This article was downloaded by: [National Chiao Tung University 國立交通大學]
On: 27 April 2014, At: 22:16
Publisher: Taylor & Francis
Informa Ltd Registered in England and Wales Registered Number: 1072954 Registered office: Mortimer House, 37-41 Mortimer Street, London W1T 3JH, UK

Engineering Optimization
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/geno20

A robust evolutionary algorithm for global optimization

Jinn-Moon Yang [a] , Chin-Jen Lin [b] & Cheng-Yan Kao [b]

[a] Department of Biological Science and Technology and Institute of Bioinformatics , National Chiao Tung University , Hsinchu, 30050, Taiwan

[b] Department of Computer Science and Information Engineering , National Taiwan University , Taipei, 106, Taiwan

Published online: 17 Sep 2010.

To cite this article: Jinn-Moon Yang , Chin-Jen Lin & Cheng-Yan Kao (2002) A robust evolutionary algorithm for global optimization, Engineering Optimization, 34:5, 405-425, DOI: 10.1080/03052150214019

To link to this article: http://dx.doi.org/10.1080/03052150214019

PLEASE SCROLL DOWN FOR ARTICLE

Taylor & Francis makes every effort to ensure the accuracy of all the information (the "Content") contained in the publications on our platform. However, Taylor & Francis, our agents, and our licensors make no representations or warranties whatsoever as to the accuracy, completeness, or suitability for any purpose of the Content. Any opinions and views expressed in this publication are the opinions and views of the authors, and are not the views of or endorsed by Taylor & Francis. The accuracy of the Content should not be relied upon and should be independently verified with primary sources of information. Taylor and Francis shall not be liable for any losses, actions, claims, proceedings, demands, costs, expenses, damages, and other liabilities whatsoever or howsoever caused arising directly or indirectly in connection with, in relation to or arising out of the use of the Content.

This article may be used for research, teaching, and private study purposes. Any substantial or systematic reproduction, redistribution, reselling, loan, sub-licensing, systematic supply, or distribution in any form to anyone is expressly forbidden. Terms & Conditions of access and use can be found at http://www.tandfonline.com/page/terms-and-conditions

Eng. Opt., 2002, Vol. 34(5), pp. 405–425

Taylor & Francis
Taylor & Francis Group

A ROBUST EVOLUTIONARY ALGORITHM FOR GLOBAL OPTIMIZATION

JINN-MOON YANG[a,*], CHIN-JEN LIN[b] and CHENG-YAN KAO[b]

[a]*Department of Biological Science and Technology and Institute of Bioinformatics, National Chiao Tung University, Hsinchu, 30050, Taiwan;* [b]*Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan*

(Received 20 July 2001; In final form 25 February 2002)

This paper studies an evolutionary algorithm for global optimization. Based on family competition and adaptive rules, the proposed approach consists of global and local strategies by integrating decreasing-based mutations and self-adaptive mutations. The proposed approach is experimentally analyzed by showing that its components can integrate with one another and possess good local and global properties. Following the description of implementation details, the approach is then applied to several widely used test sets, including problems from international contests on evolutionary optimization. Numerical results indicate that the new approach performs very robustly and is competitive with other well-known evolutionary algorithms.

Keywords: Evolutionary algorithms; Family competition; Multiple mutation operators; Adaptive rules; Global optimization

Downloaded by [National Chiao Tung University] at 22:16 27 April 2014

1 INTRODUCTION

Many practical applications from engineering, natural sciences, economics, and business can be formulated as global optimization problems, whose objective functions often possess many local minima in the region of interest. Global optimization is the task of finding the absolutely best minimum among all local minima of optimizing an objective function. In general, it is difficult to obtain an exact solution for such problems.

A general form of the global optimization problem is stated as follows [25]: Given a function $f: \mathcal{M} \subseteq \mathbb{R}^n \to \mathbb{R}$, $\mathcal{M} \neq \emptyset$, for $x^* \in \mathcal{M}$, the value $f^* \equiv f(x^*) > -\infty$ is called a *global minimum* if and only if

$$f(x^*) \leq f(x), \quad \forall x \in \mathcal{M} \tag{1}$$

Then x^* is a *global minimum point*, f is called the *objective function*, and the set $\mathcal{M}$ is called the *feasible region*. Note that finding a global maximum is equivalent to finding a minimum of $-f(x)$. Therefore, this work only considers the problem of minimization. In addition, this paper only considers the unconstrained optimization problem, that is, $\mathcal{M} = \mathbb{R}^n$.

* Corresponding author. E-mail: moon@cc.nctu.edu.tw

ISSN 0305-215X print; ISSN 1029-0273 online © 2002 Taylor & Francis Ltd
DOI: 10.1080/0305215021000033726

Many traditional approaches solve global optimization by developing an analogue that closely resembles the original function and is solvable by traditional mathematical strategies such as linear and nonlinear programming. Such techniques often require simplification of the original problem which may drive the computed solution far away from the global optimum [25]. To obtain a global solution, several probabilistic methods have been proposed. Among them, evolutionary algorithms are very promising.

Currently, there are three main independently developed but strongly related implementations of evolutionary algorithms: genetic algorithms [14], evolution strategies [23], and evolutionary programming [11]. These perform differently when applied to global optimization. For genetic algorithms, both practice and theory [9, 15, 19] entail the disadvantages of applying a binary-represented implementation to global optimization. The coding function of genetic algorithms may introduce an additional multimodality, making the combined objective function more complex than the original function. To achieve better performance, real-coded genetic algorithms [9, 17, 4] have been introduced. In contrast, evolution strategies and evolutionary programming mainly use real-valued representations and focus on self-adaptive Gaussian mutations. This type of mutation has succeeded in continuous optimization and has been widely regarded as a good operator for local searches. Unfortunately, experiments [30, 33] show that self-adaptive Gaussian mutation leaves individuals trapped near local optima for rugged functions.

Because none of these three types of original evolutionary algorithms are very efficient tools, many modifications have been proposed to improve solution quality and to speed up convergence. In particular, a recent trend [13, 34] is to incorporate local search techniques into evolutionary algorithms. Such a hybrid approach can combine the merits of an evolutionary algorithm with those of a local search technique. Generally speaking, a hybrid approach usually can make a better tradeoff between computational cost and the global optimality of the solution. However, for existing methods, local search techniques and genetic operators often work separately during the search process.

Another technique is to use multiple genetic operators [22, 28]. This approach works by assigning a list of parameters to determine the probability of using each operator. Then, an adaptive mechanism is applied to change these probabilities to reflect the performance of the operators. The main disadvantage of this method is that the mechanism for adapting the probabilities may mislead evolutionary algorithms toward local optima.

To further improve the above approaches, in this paper a new method called family competition evolutionary algorithm (FCEA) is proposed for solving global optimization problems. FCEA is a multi-operator approach which combines three mutation operators: decreasing-based Gaussian mutation, self-adaptive Gaussian mutation, and self-adaptive Cauchy mutation. It incorporates family competition and adaptive rules [29, 35] for controlling step sizes to construct the relationship among these three operators. In order to balance exploration and exploitation, each of these operators is designed to compensate for the disadvantages of the other. In addition, FCEA markedly differs from previous approaches because these mutation operators are sequentially applied with equal probability 1.

To the best of the authors' knowledge, FCEA is the first successful approach to integrate self-adaptive mutations and decreasing-based mutations using family competition principles.

The rest of this paper is organized as follows. Section 2 introduces the evolutionary nature of FCEA. Next, Section 3 presents ten widely used testing functions, then investigates the main characteristics of FCEA. It demonstrates experimentally how FCEA balances the trade-off between exploitation and exploration of the search. Section 4 compares FCEA with other methods. In Section 5 FCEA is tested on functions from two international contests on evolutionary optimization. Conclusions are drawn in Section 6.

Downloaded by [National Chiao Tung University] at 22:16 27 April 2014

2 THE FAMILY COMPETITION EVOLUTIONARY ALGORITHM

This section presents details of the family competition evolutionary algorithm (FCEA) for global optimization. The basic structure of the FCEA is as follows (Fig. 1): N solutions are randomly generated as the initial population. Then FCEA enters the main evolutionary loop, consisting of three stages in every iteration: decreasing-based Gaussian mutation, self-adaptive Cauchy mutation, and self-adaptive Gaussian mutation. Each stage is realized by generating a new quasi-population (with N solutions) as the parent of the next stage. As shown in Figure 1, these stages differ only in the mutations used and in some parameters. Hence a general procedure "FC_adaptive" is used to represent the work done by these stages.

The FC_adaptive procedure employs four parameters, namely, the parent population (P, with N solutions), mutation operator (M), selection method (S), and family competition length (L), to generate a new quasi-population. The main work of FC_adaptive is to produce offspring and then conduct the family competition. Each individual in the population sequentially becomes the "family father". With a probability p_c, this family father and another solution randomly chosen from the rest of the parent population are used as parents to do a recombination operation. Then the new offspring or the family father (if the recombination is not conducted) is operated on by a mutation. For each family father, such a procedure is repeated L times. Finally L children are produced but only the one with the best objective value survives. Since this creates L children from one "family father" and performs a selection, this is a family competition strategy. This was thought to be a good way to avoid premature convergence and also to keep the spirit of local searches, and results agree.

After the family competition, there are N parents and N children left. Based on different stages (or the parameter S of the FC_adaptive procedure), various ways are used to obtain a new quasi-population with N individuals. For both Gaussian and Cauchy self-adaptive

Downloaded by [National Chiao Tung University] at 22:16 27 April 2014

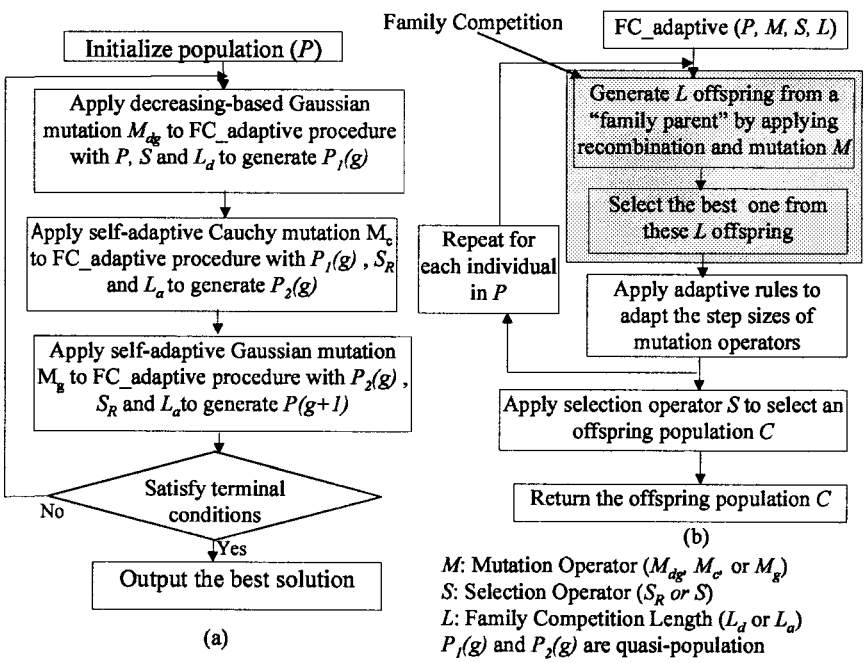


FIGURE 1 Overview of the algorithm: (a) FCEA (b) FC_adaptive procedure.

mutations, in each pair of father and child the individual with the better objective value survives. This procedure is called "family selection". On the other hand, "population selection" chooses the best N individuals from all N parents and N children. With a probability P_{ps}, FCEA applies population selection to speed up the convergence when the decreasing-based Gaussian mutation is used. For the probability $(1 - P_{ps})$, family selection is still considered. In order to reduce the ill effects of greediness on this selection, the initial P_{ps} is set to 0.05, but it is changed to 0.5 when the mean step size of self-adaptive Gaussian mutation is larger than that of decreasing-based Gaussian mutation. Note that the population selection is similar to $(\mu + \mu)$-ES in the traditional evolution strategies. Hence, through the process of selection, the FC_adaptive procedure forces each solution of the starting population to have one final offspring. Note that we create LN offspring in the procedure FC_adaptive but the size of the new quasi-population remains the same, N.

For all three mutation operators, different parameters are assigned to control their performance. Such parameters must be adjusted through the evolutionary process. They are modified first when mutations are applied. Then, after the family competition is complete, parameters are adapted again to better reflect the performance of the whole FC_adaptive procedure.

Regarding chromosome representation, each solution of a population is presented as a set of four n-dimensional vectors $(x^i, \sigma^i, \nu^i, \psi^i)$, where n is the number of variables and $i = 1, \ldots, N$. The vector x is the main variable to be optimized; and σ, ν, and ψ are the step-size vectors of decreasing-based mutations, self-adaptive Gaussian mutation, and self-adaptive Cauchy mutation, respectively. In other words, each solution x is associated with some parameters for step-size control. In this paper, the initial x is randomly chosen from the feasible search space while the initial step sizes σ, ν, and ψ vary for different problems. For easy description of operators, $a = (x^a, \sigma^a, \nu^a, \psi^a)$ is used to represent the "family father" and $b = (x^b, \sigma^b, \nu^b, \psi^b)$ as another parent (only for the recombination operator). The offspring of each operation is represented as $c = (x^c, \sigma^c, \nu^c, \psi^c)$. Also, the symbol x_j^d is used to denote the jth component of an individual d, $\forall j \in \{1, \ldots, n\}$. The rest of this section explains each important component of the FC_adaptive procedure: recombination operators, mutation operations, and rules for adapting step sizes (σ, ν, and ψ).

2.1 Recombination Operators

FCEA implements two simple recombination operators to generate offspring: modified discrete recombination and blend crossover (BLX-0.5) [9]. The intermediate recombination [1], a special case of BLX-0.5, is also applied. With probabilities 0.5, 0.25, and 0.25 at each stage only one of the three operators is chosen. Probabilities are set according to experimental experience. Here, it is noted again that recombination operators are activated with only a probability p_c. The main variable x and a step size are recombined in a recombination operator.

Modified Discrete Recombination: The original discrete recombination [1] generates a child that inherits genes from two parents with equal probability. Here the two parents of the recombination operator are the "family father" and another solution randomly selected. Experience indicates that FCEA can be more robust if the child inherits genes from the "family father" with a higher probability. Therefore, the operator was modified to be as follows:

$$x_j^c = \begin{cases} x_j^a \text{ with probability } 0.8 \\ x_j^b \text{ with probability } 0.2 \end{cases} \tag{2}$$

Downloaded by [National Chiao Tung University] at 22:16 27 April 2014

BLX-0.5 and Intermediate Recombination: BLX-0.5 [9] is successfully used in real-coded genetic algorithms. It is defined as:

$$x_j^c = x_j^a + \beta(x_j^b - x_j^a) \text{ and} \tag{3}$$

$$w_j^c = w_j^a + \beta(w_j^b - w_j^a) \tag{4}$$

where β is chosen uniformly in the range $[-0.5, 1.5]$; and w is σ, v or ψ based on the mutation operator applied in the family competition. For example, if self-adaptive Gaussian mutation is used in this FC_adaptive procedure, x in Eqs. (3) and (4) is v. When β is fixed to 0.5, BLX-0.5 is termed intermediate recombination. The work of the evolution strategies community [1] was followed to employ only intermediate recombination on step-size vectors, that is, σ, v, and ψ. To be more precise, x is also recombined when the intermediate recombination is chosen.

2.2 Mutation Operators

Mutations are main operators of FCEA. After the recombination, a mutation operator is applied to each "family father" or the new offspring generated by recombination. In FCEA, the mutation is performed independently on each vector element of the "family father" by adding a random value with expectation zero:

$$x_i' = x_i + wD(\cdot) \tag{5}$$

where x_i is the ith component of x, x_i' is the ith component of x' mutated from x, $D(\cdot)$ is a random variable, and w is the step size. In this paper, $D(\cdot)$ is evaluated as $N(0, 1)$ or $C(1)$ if the mutations are, respectively, Gaussian or Cauchy.

Self-Adaptive Gaussian Mutation: We adopted Schwefel's proposal [23] to use self-adaptive Gaussian mutation in global optimization. The mutation is accomplished by first mutating the step size v_j and then the variable x_j:

$$v_j^c = v_j^a \exp[\tau' N(0, 1) + \tau N_j(0, 1)] \tag{6}$$

$$x_j^c = x_j^a + v_j^c N_j(0, 1) \tag{7}$$

where $N(0, 1)$ is the standard normal distribution. $N_j(0, 1)$ is a new value with distribution $N(0, 1)$ that must be regenerated for each index j. For FCEA, Ref. [1] was followed in setting τ and τ' as $(\sqrt{2n})^{-1}$ and $(\sqrt{2\sqrt{n}})^{-1}$, respectively.

Self-Adaptive Cauchy Mutation: A random variable is said to have the Cauchy distribution ($\sim C(t)$) if it has the following density function:

$$f(x; t) = \frac{t/\pi}{t^2 + x^2}, \quad -\infty < x < \infty \tag{8}$$

Self-adaptive Cauchy mutation is defined as follows:

$$\psi_j^c = \psi_j^a \exp[\tau' N(0, 1) + \tau N_j(0, 1)] \tag{9}$$

$$x_j^c = x_j^a + \psi_j^c C_j(t) \tag{10}$$

Downloaded by [National Chiao Tung University] at 22:16 27 April 2014

In the present work, t is 1. Note that self-adaptive Cauchy mutation is similar to self-adaptive Gaussian mutation except that Eq. (7) is replaced by Eq. (10). That is, they implement the same step-size control.

Figure 2 compares density functions of Gaussian distribution ($N(0, 1)$) and Cauchy distributions ($C(1)$). Clearly Cauchy mutation is able to make a larger perturbation than Gaussian mutation. This implies that Cauchy mutation has a higher probability of escaping from local optima than Gaussian mutation. However, the order of local convergence is identical for Gaussian and spherical Cauchy distributions, while nonspherical Cauchy mutations lead to slower local convergence [20].

Decreasing-Based Gaussian Mutations: Decreasing-based Gaussian mutation uses the step-size vector σ with a fixed decrease rate $\gamma = 0.95$ as follows:

$$\sigma^c = \gamma\sigma^a \tag{11}$$

$$x_j^c = x_j^a + \sigma^c N_j(0, 1) \tag{12}$$

Previous results [30] demonstrated that self-adaptive mutations converge faster than decreasing-based mutations but, for rugged functions, self-adaptive mutations more easily become trapped in local optima than decreasing-based mutations.

It can be seen that step sizes are the same for all components of x^a in the decreasing-based mutation, but are different in the self-adaptive mutations. Decreasing-based mutation can be visualised as a search for a better child in a hypersphere centered on the parent. However, for self-adaptive mutation, the search space becomes a hyperellipse. Figure 3 illustrates this difference by two-dimensional contour plots. Therefore, children are searched for in two different types of regions, according to the types of mutations.

2.3 Adaptive Rules

The performance of Gaussian and Cauchy mutations is largely influenced by the step sizes. FCEA adjusts the step sizes while mutations are applied (*e.g.* Eqs. (6), (9), and (11)).

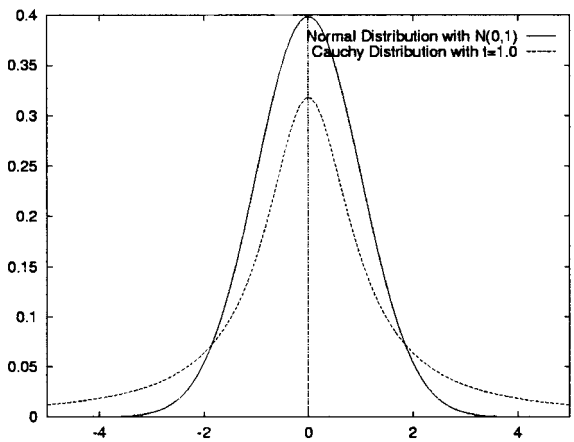


FIGURE 2 Density functions of Gaussian and Cauchy distributions.

Downloaded by [National Chiao Tung University] at 22:16 27 April 2014

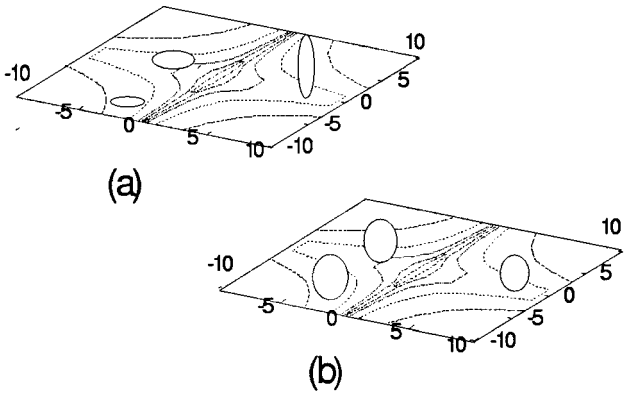


FIGURE 3 The different search spaces for self-adaptive (a) and decreasing-based (b) mutations. Using two-dimensional contour plots, the search spaces from parents are ellipses and circles.

Downloaded by [National Chiao Tung University] at 22:16 27 April 2014

However, such updates insufficiently consider the performance of the whole family. Therefore, after family competition, some additional rules are implemented:

1. *A-decrease-rule*: Immediately after self-adaptive mutations, if objective values of all offspring are greater than or equal to that of the "family parent", the step-size vectors v (Gaussian) or ψ (Cauchy) of the parent are decreased:

$$w_j^a = 0.95 w_j^a \tag{13}$$

   where w^a is the step-size vector of the parent. In other words, when there is no improvement after self-adaptive mutations, a more conservative, that is, smaller, step size tends to make better improvement in the next iteration. This is inspired by the 1/5-success rule of $(1 + \lambda)$-ES [1].
2. *D-increase-rule*: It is difficult to decide the rate γ for decreasing-based mutations. Unlike self-adaptive mutations which adjust step sizes automatically, its step size goes to zero as the number of iterations increases. Therefore, it is essential to employ a rule which can enlarge the step size in some situations. The step size of the decreasing-based mutation should not be too small, when compared to step sizes of self-adaptive mutations. In this work σ is increased if one of the two self-adaptive mutations generates better offspring. To be more precise, after a self-adaptive mutation, if the best child with step size v is better than its "family father", the step size of the decreasing-based mutation is updated as follows:

$$\sigma^c = \max(\sigma^c, \beta v_{\text{mean}}^c) \tag{14}$$

where v_{mean}^c is the mean value of the vector v; and β is chosen to be 0.2. Note that this rule is applied in the stages of self-adaptive mutations but not in those of decreasing-based mutations.

3 ANALYSIS OF FCEA

This section discusses several characteristics of FCEA by numerical experiments and mathematical explanations. The core idea of FCEA is that the mutation operators are able to cooperate with each other in order to achieve good performance by applying the adaptive rules and family competition.

Ten well-known functions, summarized in Table I, were used for the tests. The test functions reflect a different degree of complexity. Functions f_7 to f_9 are unimodal but

Downloaded by [National Chiao Tung University] at 22:16 27 April 2014

TABLE I The Function Test Bed.

Function	*Limit*	*Name*	f_{min}
$f_1(x)^* = -\sum_{i=1}^{m} c_i[e^{-\|x_i - A_i\|^2/\pi}\cos(\pi\|x_i - A_i\|^2)]$, where $m = 15$	$x_i \in [0, 10]$	Modified Langerman	-1.5
$f_2(x) = -\sum_{i=1}^{n}\sin(y_i)\sin^{2m}((iy_i^2)/\pi)$, where $y_i = \begin{cases} x_i\cos\pi/6 - x_{i+1}\sin\pi/6 & \text{if } i \bmod 2 = 1 \\ x_{i-1}\sin\pi/6 + x_i\cos\pi/6 & \text{if } i \bmod 2 = 0 \text{ and } i \neq n \\ x_i & \text{if } i = n \end{cases}$	$x_i \in [0, \pi]$	Epistatic Michalewicz	-9.66
$f_3(x) = -20\exp(-0.2\sqrt{1/n\sum_{i=1}^{n}x_i^2}) - \exp(1/n\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$	$x_i \in [-30, 30]$	Ackley	0
$f_4(x) = \sum_{i=1}^{n}[x_i^2 - 10\cos(2\pi x_i) + 10]$	$x_i \in [-5.12, 5.12]$	Rastrigin	0
$f_5(x) = \sum_{i=1}^{n} x_i\sin(\sqrt{\lvert x_i\rvert})$	$x_i \in [-500, 500]$	Schwefel	$-420.9687n$
$f_6(x) = \sum_{i=1}^{n} x_i^2/4000 - \prod_{i=1}^{n}\cos(x_i/\sqrt{i}) + 1$	$x_i \in [-600, 600]$	Griewank	0
$f_7(x) = \sum_{i=1}^{n-1}(1 - x_i)^2 + 100(x_{i+1} - x_i)^2$	$x_i \in [-5.12, 5.12]$	Rosenbrock	0
$f_8(x) = \sum_{i=1}^{n}(\sum_{j=1}^{i} x_j)^2$	$x_i \in [-65.536, 65.536]$	Ridge	0
$f_9(x) = \sum_{i=1}^{n} 10^{i-1}x_i^2$	$x_i \in [-10, 10]$	Ellipsoid	0
$f_{10}(x) = (\sum_{i=1}^{n} x_i^2)^{0.25}[\sin^2(50(\sum_{i=1}^{n} x_i^2)^{0.1}) + 1.0]$	$x_i \in [-100, 100]$	V sinwave	0

*: A_i are constant vectors. See http://homepages.ulb.ac.be/~gseront/ICEO/Functions/Functions.html

f_1 to f_6 and f_{10} are multimodal functions each of whose number of local minima increases exponentially with the problem dimension [25]. Functions f_1 and f_2 are selected from the second international contest on evolutionary optimization. Function f_7 is considered to be difficult because the minimum is located in a narrow curved valley. In addition, functions f_8 and f_9 are quadratic problems. Figures 4(a) and 4(b) show respectively the two-dimensional plots of f_5 and f_7. It can be seen that they are quite different.

Many well-known test problems are separable functions [26] which can be defined as follows: a function $f: \mathbb{R}^n \to \mathbb{R}$ is separable if and only if

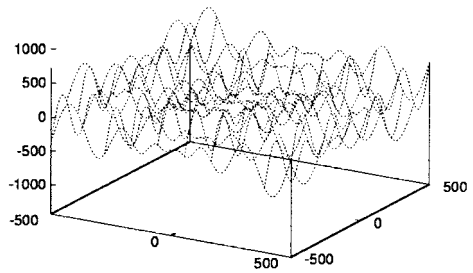
$$f(x) = \sum_{i=1}^{l} g_i(x_i) \tag{15}$$

Thus a separable function $f(x)$ can be decomposed into a sum of l functions g_i. The optimal value of a separable function f can be obtained in a sequence of l independent optimization processes for each parameter x_i.

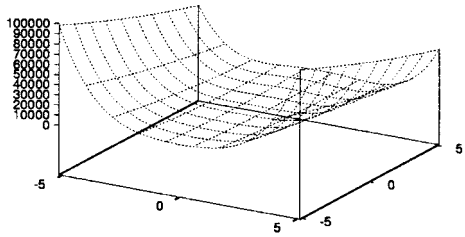
From Table I, only f_4, f_5, and f_9 are separable. Function f_6, a nonseparable function, becomes simpler and smoother as the dimensionality is increased; the problem approaches a separable function because the second term may be neglected.

3.1 Parameters of FCEA

Table II indicates the setting of FCEA parameters, such as family competition lengths and recombination probabilities. They are used for most of functions defined in this work. L_d,



(a) Schwefel's function (f_5)



(b) Rosenbrock's function (f_7)

FIGURE 4 Two-dimensional plots of Schwefel's function (f_5) and Rosenbrock's function (f_7).

Downloaded by [National Chiao Tung University] at 22:16 27 April 2014

TABLE II Parameters of FCEA and Notation Used in This Paper.

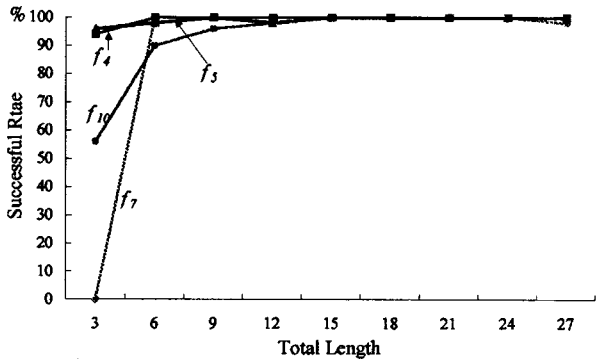
Parameter name	*Value of parameter*
Recombination probability (p_c)	$p_{cD} = 0.8$ (recombination rate in decreasing-based stage) $p_{cA} = 0.2$ (recombination rate in two adaptive stages)
Family competition length	$L_d = 2$ (length in decreasing-based stage) $L_a = 2$ (length in two adaptive stages)
Step sizes	$v_i = \psi_i = \begin{cases} 0.1\lvert b_i - a_i\rvert, & \lvert b_i - a_i\rvert \le 10 \\ 10, & \text{otherwise} \end{cases}$ $\sigma_i = 4v_i$
Decreasing rate	$\gamma = 0.95$
Other notation	n: Number of variables; N: population size; Mean: the average of the best solutions found; L_T: total family competition length ($2L_a + L_d$); SR: percentage of finding a global optimum on 50 independent runs; FE: the average number of function evaluations on 50 independent runs.

Downloaded by [National Chiao Tung University] at 22:16 27 April 2014

σ, and p_{cD} are the parameters for decreasing-based mutation; L_a, v, ψ, and p_{cA} are for self-adaptive mutations. These parameters were decided after experiments on 50 functions from previous studies [3, 7, 9, 10, 12, 18, 24, 32] with various values. For each problem FCEA is tested by 50 independent runs. Of course, not all combinations of various parameter values were tested. FCEA stops if it exceeds a maximum number of function evaluations or $|f(x_{\text{best}}) - f(x^*)| \le \varepsilon$, where x_{best} is the solution found by FCEA and x^* is a global optimum. The maximum number of function evaluations is 1,200,000 for f_7 and f_{10}; 400,000 for all others. In order to compare with other evolutionary algorithms (*e.g.* Refs. [16, 18]), the same value of $\varepsilon = 10^{-3}$ was used. Parameters listed in Table II were chosen based on the following observations:

1. Because the family length is a critical factor in FCEA, Figure 5 tests the performance of different L_T values. Figure 5(a) shows the relation between L_T and the success rate while Figure 5(b) shows the relation between L_T and the number of function evaluations. It can be seen that the number of function evaluations increases with increasing family competition length. FCEA has the worst success rates when both L_a and L_d are set to 1.

   Except for f_{10}, FCEA is unable to obtain benefits when L_a and L_d exceed 3. Therefore, both L_a and L_d were set to 6 for f_{10} which is multimodal and nonseparable function. To reduce the number of function evaluations, L_a was set to 4 and L_d to 2 for unimodal and nonseparable functions, such as f_7 and f_8. L_a and L_d are set to 2 for the others. We recommend that FCEA should use enlarged family competition lengths (L_d and L_a) on nonseparable functions in order to achieve better performance.
2. FCEA was implemented on test functions with recombination probabilities between 0.0 to 1.0. Based on experimental results, p_{cD} and p_{cA} were set to 0.8 and 0.2 respectively. The performance of FCEA is insensitive to these recombination probabilities when $p_c > 0.1$.
3. The step sizes (v and ψ) of self-adaptive mutations are set to $0.1|b_i - a_i|$ and to 10 if $0.1|b_i - a_i| \ge 10$. Decreasing-based mutation with a large initial step size ($\sigma = 4v$) is a global search strategy in FCEA. FCEA is less sensitive than evolution strategies on multimodal problems [5] because FCEA applies both self-adaptive (Eqs. (9) and (6)) and A-decrease-rule (Eq. (13)) mechanisms to adjust v_i and ψ_i according to experimental results.
4. Generally, FCEA should use an increased population size for multimodal functions. The population size should be increased when the problem size (n) becomes larger.



(a) The success rate (SR)

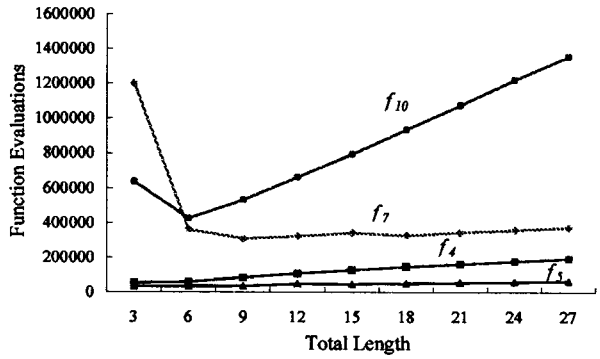


FIGURE 5 The success rates and the average numbers of function evaluations of FCEA on different family competition lengths for five test problems. Each problem is tested over 50 runs for each length.

3.2 The Effectiveness of Multiple Operators and Family Competition

Using multiple mutations in each iteration is one of the main features of FCEA. For the successful working of a global optimization algorithm, it should consist of both global and local search strategies to facilitate both exploitation and exploration. With family selection and block deletion, FCEA uses a high selection pressure along with a diversity-preserving mechanism. With a high selection pressure it become necessary to use highly disruptive search operators such as the series of three mutation operators used in FCEA. Since it always performs a selection after each mutation procedure, the sequence of three mutation operators is similar to a local search. Using numerical experiments, it can be demonstrated that the three operators work well with one another and possess good local and global properties.

Seven different uses of mutation operators are compared in Tables III and IV. Each use combines some of the three operators applied in FCEA: decreasing-based Gaussian mutation (M_{dg}), self-adaptive Cauchy mutation (M_c), and self-adaptive Gaussian mutation (M_g). For example, the M_c approach uses only self-adaptive Cauchy mutation; the $M_{dg} + M_c$ approach integrates decreasing-based Gaussian mutation and self-adaptive Cauchy mutation; and FCEA is an approach integrating M_{dg}, M_c, and M_g. The FCEA$_{ncr}$ approach is a special case of FCEA without adaptive rules, that is, without the A-decrease-rule (Eq. (13)) and

Downloaded by [National Chiao Tung University] at 22:16 27 April 2014

Downloaded by [National Chiao Tung University] at 22:16 27 April 2014

TABLE III Comparison of One-operator Approaches of FCEA Based on 50 Runs.

				M_c			M_{dg}			M_g		
	n	N	L_T	*SR*	*FE*	*Mean*	*SR*	*FE*	*Mean*	*SR*	*FE*	*Mean*
f_1	10	150	6	80%	222,383	−1.313	96%	111,493	−1.403	88%	158,034	−1.347
f_2	10	150	6	54%	337,131	−9.644	100%	186,131	−9.659	92%	230,045	−9.657
f_3	10	10	6	100%	14,409	0.001	100%	19,394	0.001	100%	8893	0.001
f_4	20	40	6	100%	82,203	0.001	100%	60,453	0.001	98%	54,619	0.021
f_5	10	40	6	100%	37,401	−0.001	92%	43,129	9.474	100%	22,757	−0.001
f_6	10	40	6	100%	53,694	0.001	100%	43,939	0.001	100%	34,640	0.001
f_7	10	10	10	98%	468,916	0.081	0%	1,250,040	6.100	100%	287,946	0.001
f_8	10	20	10	100%	163,059	0.001	66%	1,44,534	0.004	100%	92,922	0.001
f_9	10	10	6	100%	19,786	0.001	0%	400,000	32124.2	100%	16,121	0.001
f_{10}	10	100	18	20%	1,137,597	0.006	100%	1,026,140	0.001	50%	923,740	0.003

Downloaded by [National Chiao Tung University] at 22:16 27 April 2014

TABLE IV Comparison of Multi-operator Approaches of FCEA and $FCEA_{ncr}$ (Which Does not Implement Adaptive Rules).

				$M_{dg}+M_c$			$M_{dg}+M_g$			*FCEA*			$FCEA_{ncr}$		
	n	*N*	L_T	*SR*	*FE*	*Mean*	*SR*	*FE*	*Mean*	*SR*	*FE*	*Mean*	*SR*	*FE*	*Mean*
f_1	10	150	6	86%	156,484	−1.339	86%	137,939	−1.342	98%	140,033	−1.452	92%	194,541	−1.374
f_2	10	150	6	94%	180,354	−9.657	100%	152,911	−9.659	100%	213,389	−9.659	76%	313,373	−9.656
f_3	10	10	6	100%	17,870	0.001	98%	13,290	0.024	100%	14,588	0.001	98%	19,892	0.001
f_4	20	40	6	100%	60,886	0.001	98%	58,932	0.021	100%	59,397	0.001	96%	65,302	0.061
f_5	10	40	6	100%	32,493	−0.001	98%	27,136	2.367	100%	27,638	−0.001	100%	35,096	−0.001
f_6	10	40	6	100%	45,391	0.001	100%	41,648	0.001	100%	43,330	0.001	100%	49,863	0.001
f_7	10	10	10	96%	482,837	0.160	98%	301,272	0.001	100%	306,330	0.001	18%	1,121,018	5.589
f_8	10	20	10	100%	142,880	0.001	98%	87,273	0.001	100%	91,358	0.001	0%	400,000	10.400
f_9	10	10	6	100%	29,456	0.001	100%	23,002	0.001	100%	29,793	0.001	0%	400,000	4411.70
f_{10}	10	100	18	100%	956,378	0.001	98%	926,266	0.001	100%	932,015	0.001	98%	1,„042,104	0.001

D-increase-rule (Eq. (14)). Except for $FCEA_{ncr}$, the other uses employ adaptive rules. In order to have a fair comparison, the total length of family competition (L_T) of all seven approaches was set to the same value. For example, if $L_d = L_a = 2$ in FCEA, $L_T = 6$ for one-operator approaches (M_{dg}, M_c, and M_g) and $L_d = L_a = 3$ for two-operator approaches ($M_{dg} + M_c$ and $M_{dg} + M_g$).

There are some observations from experimental results:

1. One-operator approaches (M_{dg}, M_c, and M_g) have widely different performances. Table III shows that self-adaptive mutations (M_c and M_g) outperform the decreasing-based mutation (M_{dg}) on nonseparable unimodal functions (f_7 and f_8). On the other hand, the former performs worse than the latter on nonseparable multimodal functions (f_1, f_2, and f_{10}).
2. Self-adaptive Gaussian mutation converges faster than Cauchy mutation. This is consistent with the theoretical results [20].
3. Generally, strategies with a suitable combination of multiple mutations (FCEA and $M_{dg} + M_g$) perform better than one-operator strategies, in terms of the solution quality. However, the number of function evaluations does not increase much when using multi-operator approaches. Sometimes the number even decreases (*e.g.* FCEA versus M_{dg}). Overall FCEA has the best performance and the number of function evaluations is very competitive. The different approaches have very different performance on f_1, f_2, f_7, and f_{10}. We claim that mutation operators used in FCEA are able to cooperate with each other. The decreasing-based mutation may lead self-adaptive mutations into the global basin.
4. Family competition is a useful strategy. Evolution strategies [33] and evolutionary programming [32] are often unsatisfactory on multimodal functions, but Table III shows that M_g performs very well on such functions (f_4 to f_6). The main reason appears to that M_g applies adaptive rules and family competition.
5. The control of step sizes is important, according to the comparison of $FCEA_{ncr}$ and FCEA in Table IV. Similar observations were made when FCEA was applied in training neural networks [31].
6. The family competition length is crucial for solving complex problems, such as multimodal and nonseparable functions. For example, the length must be increased for solving f_{10} to obtain robust performance.

4 COMPARISON WITH OTHER METHODS

Following the detailed discussion of FCEA in the last section, it is now compared with other methods. The same test problems f_1 to f_{10} are used, but since not all previous studies have released results on all functions, some of the comparisons here are not complete. In addition, the scalability (*i.e.* the same function with different sizes) of FCEA and other approaches was examined.

Before showing results, some interesting observations about existing methods may be made:

1. The genetic algorithms community [3, 6–8, 17, 18] has often studied separable functions, but not large ($n > 10$) nonseparable functions for both unimodal and multimodal functions.
2. In contrast to genetic algorithms, research on evolution strategies and evolutionary programming [10, 22] had seldom studied multimodal functions whose numbers of local optima are large, even when those functions are separable.
3. Traditional global optimization algorithms (*e.g.* [24, 25]) are usually tested by using small problems.

Downloaded by [National Chiao Tung University] at 22:16 27 April 2014

TABLE V The Average Number of Function Evaluations of FCEA on f_3 to f_8 with n from 10 to 400 Where n is the Number of Variables.

n	f_3	f_4	f_5	f_6	f_7	f_8
10	18,215	34,785	29,459	43,445	301,272	19,451
30	66,410	97,887	115,652	77,512	1,775,917	385,025
50	103,726	344,114	219,053	89,293	4,670,092	1,886,624
100	120,369	653,580	424,247	150,325	N/A*	N/A*
200	304,272	1,300,573	872,269	382,344	N/A*	N/A*
400	913,528	2,854,456	2,076,059	750,524	N/A*	N/A*

*Not applicable.

TABLE VI The Performance and Average Number of Function Evaluations of FCEA on f_{10} with n from 2 to 15 by Using Various Family Competition Lengths where n is the Number of Variables.

n	N	L_a	L_d	*Mean*	*FE*
2	20	2	2	0.001	21,871
5	30	2	2	0.001	83,595
8	50	4	4	0.001	301,810
10	80	9	6	0.001	1,027,294
13	250	12	9	0.001	5,240,723
15	400	12	12	0.0049	7,995,965

Downloaded by [National Chiao Tung University] at 22:16 27 April 2014

First the scalability of FCEA is examined. To the best of the authors' knowledge, among existing methods only BGA [17] solved functions f_3 to f_6 with $n > 100$. They claimed that the number of function evaluations scales only as $n\ln(n)$. FCEA was applied to these functions with n from 10 to 400 by enlarging the population size from 40 to 600. Table V shows that for functions f_3 to f_6 the number of function evaluations increases approximately in the order of n or $n\ln(n)$. FCEA also can solve f_7 and f_8 with $n \leq 50$. However, BGA cannot locate global minima for these two problems with $n > 10$.

Table VI presents results of FCEA on f_{10} with $n = 2$, 5, 10, and 15. FCEA can locate a local optimum 0.00536 when $n > 20$. Nevertheless, to the best of our knowledge no other methods have located the global optimum of f_{10} when $n > 12$. Again, note that it is necessary to enlarge family competition lengths for multimodal and nonseparable functions.

Table VII shows the comparison of FCEA with other well-known genetic algorithms, such as CHC [16] and a real-coded genetic algorithm, BGA [18], on functions f_4 to f_7. FCEA offers more stable performance than all other approaches, except BGA [18]. Though CHC used

TABLE VII Comparison of FCEA with CHC and a Real-coded Genetic Algorithm (BGA) on f_4 to f_7.

		FCEA		*CHC [16, 26]*		*BGA [17]*	
	n	*FE*	*SR*	*FE*	*SR*	*FE*	*SR*
f_4	20	59,115	100%	158,839	100%	9900	100%
f_5	10	29,459	100%	9803	100%	8699	100%
f_6	10	43,445	100%	51,015	100%	59,520	92%
f_7	2	3709	100%	9455	100%	1671	100%
f_7	10	301,272	100%	N/A*	3%	N/A*	N/A*

*Not available in the original papers.

Downloaded by [National Chiao Tung University] at 22:16 27 April 2014

TABLE VIII Comparison of FCEA with Evolution Strategies (ES_{Gau} and ES_{Cau}), Evolutionary Programming (EP_{Gau} and EP_{Cau}), and a Multi-operator Evolutionary Programming Method ($EP_{Cau+Gau}$).

		FCEA							
	n	*FE*	*Mean*	*FE*	ES_{Gau} [33] *Mean*	ES_{Cau} [33] *Mean*	EP_{Gau} [32] *Mean*	EP_{Cau} [32] *Mean*	$EP_{Cau+Gau}$ [22] *Mean*
f_3	30	66,410	0.001	150,000	9.07	0.012	9.2	0.018	0.0004
f_4	30	97,887	0.001	500,000	70.82	0.16	89.0	0.046	N/A*
f_5	30	115,652	−12629.01	900,000	−7549.9	−12556.4	−7917.1	−12554.5	N/A*
f_6	30	77,512	0.001	200,000	0.38	0.037	0.086	0.016	N/A*
f_7	30	1,775,917	0.001	2,000,000	6.69	33.28	6.17	5.06	1.58

*Not available in the original papers.

Downloaded by [National Chiao Tung University] at 22:16 27 April 2014

TABLE IX Test Bed of the 1st International Contest on Evolutionary Optimization.

Function	*Limit*	*Name*	f_{min}
$f_{B1}(x) = \sum_{i=1}^{n}(x_i - 1)^2$	$x_i \in [-5, 5]$	Sphere	0
$f_{B2}(x) = 1/4000 \sum_{i=1}^{n}(x_i - 100)^2 - \prod_{i=1}^{10} \cos((x_i - 100)/\sqrt{i}) + 1$	$x_i \in [-600, 600]$	Griewank	0
$f_{B3}(x) = -\sum_{i=1}^{m} 1/(\|x_i - A_i\|^2 + c_i)$, where $m = 30$	$x_i \in [0, 10]$	Shekel's foxholes	-10.2078
$f_{B4}(x) = -\sum_{i=1}^{n} \sin(x) \sin^{2m}((ix_i^2)/\pi)$, where $m = 10$	$x_i \in [0, \pi]$	Michalewicz	-9.66
$f_{B5}(x) = -\sum_{i=1}^{m} e^{-1/\pi\|x_i - A_i\|^2} \cos(\pi \cdot \|x_i - A_i\|^2)$, where $m = 5$	$x_i \in [0, 10]$	Langerman	-1.5

A_i are constant vectors. See http://homepages.ulb.ac.be/~gseront/ICEO/Functions/Functions.html

Gray coding rather than binary coding to achieve better performance, it cannot stably solve the Rosenbrock problem (f_7) with $n > 10$ [26]. In addition, though BGA performs well in general, it only solves 92% of trials on f_6. Salomon's work (21) is used to compare FCEA with BGA on coordinate rotation of functions f_3 to f_5, f_8, and f_9. Rotation means that a linear and orthogonal transformation matrix $\mathcal{T}$ is applied so the new problem is to minimize $f(\mathcal{T}, x)$. The same method used in Ref. [21] is used here to generate $\mathcal{T}$. FCEA performs better than BGA on these rotation functions and nonseparable function f_8. According to Ref. [21], Gaussian mutations with one global step size are rotationally invariant. Since the decreasing-based mutation uses one global step size, FCEA still performs well on rotated functions. Overall, FCEA seems more stable than these modified genetic algorithms for a test bed containing various types of functions.

Table VIII shows the comparison of FCEA with evolution strategies [33] and evolutionary programming [32] approaches which use Gaussian or Cauchy mutations on functions f_3 to f_7 with $n = 30$. Also include are results of a two-operator approach [22] which in general outperforms one-operator approaches. For other approaches than FCEA, solutions after exceeding a fixed number of function evaluations are reported. FCEA again offers a stable performance. From Tables VII and VIII, it seems that evolution strategies and evolutionary programming are better than modified genetic algorithms on the Rosenbrock problem (f_7) but worse on the Schwefel problem (f_5). Some previous results [30] also demonstrated the same observation.

Downloaded by [National Chiao Tung University] at 22:16 27 April 2014

5 TESTING FUNCTIONS FROM THE INTERNATIONAL CONTESTS ON EVOLUTIONARY OPTIMIZATION

The aim of the first [2] and second international contests on evolutionary optimization is to allow researchers to compare their algorithms on a common test bed. Two indices are defined to measure the performance of the proposed algorithm: the expected number of evaluations per success, FE, and the best value reached, BV. Here BV is the best value reached during the 20 runs by an algorithm.

Functions from these two international contests are shown in Table IX [2] and Table XI. Results of comparing FCEA with the three evolutionary algorithms [2] on problems from the first contest are given in Table X. Sto-Pri, Van-Ke, and Se-Be represent the authors of these algorithms which are the best among comparative approaches, while non-evolutionary approaches are not considered. Table XII presents results of the second contest. It can be observed that FCEA is very competitive with state-of-the-art evolutionary algorithms.

TABLE X Average Results of 50 Runs of FCEA on Problems of the 1st International Contest on Evolutionary Optimization, and Comparison with Three Hybrid Evolutionary Algorithms. Sto-Pri, Van-Ke, and Se-Be Represent the Authors of These Algorithms.

		FCEA				*Sto-Pri [2]*	*Van-Ke [2]*	*Se-Be [2]*
Function	*n*	*N*	*FE*	*BV**	*SR*	*FE*	*FE*	*FE*
f_{B1}	10	2	1658	0.0004	100%	1892	3462	1099
f_{B2}	10	40	43,905	0.0004	100%	13,508	19,125	6446
f_{B3}	10	150	99,169	−10.17	96%	744,250	363,685	259,477
f_{B4}	10	50	42,201	−9.6597	100%	10,083	41,765	236,348
f_{B5}	10	60	30,982	−1.498	100%	44,733	61,729	1,032,627

*The best value reached in 50 runs.

Downloaded by [National Chiao Tung University] at 22:16 27 April 2014

TABLE XI Test Bed of 2nd International Contest on Evolutionary Optimization.

Function	*Limit*	*Name*	f_{min}
$f_{C1}(x) = \sum_{i=1}^{n-1}(1 - x_i)^2 + 100(x_{i+1} - x_i)^2$	$x_i \in [-5.12, 5.12]$	Rosenbrock	0
$f_{C2}(x) = e^{-(\|x_i - A_i\|_\infty^2)/2\pi} \cos(\pi\|x_i - A_i\|_\infty^2)(1 + c_1(\|x_i - A_i\|^2)/(\|x_i - A_i\|_\infty^2 + 0.01))$	$x_i \in [-5\pi, 5\pi]$	Odd square	-1.023
$f_{C3}(x) = -\sum_{i=1}^{m} c_i[e^{-\|x_i - A_i\|^2/\pi} \cos(\pi\|x_i - A_i\|^2)]$, where $m = 15$	$x_i \in [0, 10]$	Modified Langerman	-1.5
$f_{C4}(x) = -\sum_{i=1}^{m} 1/(\|x_i - A_i\|^2)$, where $m = 30$	$x_i \in [0, 10]$	Shekel's foxholes	-10.2078
$f_{C5}(x) = -\sum_{i=1}^{n} \sin(y_i) \sin^{2m}\left(\frac{iy_i^2}{\pi}\right)$, where $y_i = \begin{cases} x_i \cos\pi/6 - x_{i+1} \sin\pi/6 & \text{if } i \bmod 2 = 1 \\ x_{i-1} \sin\pi/6 + x_i \cos\pi/6 & \text{if } i \bmod 2 = 0 \text{ and } i \neq n \\ x_i & \text{if } i = n \end{cases}$	$x_i \in [0, \pi]$	Epistatic Michalewicz	-9.66
$f_{C6}(x) = (\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_z))/\sqrt{\sum_{i=1}^{n} ix_i^2}$, subject to $\prod_{i=1}^{n} x_i \geq 0.75$, $\sum_{i=1}^{n} x_i \geq 0.75n$	$x_i \in [0, 10]$	Bump	0

A_i are constant vectors. See http://homepages.ulb.ac.be/~gseront/ICEO/Functions/Functions.html

TABLE XII Results Averaged Over 50 Runs of FCEA on the Real Function Test Bed of the 2nd International Contest on Evolutionary Optimization.

Function	*n*	*N*	*FE*	*BV**	*SR*
f_{C1}	10	10	301,272	0.0010	100%
f_{C2}	10	150	353,646	−1.0224	100%
f_{C3}	10	150	140,033	−1.452	100%
f_{C4}	10	150	99,169	−10.17	96%
f_{C5}	10	150	213,389	−9.659	100%
f_{C6}	10	50	6760	0.0003	100%

*The best value reached in 50 runs.

Downloaded by [National Chiao Tung University] at 22:16 27 April 2014

Function f_{B2} is essentially identical to f_6, except its solution is shifted. Comparing Tables X and IV, it is seen that FCEA is little affected by such shifts. Function f_{B3} is a 10-dimensional Shekel's foxholes problem, while in DeJong's test bed [3], only a smaller problem ($n = 2$) is included. Table X shows that f_{B3} is difficult for most algorithms. FCEA must change the recombination rate to be less than 0.05 in order to solve this problem.

It is noted that functions f_{C3} and f_{C5} increase the epistasis between variables of f_{B5} and f_{B4}, respectively. FCEA needs more function evaluations to reach optimal solutions for f_{C3} and f_{C5}.

6 CONCLUSIONS AND FUTURE RESEARCH

The "No Free Lunch Theorem" [27] shows that there is no efficient algorithm for global optimization, in general. However, instead of seeking for a fast algorithm, this research concentrate more on stability. This study has demonstrated that FCEA is a robust approach for global optimization. Experience suggests that a global optimization method should consist of both global and local search strategies. For FCEA, decreasing-based mutations with a large initial step size is a global search strategy; self-adaptive mutations with family competition procedure and replacement selection are local search strategies. These mutation operators can closely cooperate with one another.

Experiments on several well-known functions verify that the proposed approach is very competitive with other algorithms, including genetic algorithms, evolution strategies, and evolutionary programming. The authors believe that the flexibility and robustness of FCEA make it a highly effective global optimization tool.

References

[1] Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice*, Oxford University Press.
[2] Bersini, H., Dorigo, M., Langerman, S., Seront, G. and Gambardella, L. (1996). Results of the first international content on evolutionary optimization (1st ICEO). In: *Proc. of the IEEE Int. Conf. on Evolutionary Computation*, pp. 611–614.
[3] De Jong, K. A. (1975). An analysis of the behavior of a class of genetic adaptive systems. *PhD thesis*, University of Michigan.
[4] Deb, K. and Agrawal, R. B. (1995). Simulated binary crossover for continuous search space. *Complex Systems*, **9**, 115–148.
[5] Deb, K. and Beyer, H.-G. (1999). Self-adaptation in real-parameter genetic algorithms with simulated binary crossover. In: *Genetic and Evolutionary Computation Conf. (GECCO-99)*, pp. 172–179.
[6] Deb, K. and Goldberg, D. E. (1989). An investigation of niche and species formation in genetic function optimization. In: *Proc. of the Third International Conf. on Genetic Algorithms*, pp. 42–50.
[7] Eshelman, L. J. (1991). The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In: Rawlins, G. J. (Ed.), *Foundation of Genetic Algorithms 1*. Morgan Kaufmann Publishers, Inc., San Mateo, CA, USA, pp. 265–283.

[8] Eshelman, L. J. and Schaffer, J. D. (1993). Crossover's niche. In: Forrest, S. (Ed.), *Proc. of the Fifth International Conf. on Genetic Algorithms.* Morgan Kaufmann Publishers, Inc., pp. 9–14.
[9] Eshelman, L. J. and Schaffer, J. D. (1993). Real-coded genetic algorithms and interval-schemata. In: Whitley, L. D. (Ed.), *Foundation of Genetic Algorithms 2.* Morgan Kaufmann Publishers, Inc., pp. 187–202.
[10] Fogel, D. B. and Atmar, W. (1990). Comparing genetic operators with Gaussian mutations in simulated evolutionary processes using linear systems. *Biological Cybernetics*, **63**, 111–114.
[11] Fogel, L. J., Owens, A. J. and Walsh, M. J. (1966). *Artificial Intelligence Through Simulated Evolution.* Wiley, New York.
[12] Gordon, V. S. and Whitley, D. (1993). Serial and parallel genetic algorithms as function optimizers, 177–183.
[13] Hart, W. E. (1994). Adaptive global optimization with local search. *PhD thesis*, University of California, San Diego.
[14] Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, Ann Arbor, MI.
[15] Janikow, C. Z. and Michalewicz, Z. (1991). An experimental comparison of binary and floating point representations in genetic algorithms. In: *Proc. of the Fourth International Conf. on Genetic Algorithms*, pp. 31–36.
[16] Mathias, K. E. and Whitley, L. D. (1994). Changing representations during search: A comparative study of delta coding. *Evolutionary Computation*, **2**(3), 249–278.
[17] Mühlenbein, H. and Schlierkamp-Voosen, D. (1993). Predictive models for the breeder genetic algorithm I. Continuous parameters optimization. *Evolutionary Computation*, **1**(1), 24–49.
[18] Mühlenbein, H., Schomisch, M. and Born, J. (1991). The parallel genetic algorithm as function optimizer. *Parallel Computing*, **17**, 619–632.
[19] Radcliffe, N. J. (1991). Equivalence class analysis of genetic algorithms. *Complex Systems*, **5**(2), 183–206.
[20] Rudolph, G. (1997). Local convergence rates of simple evolutionary algorithms with Cauchy mutations. *IEEE Trans. on Evolutionary Computation*, **1**(4), 249–258.
[21] Salomon, R. (1996). Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions; a survey of some theoretical and practical aspects of genetic algorithms. *BioSystems*, **39**(3), 263–278.
[22] Saravanan, N. and Fogel, D. B. (1997). Multi-operator evolutionary programming: a preliminary on function optimization. In: Angeline, P. J., *et al.* (Eds.), *Proc. of the 6th Annu. Conf. on Evolutionary Programming (Lecture Notes in Computer Science, 1213)*, pp. 215–222.
[23] Schwefel, H.-P. (1981). *Numerical Optimization of Computer Models.* Wiley, Chichester.
[24] Snyman, J. A. and Fatti, L. P. (1987). A multi-start global minimization algorithm with dynamic search trajectories. *Journal of Optimization Theory and Applications*, **54**(1), 121–142.
[25] Törn, A. and Žilinskas, A. (1989). *Global Optimization, Lecture Notes in Computer Science*, Vol. 350, Springer-Verlag.
[26] Whitley, D., Mathias, K., Rana, S. and Dzubera, J. (1995). Building better test functions. In: *Proc. of the Sixth Int. Conf. on Genetic Algorithms*, pp. 239–246.
[27] Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Trans. on Evolutionary Computation*, **1**, 67–82.
[28] Xiao, J., Michalewicz, Z., Zhang, L. and Trojanowski, K. (1997). Adaptive evolutionary planner/navigator for mobile robots. *IEEE Trans. on Evolutionary Computation*, **1**(1), 18–28.
[29] Yang, J.-M., Chen, Y.-P., Horng, J.-T. Kao, C.-Y. (1997). Applying family competition to evolution strategies for constrained optimization. In: Angeline, P. J., Reynolds, R. G., McDonnell, J. R. and Eberhart, R. (Eds.), *Lecture Notes in Computer Science*, Vol. 1213, 201–211.
[30] Yang, J.-M., Kao, C.-Y. and Horng, J. T. (1997). A continuous genetic algorithm for global optimization. In: *Proc. of the Seventh Int. Conf. on Genetic Algorithms*, pp. 230–237.
[31] Yang, J.-M., Kao, C.-Y. and Horng, J.-T. (1998). A new evolutionary approach to developing neural autonomous agents. In: *IEEE Int. Conf. on Robotics and Automation*, pp. 1411–1416.
[32] Yao, X. and Liu, Y. (1996). Fast evolutionary programming. In: *Proc. of the Fifth Annual Conf. on Evolutionary Programming*, pp. 451–460.
[33] Yao, X. and Liu, Y. (1997). Fast evolution strategies. In: Angeline, P. J., Reynolds, R. G., McDonnell, J. R. and Eberhart, R. (Eds.), *Proc. of the 6th Annu. Conf. on Evolutionary Programming (Lecture Notes in Computer Science, 1213)*, pp. 151–161.
[34] Yen, J., Liao, J. C., Lee, B. and Randoph, D. (1998). A hybrid approach to modeling metabolic systems using a genetic algorithm and simplex method. *IEEE Trans. on Systems, Man, and Cybernetics – Part B*, **28**(2), 173–191.
[35] Yip, P. P. C. and Pao, Y. H. (1995). Combinatorial optimization with use of guided evolutionary simulated annealing. *IEEE Trans. on Neural Networks*, **6**(2), 290–195.

Downloaded by [National Chiao Tung University] at 22:16 27 April 2014