$$\forall \boldsymbol{x} \in S : f(x_{\min}) \leq f(x). \tag{1}$$

Here f does not need to be continuous but it must be bounded.

A. Conventional Evolutionary Programming

Conventional evolutionary programming (CEP) using the Gaussian mutation operator (GMO) and lognormal self-adaptive mutation for continuous parameter optimization is implemented as follows.

1) *Initialization*: Generate an initial population of μ individuals, and set the generation number k to one. Each individual is taken as a pair of real-valued vectors, $(\boldsymbol{x}_i, \boldsymbol{\sigma}_i)$, $\forall i \in \{1, \ldots, \mu\}$. The $\boldsymbol{x}_i$'s give the ith member's object variables and $\boldsymbol{\sigma}_i$'s the associated strategy parameters. The object variables typically are the real parameters to be optimized and the strategy parameters represent standard deviations of the associated Gaussian mutations [see Step 3)].
2) Evaluate the objective function, $f(\boldsymbol{x}_i)$, for each individual, $(\boldsymbol{x}_i, \boldsymbol{\sigma}_i)$, $\forall i \in \{1, \ldots, \mu\}$.
3) *Mutation*: Creates a single offspring $(\boldsymbol{x}'_i, \boldsymbol{\sigma}'_i)$ from each parent $(\boldsymbol{x}_i, \boldsymbol{\sigma}_i)$, $\forall i \in \{1, \ldots, \mu\}$ by

$$\boldsymbol{\sigma}'_i(j) = \boldsymbol{\sigma}_i(j) \exp(\tau N(0,1) + \tau' N_j(0,1)) \tag{2}$$
$$\boldsymbol{x}'_i(j) = \boldsymbol{x}_i(j) + \boldsymbol{\sigma}'_i(j) N_j(0,1) \tag{3}$$

for $j = 1, \ldots, n$, where $\boldsymbol{x}_i(j)$, $\boldsymbol{x}'_i(j)$, $\boldsymbol{\sigma}_i(j)$, and $\boldsymbol{\sigma}'_i(j)$ denote the jth component of the vectors $\boldsymbol{x}_i$, $\boldsymbol{x}'_i$, $\boldsymbol{\sigma}_i$, and

Manuscript received June 22, 1998; revised July 17, 1998.

The author is with the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093-4007 USA (e-mail: kchellap@ece.ucsd.edu).

Publisher Item Identifier S 1089-778X(98)08922-X.

$\boldsymbol{\sigma}'_i$, 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 a different random number is generated for each value of j. The factors τ and τ' are commonly set to $1/(\sqrt{2n})$ and $1/(\sqrt{2\sqrt{n}})$, respectively [2], [6].

4) *Fitness*: Calculate the objective function value $f(x_i)$ of each offspring $(\boldsymbol{x}'_i, \boldsymbol{\sigma}'_i)$, $\forall i \in \{1, \ldots, \mu\}$.
5) *Selection*: Conduct pairwise comparison over the union of parents $(\boldsymbol{x}_i, \boldsymbol{\sigma}_i)$ and offspring $(\boldsymbol{x}'_i, \boldsymbol{\sigma}'_i)$, $\forall i \in \{1, \ldots, \mu\}$. For each individual, q opponents are chosen randomly from all the parents and offspring with equal probability. For each comparison, if the individual's objective function value is no greater than the opponent's, it receives a "win." Select the μ individuals out of $(\boldsymbol{x}_i, \boldsymbol{\sigma}_i)$ and $(\boldsymbol{x}'_i, \boldsymbol{\sigma}'_i)$, $\forall i \in \{1, \ldots, \mu\}$, that have the most wins to be parents of the next generation.
6) Stop if the halting criterion is satisfied; otherwise, increment the generation number, $k = k + 1$, and go to Step 3).

In the current simulations, since the amount of computing time required to obtain solutions of a desired quality were not known *a priori*, the halting criterion was taken to be a certain maximum number of generations $(k_{\max})$.

Self-adaptive update schemes, in conjunction with selection, favor strategy parameters that yield a higher probability of improving the quality of the parent. For convex objective functions, when using mutations drawn from a zero mean symmetric density function, this probability of improvement peaks at 0.5 for infinitesimally small step sizes. Thus self-adaptation can be biased toward lowering the standard deviations too quickly resulting in stagnation (described as an extremely low rate of convergence). As a safeguard against such stagnation, a lower bound for the strategy parameters has been proposed [2], [6], used [4], [5], [7], and investigated [12]. When a lower bound b is used, all $\boldsymbol{\sigma}'_i(j)$'s that fall below b in (2) are reset to b.

B. Mutation Operators

CEP with the Cauchy mutation operator (CMO) is obtained by replacing the object variable update equation given in (3) with

$$\text{CMO: } \boldsymbol{x}'_i(j) = \boldsymbol{x}_i(j) + \boldsymbol{\sigma}'_i(j) C_j(0,1) \tag{4}$$

where $C(0,1)$ denotes a Cauchy random number centered at zero with a scale parameter of one.

Two new mutation operators are introduced, namely *mean* and *adaptive mean* mutation operators (MMO's and AMMO's, respectively). Both of these operators consist of a linear combination of Gaussian and Cauchy mutations.

The MMO uses two random variables (RV's). The first is distributed $N(0,1)$, and the second is distributed $C(0,1)$. The mean of samples from these two RV's is scaled by the self-adaptive parameter $\boldsymbol{\sigma}'_i(j)$ and used to perturb the jth component of the parent to obtain the jth component of the offspring. The object variable update equation for the MMO

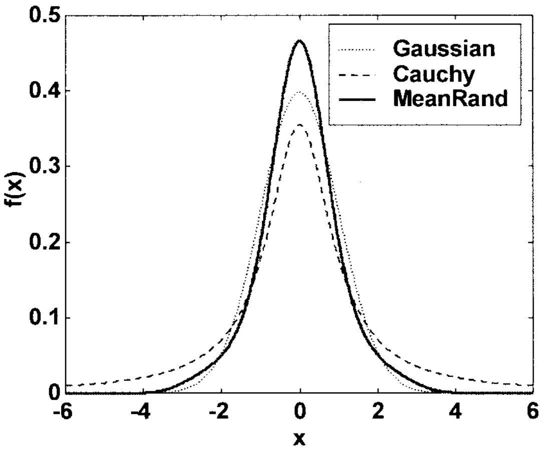

Fig. 1. The pdf of the mean random numbers in comparison with the standard Gaussian and Cauchy pdf's. Among the three pdf's there exists a tradeoff between the probabilities of generating very small (0.0–0.6), small (0.6–1.2), medium (1.2–2.0), large (2.0–4.8), and very large (>4.8) mutations.

TABLE I
PROBABILITY OF GENERATING DIFFERENT DEGREES OF MUTATIONS USING THE GMO'S, CMO'S, AND MMO'S

Degree of Mutation	Probability of generating mutations		
	Largest	Middle	Smallest
very small (0.0-0.6)	MMO	GMO	CMO
small (0.6-1.2)	GMO	MMO	CMO
medium (1.2-2.0)	GMO	CMO	MMO
large (2.0-4.8)	CMO	MMO	GMO
very large (>4.8)	CMO	GMO	MMO

is given by

$$\boldsymbol{x}'_i(j) = \boldsymbol{x}_i(j) + 0.5\boldsymbol{\sigma}'_i(j)(N_j(0,1) + C_j(0,1)). \tag{5}$$

These mean random numbers follow a pdf given by the convolution of the Gaussian and Cauchy pdf's followed by scaling. Mathematically

$$\begin{aligned}\text{PDF}_{\text{Mean}}(x) &= \text{PDF}_{N(0,1)}(2x) * \text{PDF}_{C(0,1)}(2x)\\ &= \left\{\frac{1}{\sqrt{\pi}}\exp(-2x^2)\right\} * \left\{\frac{2}{\pi}\left(\frac{1}{1+4x^2}\right)\right\}\end{aligned} \tag{6}$$

where $*$ denotes the convolution operator. Fig. 1 shows the pdf of the mean random numbers. The standard Gaussian and Cauchy pdf's are also plotted for comparison. For analysis, based on Fig. 1, the range of mutations (absolute values) in $[0,\infty)$ is split into five categories, very small (0–0.6), small (0.6–1.2), medium (1.2–2), large (2–4.8), and very large ($>$4.8). There exist tradeoffs among the three distributions between the probabilities of generating very low, low, medium, large, and very large mutations. These are summarized in Table I. In comparison with Gaussian mutations, the mean of a Gaussian and a Cauchy generates more very small and large mutations. In comparison with Cauchy mutations, it generates more very small and small mutations. Thus the MMO generally produces mutations that are larger than Gaussian mutations but smaller than Cauchy mutations.

During evolution, the shape of the pdf that produces mean mutations is fixed and the pdf parameters are self-adapted. Self-adaptation of the pdf shape also, along with its parameters, could make the self-adaptation scheme more robust to the type of objective function being optimized. Therefore, the MMO is extended to an AMMO where the object variable update equation is given by

$$\begin{aligned}\boldsymbol{x}_i'(j) &= \boldsymbol{x}_i(j) + \boldsymbol{\sigma}_{1i}'(j)N_j(0,1) + \boldsymbol{\sigma}_{2i}'(j)C_j(0,1)\\ &= \boldsymbol{x}_i(j) + \boldsymbol{\alpha}_i'(j)(C_j(0,1) + \boldsymbol{\beta}_i'(j)N_j(0,1)). \end{aligned} \tag{7}$$

The AMMO has two sets of self-adaptive parameters $\boldsymbol{\sigma}_{1i}'(j)$ and $\boldsymbol{\sigma}_{2i}'(j)$ that act as standard deviations and scale parameters of the Gaussian and Cauchy parts, respectively. The mutation step can be rewritten in terms of $\boldsymbol{\alpha}_i'(j)$ and $\boldsymbol{\beta}_i'(j)$ as given in (7), wherein $\boldsymbol{\alpha}_i'(j)$ plays the role of the overall scaling parameter and $\boldsymbol{\beta}_i'(j)$ determines the shape of the pdf. Mathematically, $\boldsymbol{\alpha}_i'(j) = \boldsymbol{\sigma}_{2i}'(j)$ and $\boldsymbol{\beta}_i'(j) = \boldsymbol{\sigma}_{1i}'(j)/\boldsymbol{\sigma}_{2i}'(j)$. For low values of $\boldsymbol{\beta}_i'(j)$ the pdf resembles a Cauchy, whereas for large values of $\boldsymbol{\beta}_i'(j)$ it resembles a Gaussian. Thus with the variation of the $\boldsymbol{\alpha}_i'(j)$ and $\boldsymbol{\beta}_i'(j)$ parameters (through the self-adaptation of $\boldsymbol{\sigma}_{1i}'(j)$ and $\boldsymbol{\sigma}_{2i}'(j)$) it is possible to generate a large number of different pdf's that have a shape between a Gaussian and a Cauchy. Since the AMMO contains twice as many strategy parameters as the other operators, however, the time taken to find good strategy parameters through self-adaptation might be expected to increase.

III. Method

CEP, with the above-described operators, was tested on a set of nine well-investigated function minimization problems

$$f_1(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 \tag{8}$$

$$f_2(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e \tag{9}$$

$$f_3(\mathbf{x}) = \sum_{i=1}^{n-1}\left\{100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right\} \tag{10}$$

$$f_4(\mathbf{x}) = \sum_{i=1}^{n} x_i^4 + U([0,1)) \tag{11}$$

where $U([0,1))$ is a uniform random variable in $[0,1)$

$$f_5(\mathbf{x}) = \sum_{i=1}^{n}\left\{x_i^2 - 10\cos((2\pi x_i) + 10)\right\} \tag{12}$$

$$f_6(\mathbf{x}) = \sum_{i=1}^{n}|x_i| + \prod_{i=1}^{n}|x_i| \tag{13}$$

$$f_7(\mathbf{x}) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2 \tag{14}$$

$$f_8(\mathbf{x}) = \max\{|x_i|, i = 1, \ldots, n\} \tag{15}$$

$$f_9(\mathbf{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1. \tag{16}$$

Function f_1 is the sphere function, and f_2 is a modified version of the Ackley function [13], with the term $20+e$ added to move the global optimum function value to zero. Function f_3 is the extended Rosenbrock function [6], f_4 is the noisy quadratic function, and f_5 is the Rastrigin function [14]. Functions f_6, f_7, and f_8, are test problems numbered 2.22, 1.2, and 2.21 from [6, pp. 341, 326, and 340], respectively. Function f_9 is the Griewank function [15].

Simulations were conducted with Gaussian, Cauchy, mean, and adaptive mean mutation operators using a population size of 50 and an opponent size $q = 10$. Alternative choices could be offered but these follow typical implementations. For all the nine benchmark functions, n was set to 30. For functions f_1 and f_2, all components were initialized uniformly in $[-100, 100]$. For f_3, f_4, f_5, f_6, f_7, f_8, and f_9, all components were initialized in $[-30, 30]$,$[-1.28, 1.28]$, $[-5.12, 5.12]$, $[-10, 10]$, $[-100, 100]$, $[-100, 100]$, and $[-600, 600]$, respectively. The self-adaptive parameters were initialized to three. The termination criterion varied with the function being optimized. The EP trials for f_1 and f_2 were terminated after 3000 generations and for f_3–f_9 were terminated after 5000 generations. The above simulation parameters were chosen following [4], [5], [7], and [9]–[11].

Two sets of 50 independent trials were conducted on each of the nine test functions for each of the four mutation operators (GMO, CMO, MMO, and AMMO). The first set of experiments did not use a lower bound for the strategy parameters, whereas the second set used a lower bound of $b = 0.0001$ [4], [5].

IV. Results

The mean best scores and the associated results of the t-test for statistical significance [16], taken over 50 independent trials for f_1–f_9 with and without the lower bound on the strategy parameters, are presented in Table II. Based on these results, EP with the four different operators was ranked as shown in the last two columns of Table II (G, C, M, and A representing EP with the GMO, CMO, MMO, and AMMO, respectively). The operator ranked first (shown to the left of the table column) had the lowest mean (i.e., best) function value. As an example, the entry "A, (M, G, C)" for the results on f_1 without a lower bound indicates that EP with the AMMO operators generated solutions that had the lowest mean function value and that these values were unambiguously statistically significantly better than those generated by EP with the MMO, GMO, or CMO. The keyword unambiguously means that the EP with AMMO results were statistically significantly better in all comparisons with the other operators. The parentheses grouping M, G, and C indicate that the pair-wise t-test results for the

TABLE II
THE MEAN BEST SCORES AND THE UNAMBIGUOUS RANK RESULTS OF THE t-TEST FOR STATISTICAL SIGNIFICANCE [16] FOR THE EP TRIALS WITH THE DIFFERENT MUTATION OPERATORS, NAMELY GMO'S, CMO'S, MMO'S, AND AMMO'S. BASED ON THE t-TEST RESULTS, EP WITH THE FOUR DIFFERENT OPERATORS WAS RANKED AS SHOWN IN THE LAST TWO COLUMNS (G, C, M, AND A REPRESENTING EP WITH THE GMO, CMO, MMO, AND AMMO, RESPECTIVELY). THE OPERATOR RANKED FIRST (SHOWN TO THE LEFT OF THE TABLE COLUMN) HAD THE LOWEST MEAN FUNCTION VALUE AT THE END OF THE TRIAL. AS AN EXAMPLE, THE ENTRY "A,(M,G,C)" FOR THE RESULTS ON F_1 WITHOUT A LOWER BOUND INDICATES THAT EP WITH THE AMMO OPERATOR GENERATED SOLUTIONS THAT HAD THE LOWEST MEAN FUNCTION VALUE AND THAT THESE VALUES WERE STATISTICALLY SIGNIFICANTLY UNAMBIGUOUSLY BETTER THAN THOSE GENERATED BY EP WITH THE MMO'S, GMO'S, OR CMO'S. THE KEYWORD UNAMBIGUOUSLY MEANS THAT THE EP WITH AMMO RESULTS WERE STATISTICALLY SIGNIFICANTLY BETTER IN ALL COMPARISONS WITH THE OTHER OPERATORS. THE PARENTHESES GROUPING M, G, AND C INDICATE THAT THE t-TEST RESULTS FOR EP WITH MMO, GMO, AND CMO WERE NOT ALL STATISTICALLY SIGNIFICANTLY DIFFERENT

Function	Mean Best Fitness								Unambiguous Rank ordering based on the t-test result	
	GMO		CMO		MMO		AMMO			
	$b = 0$	b = 1e-4	$b = 0$	b = 1e-4	$b = 0$	b = 1e-4	$b = 0$	b = 1e-4	$b = 0$	b = 1e-4
f_1	83.51	3.09e-7	104.74	3.07e-6	41.12	9.81e-7	38.22	1.61e-6	A,(M,G,C)	G,M,C,A
f_2	10.47	9.10e+0	4.52	1.30e-3	3.75	7.49e-4	3.41	9.43e-4	A,(M,C),G	M,A,C,G
f_3	2.33e4	8.67e+1	4.37e3	1.14e+2	2.87e3	6.38e+1	1.45e3	1.44e+2	(A,M,C,G)	(M,G,C),A
f_4	44.15	1.22e+1	42.01	9.42e+0	33.98	9.53e+0	14.86	9.64e+0	A,(M,C,G)	(C,M,A,G)
f_5	113.72	1.20e+2	55.82	4.73e+0	46.96	9.52e+0	52.59	1.19e+1	(A,M,C),G	(C,M),A,G
f_6	9.74e3	1.99e-3	4.23e3	5.87e-3	5.55e3	3.23e-3	81.97	3.99e-3	A,(C,M,G)	G,M,A,C
f_7	2920.63	1.76e+1	2740.70	5.78e+0	2812.26	1.18e+1	1545.01	6.12e-1	A,(C,M,G)	A,C,(M,G)
f_8	5.89	5.18e+0	5.89	6.60e-1	6.31	1.88e+0	3.14	3.23e-1	A,(G,C,M)	A,C,M,G
f_9	6.05	2.52e-7	7.08	2.20e-6	3.84	6.99e-7	1.85	1.02e-6	A,(M,G,C)	A,G,M,C

EP with MMO, GMO, and CMO were not all statistically significantly different.

Space considerations preclude presenting all of the results obtained. Fig. 2(a)–(c) graphs the rate of optimization of EP with the four different mutation operators without a lower bound for the functions f_1, f_2, and f_8, respectively. Fig. 2(d) and (e) presents the corresponding results with a lower bound for f_1 and f_9, respectively. The rate of optimization curves for the other functions were similar and are available on line[1] for the interested reader.

When a lower bound was not used, EP with AMMO showed faster convergence than all other operators and found solutions that were as good as (f_3, and f_5) or statistically significantly better (f_1, f_2, f_4, f_6, f_7, f_8, and f_9) than those found by the other operators.

When a lower bound was used, the optimization trajectories for f_1–f_6 and f_9 saturated as their strategy parameters reached the lower bound. In these trials, the solutions found were several orders of magnitude better than those found without a lower bound (see Table II). In trials wherein the strategy parameters did not saturate (f_7 and f_8), the differences between the performance of EP with and without a lower bound were not statistically significant.

[1]All the figures, tables, and results presented in this paper along those omitted due to space constraints are available at http://vision.ucsd.edu/~kchellap/IEEETECV1N4.ps.

In the trials with a lower bound, even though EP with AMMO showed faster convergence it stagnated earlier. This was caused by the restrictions that the lower bound generated on the allowed shape of the pdf of the AMMO mutations.

The $\beta_i'(j)$ parameter [see (7)] yields insight as to which form of the pdf produced beneficial mutations during different phases of evolution. Large values of $\beta_i'(j)$ ($\gg 1$) imply a pdf that closely resembles a Gaussian pdf, whereas small values of $\beta_i'(j)$ ($\ll 1$) imply a pdf that closely resembles a Cauchy. A value of $\beta_i'(j) = 1$ represents the midpoint, corresponding to the MMO. The $\beta_i'(j)$ parameters started out at one (as the strategy parameters $\sigma_{1i}(j)$ and $\sigma_{2i}(j)$ were both initialized to three) in each trial. For all test problems when a lower bound was not used, there was a gradual increase in $\beta_i'(j)$ from one to a value between four and ten during the first 1000–2000 generations. After the initial phase of growth, the $\beta_i'(j)$ value oscillated about the final value. Thus, the AMMO mutations followed a pdf whose form resembled that of the MMO at the beginning of the search and gradually resembled the Gaussian. Pure Cauchy mutations were found to be less efficient than MMO and Gaussian mutations at the beginning of the search. Moreover, as solutions were obtained that were closer to the global optimum, Gaussian mutations were increasingly preferred over Cauchy mutations.

Fig. 3 shows the mean $\beta_i'(j)$ value as a function of the number of generations (averaged over all the dimensions and the 50 trials) for the sphere and Rosenbrock functions both with

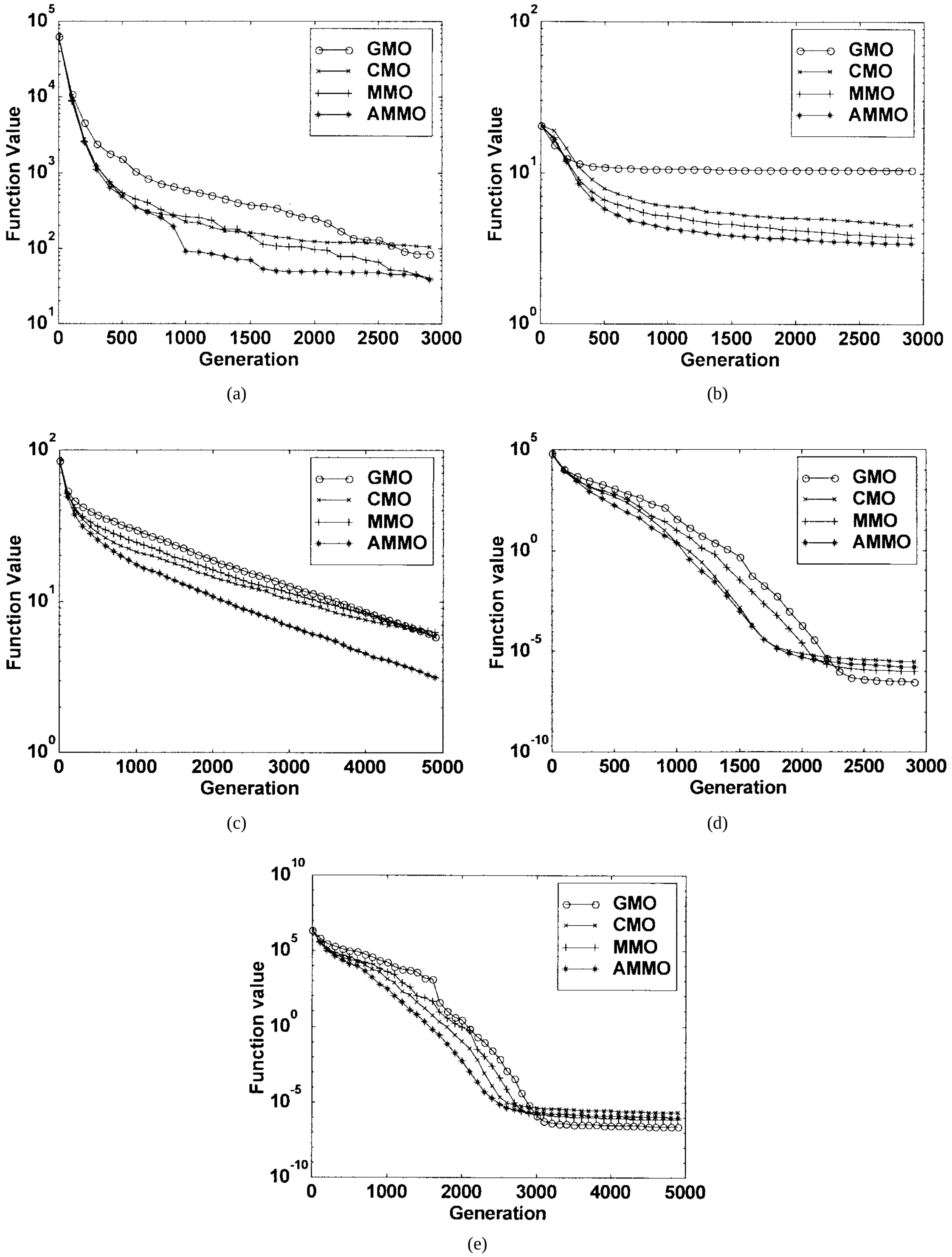

Fig. 2. The rate of optimization (averaged over 50 trials) of EP with the four mutation operators for the (a) sphere function (f_1), (b) Ackley function (f_2), (c) function 2.21 (f_8) from [6], (d) sphere function (f_1) when using a lower bound of 0.0001 on the strategy parameters, and (e) Griewank function (f_9) when using a lower bound of 0.0001. Without a lower bound, EP with AMMO shows faster convergence than the other operators and the final mean best solution found by the AMMO is as good as or statistically significantly better than those found by using the other operators. With a lower bound, all the function value trajectories, with the exception of those for f_8, saturate as their strategy parameters reach the lower bound. With a lower bound, however, even though EP with AMMO shows faster convergence than EP with any of the other operators, it stagnates earlier. The addition of a lower bound enhanced the quality of the final solutions by several orders of magnitude for most of the examined functions.

and without a lower bound. When using a lower bound, as both the $\sigma_{1i}(j)$ and $\sigma_{2i}(j)$ parameters reached that lower bound the $\beta'_i(j)$ value moved toward a value of one, and degraded the performance of the AMMO operator. When all the strategy parameters reached the lower bound, the AMMO operator became equivalent to the MMO, and the GMO was better suited for local optimization due to its larger probability of generating smaller mutations in comparison with the MMO and CMO.

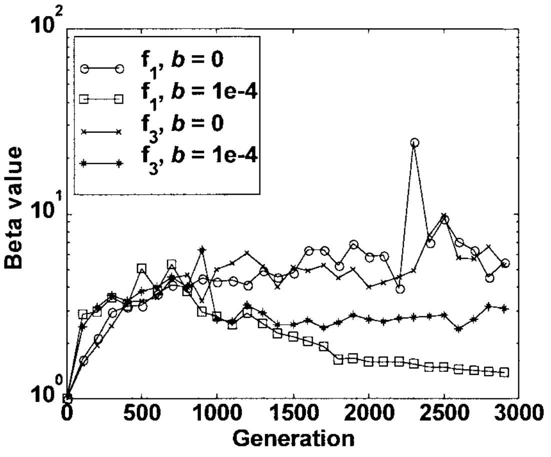

Fig. 3. The mean trajectory of the best member's $\beta_i'(j)$ (averaged over all dimensions, j, and over the 50 trials) parameter during the EP with AMMO trials for the sphere (f_1) and Rosenbrock (f_3) functions with and without a lower bound on the strategy parameters. Without a lower bound (i.e., $b = 0$), the $\beta_i'(j)$ values grew during the first 1000–2000 generations from one (which was the value on initialization) to a value between four and ten (with the exception of the outlier for $f_1, b = 0$). The corresponding $\beta_i'(j)$ values when using a lower bound showed a similar increase early in the trials, but were later forced to lower values as the strategy parameters became close to the lower bound. If evolution were carried on for a sufficiently long time and a lower bound was used, all the strategy parameters would reach the lower bound and the $\beta_i'(j)$ values would converge to 1.0. Similar results were observed in experiments with the other test functions.

V. Summary

Without a lower bound, the MMO and AMMO mutations consistently produced solutions that were as good as or better than those produced by Gaussian and Cauchy operators acting alone. In seven of the nine test functions, the differences between the quality of the solutions produced by the AMMO and those produced by all other operators (GMO, CMO, and MMO) were statistically significant. In addition, in eight of the nine test functions, the MMO produced solutions that were statistically better than those produced by using Gaussian mutations alone.

With a lower bound, on eight of the nine test problems, either the MMO or the AMMO mutations (or both) consistently generated solutions that were as good as or statistically significantly better than those produced by the GMO or CMO mutations. The degradation in performance of AMMO relative to that of the other operators when using a lower bound was caused by the resulting constraints on the ratio of the Gaussian and Cauchy parts of the AMMO mutations toward the end of the trial. Such lower bounds must be designed in view of their interactions with the mutation operators being used. The addition of a lower bound on the strategy parameters did not statistically significantly degrade solution quality on any of the test functions as compared without using a lower bound. In most cases, the lower bound actually enhanced the solution quality by several orders of magnitude.

The success of the AMMO could be attributed to its ability to adapt the shape of the pdf that generates the mutations during the trial. These results indicate that self-adapting not only the scaling factors but also the shape of the mutation pdf can significantly enhance the quality of the solutions obtained, and reduce the time taken to reach such solutions. Areas for future investigation include other methods of combining mutation operators and designing dynamic lower bounds that can better interact with the utilized mutation operators.

Acknowledgment

The author would like to thank D. B. Fogel for his encouragement, helpful comments, and suggestions. The author would also like to thank the anonymous reviewers for their helpful suggestions.

References

[1] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. Piscataway, NJ: IEEE Press, 1995.
[2] T. Bäck, *Evolutionary Algorithms in Theory and Practice*. New York: Oxford Univ. Press, 1996.
[3] H. H. Szu and R. L. Hartley, "Nonconvex optimization by fast simulated annealing," *Proc. IEEE*, 1987, vol. 75, pp. 1538–1540.
[4] X. Yao and Y. Liu, "Fast evolutionary programming," in *Proc. Fifth Annual Conf. Evolutionary Programming (EP'96)*, L. J. Fogel, P. J. Angeline, and T. Bäck, Eds. Cambridge, MA: MIT Press, 1996, pp. 451–460.
[5] X. Yao and Y. Liu, "Fast evolution strategies," in *Evolutionary Programming VI: Proc. of the Sixth Int. Conf. Evolutionary Programming (EP'97)*, P. J. Angeline, R. G. Reynolds, J. R. McDonnell, and R. Eberhart, Eds. Berlin, Germany: Springer, 1997, pp. 151–161.
[6] H.-P. Schwefel, *Evolution and Optimum Seeking*. New York: Wiley, 1995.
[7] N. Saravanan, D. B. Fogel, and K. M. Nelson, "A comparison of methods for self-adaptation in evolutionary algorithms," *Biosystems*, vol. 36, pp. 157–166, 1995.
[8] G. Rudolph, "Local convergence rates of simple evolutionary algorithms with Cauchy mutations," *IEEE Trans. Evol. Comput.*, vol. 1, pp. 249–258, Nov. 1997.
[9] X. Yao, G. Lin, and Y. Liu, "An analysis of evolutionary algorithms based on neighborhood and step sizes," in *Evolutionary Programming VI: Proc. 6th Int. Conf. Evolutionary Programming (EP'97)*, P. J. Angeline, R. G. Reynolds, J. R. McDonnell, and R. Eberhart, Eds. Berlin, Germany: Springer, 1997, pp. 297–307.
[10] N. Saravanan and D. B. Fogel, "Multi-operator evolutionary programming: A preliminary study on function optimization," in *Evolutionary Programming VI: Proc. 6th. Int. Conf. Evolutionary Programming (EP'97)*, P. J. Angeline, R. G. Reynolds, J. R. McDonnell, and R. Eberhart, Eds. Berlin, Germany: Springer, 1997, pp. 215–221.
[11] K. Chellapilla and D. B. Fogel, "Two new mutation operators for enhanced search and optimization in evolutionary programming," in *Proc. SPIE's Int. Symp. Optical Science, Engineering, and Instrumentation, Conference 3165: Applications of Soft Computing*, B. Bosacchi, J. C. Bezdek, and D. B. Fogel, Eds. Billingham, WA: SPIE, 1997, pp. 260–269.
[12] K.-H. Liang, X. Yao, Y. Liu, C. Newton, and D. Hoffman, "An experimental investigation of self-adaptation in evolutionary programming," in *Proc. 7th Annu. Conf. Evolutionary Programming (EP'98)*, V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, Eds. Berlin, Germany: Springer, 1998, pp. 291–300.
[13] D. H. Ackley, *A Connectionist Machine for Genetic Hillclimbing*. Boston, MA: Kluwer, 1987.
[14] A. Törn and A. Zilinskas, *Global Optimization* (Lecture Notes in Computer Science, vol. 350). Berlin, Germany: Springer-Verlag, 1989.
[15] A. O. Griewank, "Generalized descent for global optimization," *JOTA* vol. 34, pp. 11–39, 1981.
[16] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*. New York: Cambridge Univ. Press, 1992.