Applied Intelligence 15, 171–180, 2001
© 2001 Kluwer Academic Publishers. Manufactured in The Netherlands.

Adapting Self-Adaptive Parameters in Evolutionary Algorithms

KO-HSIN LIANG
School of Computer Science, University College, The University of New South Wales, Australian Defence Force Academy, Canberra, ACT, Australia 2600
liangk@cs.adfa.edu.au

XIN YAO
School of Computer Science, The University of Birmingham Edgbaston, Birmingham B15 2TT, UK
x.yao@cs.bham.ac.uk

CHARLES S. NEWTON
School of Computer Science, University College, The University of New South Wales, Australian Defence Force Academy, Canberra, ACT, Australia 2600
csn@cs.adfa.edu.au

Abstract. The lognormal self-adaptation has been used extensively in evolutionary programming (EP) and evolution strategies (ES) to adjust the search step size for each objective variable. However, it was discovered in our previous study (K.-H. Liang, X. Yao, Y. Liu, C. Newton, and D. Hoffman, in *Evolutionary Programming VII. Proc. of the Seventh Annual Conference on Evolutionary Programming*, vol. 1447, edited by V. Porto, N. Saravanan, D. Waagen, and A. Eiben, Lecture Notes in Computer Science, Springer: Berlin, pp. 291–300, 1998) that such self-adaptation may rapidly lead to a search step size that is far too small to explore the search space any further, and thus stagnates search. This is called *the loss of step size control*. It is necessary to use a lower bound of search step size to avoid this problem. Unfortunately, the optimal setting of lower bound is highly problem dependent. This paper first analyzes both theoretically and empirically how the step size control was lost. Then two schemes of dynamic lower bound are proposed. The schemes enable the EP algorithm to adjust the lower bound dynamically during evolution. Experimental results are presented to demonstrate the effectiveness and efficiency of the dynamic lower bound for a set of benchmark functions.

Keywords: self-adaptation, evolutionary programming, evolution strategies, global function optimization, dynamic lower bound

1. Introduction

Evolutionary algorithms (EAs) have been applied to many optimization problems successfully in recent years. They are population-based search algorithms with the generation-and-test feature [1]. New offspring are generated by perturbations and tested to determine the acceptable individuals for the next generation.

One of the major applications of EA is global optimization on numerical problems [2–6]. A global optimization problem can be formalized as a pair (S, f), where $S \subseteq R^n$ is a bounded set in R^n and $f : S \to R$ is an n-dimensional real-valued function. The problem is to find a vector $x_{\min} \in S$ such that $f(x_{\min})$ is a global minimum on S. More specifically, it is required to find an $x_{\min} \in S$ such that

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

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

A common feature of EA, especially evolutionary programming (EP) and evolution strategies (ES), in numerical function optimization is self-adaptive step size. During mutation, each object variable $x(j)$, $j = 1, \dots, n$ is added by a normally distributed random number with mean 0 and standard deviation $\eta(j)$, which is referred to as the search step size. These $\eta(j)$'s are not predefined and fixed. They are self-adaptive and evolve along with $x(j)$'s. It has been shown by many researchers that self-adaptation helps evolutionary search. However, self-adaptation is not perfect. It has also been shown that self-adaptation may lead quickly to a very small search step size in EP and prevent search from making progress [7]. A lower bound on the search step size is often needed in order to avoid this problem. This paper studies the issue of dynamic lower bound in EP, although the techniques proposed can be applied equally to any other self-adaptive EAs.

According to the description of Fogel [8] and Bäck and Schwefel [2], EP is implemented in our study as follows:

1. Generate the initial population of μ individuals at random, and set the generation counter $\kappa \leftarrow 1$. Each individual is taken as a pair of real-valued vectors, $(x_i, \eta_i), \forall i \in \{1, \dots, \mu\}$, where η_i is the search step size. Each x_i has n components $x_i(j)$, $j = 1, \dots, n$.
2. Evaluate the fitness of each individual $(x_i, \eta_i), \forall i \in \{1, \dots, \mu\}$, in the population based on the objective function, $f(x_i)$.
3. For each parent (x_i, η_i), $i = 1, \dots, \mu$, create a single offspring (x_i', η_i') as follows:

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

   where $x_i(j)$, $x_i'(j)$, $\eta_i(j)$ and $\eta_i'(j)$ denote the j-th component of the vectors x_i, x_i', η_i and η_i', respectively. $N(0, 1)$ denotes a normally distributed one-dimensional random number with mean 0 and standard deviation 1. $N_j(0, 1)$ indicates that the random number is generated anew for each value of j. The parameters τ and τ' are commonly set to $(\sqrt{2\sqrt{n}})^{-1}$ and $(\sqrt{2n})^{-1}$, respectively [4].
4. Calculate the fitness of each offspring $(x_i', \eta_i'), \forall i \in \{1, \dots, \mu\}$.
5. Conduct pairwise comparisons over the union of parents (x_i, η_i) and offspring (x_i', η_i'), $\forall i \in \{1, \dots, \mu\}$. For each individual, q opponents are chosen randomly from all the parents and offspring with an equal probability. For each comparison, if the individual's fitness is no smaller than the opponent's, it receives a "win."
6. Select the μ individuals out of (x_i, η_i) and (x_i', η_i'), $\forall i \in \{1, \dots, \mu\}$, that have the most wins to be parents of next generation.
7. Stop if the halting criterion is satisfied; otherwise, $\kappa \leftarrow \kappa + 1$ and go to Step 3.

Equation (1) above describes the implementation of lognormal self-adaptation of search step size $\eta_i(j)$. The evolution of $\eta_i(j)$ enables EP to adaptively adjust the search step size for each objective variable. The ideal situation is to have a large step size for the objective variable at the beginning of the evolutionary process in order to explore different regions of the search space, and have a smaller step size at the later stage for better exploitation within a good region. However, self-adaptation may not work sometimes. The search step size $\eta_i(j)$ may reduce to a very small value quickly and thus prevents EP from searching for better solutions [7].

For example, if the distance from $x(j)$ to the minimum $x(j)^*$ for the j-th component is $|x(j) - x(j)^*| \geq 1$ and the adaptive parameter $\eta(j) < 10^{-6}$, the probability for $x(j)$ to mutate into a small neighborhood of $x(j)^*$ will be extremely small. If such an individual x survives in the population, it will propagate the poor η values and stagnate the whole search process. Section 2 of this paper shows an example how this happens and why it is harmful to search.

In ES [3, 9], a fixed lower bound η_- is used to prevent the step size control from being lost. A lower bound was also used in EP [10], but for a very different purpose. It was used to replace negative values of η [10]. Applying a recombination operator to η may reduce the chance of losing the step size control since the probability of recombining two individuals with very small η values is small [2, 3]. Some previous empirical studies [7, 11] have shown that a properly set lower bound on η can improve EP's performance significantly.

Setting a near optimal lower bound for η is a difficult task since it is problem dependent. Different problems require different settings. It is difficult for human beings to guess what a suitable lower bound should be for a given new problem. Furthermore, an optimal lower bound at the beginning of search may not be the same as that in the end because different stages of search requires different search strategies (and thus different lower bounds). In this paper, we propose two schemes to adjust lower bounds dynamically. Only information contained in the population is used in the dynamic schemes. The first scheme is based on the success rate and the second is based on mutation's step size. The use of such dynamic schemes can improve the performance of EP significantly.

The rest of this paper is organized as follows. Section 2 explains how the loss of step size control happens. Section 3 introduces two dynamic schemes for setting lower bounds automatically based on the population information. Section 4 presents the experimental results of EP with different schemes for setting lower bounds. Finally, Section 5 concludes the paper with some remarks.

2. Analysis of Self-Adaptation of Search Step Size

Self-adaptation of search step size in EP does not work as well as one hopes without a proper lower bound. The loss of step size control happens when an individual with very small η but a good fitness value survives and reproduces in the population. The poor η could be regarded as a parasite in this case. It survives not because it is good, but because the individual is good. To observe how this happens in the evolutionary process, a set of experiments of EP without any lower bound on η were carried out on a set of five benchmark functions [7]. Figure 1 shows the average result over 50 independent runs for the 30-dimensional sphere model, i.e.,

$$f(x) = \sum_{i=1}^{30} x_i^2,$$

which is the simplest function among the five.

It is interesting to note that EP without any lower bound on η could only find a value which is well above 100, a very poor result indeed considering that the

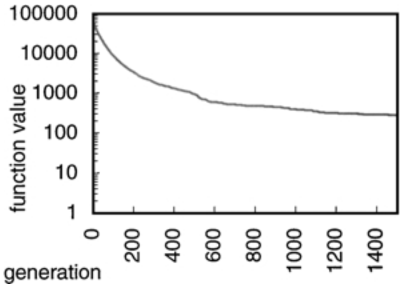

Figure 1. The sphere model stagnates early from mean of 50 runs.

Table 1. The 19th component and the fitness of the best individual in a typical run.

Generation	$(x_1(19), \eta_1(19))$	$f(x_1)$	$\frac{1}{\mu}\sum f(x_i)$
	:		
300	(−14.50, 4.52E–3)	812.85	846.52
	:		
600	(−14.50, 8.22E–6)	547.05	552.84
	:		
1000	(−14.50, 1.33E–8)	504.58	504.59
	:		
1500	(−14.50, 1.86E–12)	244.93	244.93

exact global optimum is 0. It is even more interesting to discover that the worst η values for all individuals in the population were around 10^{-12} while the related object variables were still large. Hence little process could be made after 1500 generations.

In order to understand how and why η values were reduced to such a small value while the objective values were still large, we followed the evolutionary process from generation to generation. It was discovered that mutation of an individual might generate a small η in one dimension, but the fitness of the individual could still be very high due to good mutations along other dimensions. Such a good individual would survive and reproduce quickly in the population and carry the small η with it. Some generations later, the whole population would be filled with individuals with such a small η and would not be able to make much progress. Table 1 shows how the step size control was lost using a randomly selected example from our experiment. In generation 600, the 19th component of the best individual in the population was -14.5. Its η value was only $8.22E$-6. However, the individual survived and reproduced quickly because it was the best one in the population. After another 400 generations (i.e., in generation 1000), the average fitness of the population had become very close to the fitness of the best individual. In other words, all individuals have a similarly small η in the 19th component. The evolution almost entirely stagnated. Similar phenomena could be observed on other benchmark functions we tested.

Figure 2 compares the average result over 50 independent runs of EP with and without a lower bound on η. It is clear that using a lower bound has improved EP's performance significantly. A lower bound enables EP to reduce the function value steadily. Similar significant improvement has been shown on other benchmark functions [7].

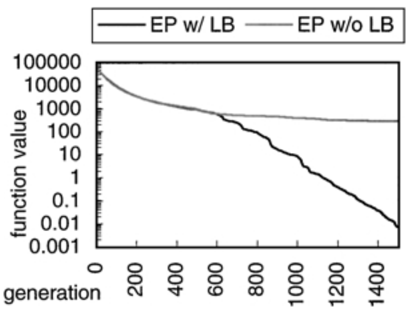

Figure 2. Comparison between EP with and without a lower bound on the sphere model.

In order to get some ideas on how likely the step size control will be lost in evolutionary search, it is worth analyzing the impact of the number of generations on the likelihood of losing the step size control. While it is obvious that the larger the number of generations, the more likely the step size control is lost, it is unclear how fast the increase in likelihood will be. In other words, we are most interested in the rate of such increase. To simplify our analysis, a (1 + 1) EP will be used in the following discussion.

Given an n-dimensional real-valued function $f(x)$, using one parent in each generation, the adaptive parameter $\eta'(j)$ is created by:

$$\eta^{(\kappa+1)}(j) = \eta^{(\kappa)}(j)\exp(\tau N(0,1))$$

where j denotes the j-th component and $\tau = \frac{1}{\sqrt{n}}$ [12]. This is a modified version of Eq. (1). Given initial $\eta^{(0)}(j)$, we can find $\eta^{(\kappa)}(j)$ after running κ generations of *successful* mutations. Note that the actual generation number will be greater or equal to κ as the success rate of generating a better offspring is no more than 1. Therefore, through the sequence

$$\{\eta^{(1)}(j),\ \eta^{(2)}(j),\ \eta^{(3)}(j), \ldots,\ \eta^{(\kappa)}(j)\},$$

we get

$$\eta^{(\kappa)}(j) = \eta^{(0)}(j)\exp\left(\tau\sum_{i=1}^{\kappa} N_i(0,1)\right).$$

The probability that $\eta^{(\kappa)}(j)$ will be smaller than an arbitrary small number ϵ ($\epsilon > 0$) is:

$$\begin{aligned} P_\eta &= P\big(\eta^{(\kappa)}(j) < \epsilon\big) \\ &= P\left(\eta^{(0)}(j)\ \exp\left(\tau\sum_{i=1}^{\kappa} N_i(0,1)\right) < \epsilon\right) \end{aligned}$$

Since the sum of κ independent $N(0, 1)$ random variables has the distribution [13, p. 267]:

$$\sum_{i=1}^{\kappa} N_i(0,1) \sim N(0,\kappa),$$

we get

$$\begin{aligned} P_\eta &= P\big(\eta^{(0)}(j)\exp(\tau N(0,\kappa)) < \epsilon\big) \\ &= P\left(N(0,\kappa) < \ln\left(\frac{\epsilon}{\eta^{(0)}(j)}\right)\Big/\tau\right) \\ &= \int_{-\infty}^{C} \frac{1}{\sqrt{2\pi\kappa}}\exp\left(-\frac{t^2}{2\kappa}\right)dt \\ &= \Phi\left(\frac{C}{\sqrt{\kappa}}\right) \end{aligned}$$

where $C = \ln(\frac{\epsilon}{\eta^{(0)}(j)})/\tau$. For sufficiently large $\frac{C}{\sqrt{\kappa}}$, the following approximation [14, p. 175] can be used:

$$\Phi(x) \simeq 1 - \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}x^2\right)\cdot\frac{1}{x}$$

The derivative $\frac{\partial}{\partial\kappa}(P_\eta)$ can be used to evaluate the impact of κ on P_η.

$$\begin{aligned} \frac{\partial}{\partial\kappa}(P_\eta) &= \frac{\partial}{\partial\kappa}\left(1 - \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{C^2}{2\kappa}\right)\cdot\frac{\sqrt{\kappa}}{C}\right) \\ &= \frac{-1}{\sqrt{2\pi}\cdot C}\exp\left(-\frac{C^2}{2\kappa}\right)\left(\frac{1}{2\sqrt{\kappa}} + \frac{C^2}{2}\kappa^{-\frac{3}{2}}\right) \end{aligned}$$

For $C = \sqrt{n}\ \ln(\epsilon/\eta^{(0)}(j))$, it is apparent from the above equation that

$$\frac{\partial}{\partial\kappa}(P_\eta)\begin{cases} >0, & \text{if } \epsilon < \eta^{(0)}(j) \\ <0, & \text{if } \epsilon > \eta^{(0)}(j) \end{cases}.$$

According to the above inequalities, the initial search step size can influence the likelihood of losing the step size control. If the initial step size is larger than ϵ, which is usually the case in practice, the likelihood of losing the step size control will accelerate quickly with large κ. In other words, the probability that the adaptive parameter $\eta^{(k)}(j)$ becomes smaller than an arbitrary small number ϵ will be higher when κ is large. This can also be shown empirically.

We conducted an experiment with $(1+1)$ EP using the three-dimensional sphere function $f(x) = \sum_{i=1}^{3} x_i^2$. Starting point of the experiment was (10,10,10). The total number of trials was 100, the

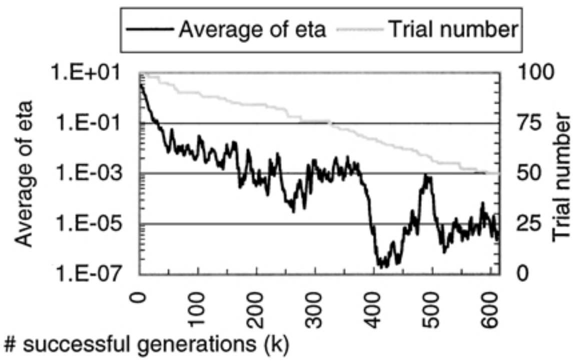

Figure 3. The average variations of $\eta(3)$ are shown. The start of $\eta_j^{(0)} > \eta_j^{(\kappa)}$ is found at $\kappa = 4$.

maximum generation was 3000, the initial adaptive parameter $\eta^{(0)}(j)$ was 3. We randomly selected one vector of η to observe the variation. Only η's with successful mutations were recorded. Figure 3 shows the average variations of the adaptive parameter $\eta(3)$. All $\eta(3)$'s on every successful generation were averaged over the trial number. Only trial numbers over 50 were drawn. It is clear from Fig. 3 that less trials can generate successful mutations for large κ.

When the adaptive parameters decrease, the best situation is when the objective variables are already very close to the global optimum. Thus, smaller step sizes are preferred. If any of the step sizes decreases faster than the rate that the objective variables approach to the optimum, the search process may stagnate. That is, the step size becomes too small to change objective variables sufficiently differently. In Fig. 4, the ratio $x(3)/\eta(3)$ for each successful generation is shown.

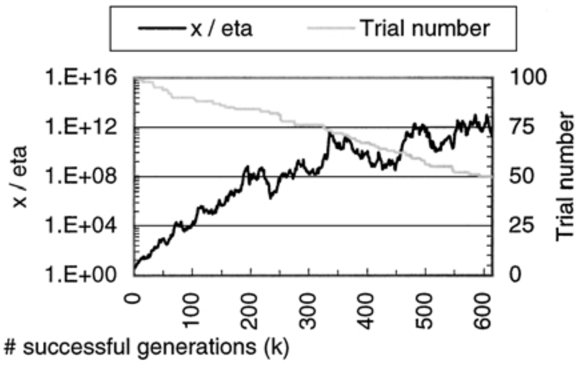

Figure 4. The average relation pairs x_3/η_3 are shown. For, example, after $\kappa = 158$ on average of 85 trials, the worst pair begins to be greater than 10^6 where the stagnation is about to happen.

The average per trial number and the experimental data were obtained from the same experiment as above. The stagnation largely begins when $x(3)/\eta(3)$ is over 10^6.

The previous analysis and experimental results demonstrate that the lognormal self-adaptation in EP does not work very well without a lower bound. Fixed lower bounds can improve EP's performance significantly [7], but they do not take into account that different stages in evolutionary search require different lower bounds and different functions require different lower bounds. In the next section, two different dynamic lower bound schemes will be introduced.

3. Dynamic Lower Bounds (DLBs)

The key issue in developing a dynamic lower bound scheme is how to adjust a lower bound based on the information accumulated so far in evolutionary search. Two schemes are proposed in this section. One is based on the success rate. The other is based on mutation's step size.

3.1. *Success Rate Based DLB Scheme—DLB1*

The lower bound in EP has a major impact on how evolutionary search is conducted. A large lower bound encourages long-range search and makes escaping from a poor local optimum easier, while a small lower bound is only good at exploitation in a small region. For an unknown function to be optimized, it is hard to predict when to explore in a large space and when to exploit in a small region.

We propose that population information can be used to guide the tradeoff between coarse-grained exploration and fine-grained exploitation. Instead of using component level adaptation [15], such as the "1/5 success rule" [9, p. 110], we introduce a dynamic lower bound scheme based on the success rate of the whole *population*. The population level adaptation may provide us with richer and more accurate information about the search because it is the whole population which is evolving, not just separate individuals. In particular, if the success rate of the population is high, the lower bound will be increased. If it is low, the lower bound will be decreased. That is, the lower bound can be updated according to the following rule:

$$\eta_{-}^{\kappa+1} = \eta_{-}^{\kappa}\left(\frac{S_\kappa}{A}\right), \tag{3}$$

where S_κ is the success rate at generation κ and A is a reference parameter, which has been set between 0.25 and 0.45 in our experiments. A here is nice because it gives us a handle on defining what "large" and "small" mean in terms of lower bound. The success rate S_κ is obtained by first computing the number of offspring selected for the next generation and then taking the ratio of successes to all offspring. It will be shown later in the paper that this dynamic lower bound update scheme works very well, at least for the benchmark functions we have tested.

The update scheme in Eq. (3) contains the reference rate A, which is a parameter needs to be determined by the user. It is a convenient way for a user to define what is large and what is small for his/her problem. However, it may be inconvenient for a user which has little knowledge about his/her problem and thus unable to decide what is large or small. The next subsection proposes a parameter-free dynamic lower bound updating scheme that can get around this problem.

3.2. Mutation Step Size based DLB Scheme—DLB2

The survival of an offspring is a good indicator that the mutation which generated this offspring may be a good one. It is the right mutation applied to the right parent that has generated a successful offspring. The mutation performed by Eq. (2) described in Section 1 is additively applied with the normal distributed random number. DLB2 uses the median of the mutation step size from all accepted (successful) offspring as the new lower bound for the next generation.

Note that the mutation step size added to the j-component of the object vector can be described as:

$$\delta_i(j) = \eta_i'(j)N_j(0,1). \tag{4}$$

We first calculate the average mutation step size from all accepted (successful) offspring:

$$\overline{\delta}(j) = \frac{1}{m}\sum_{v=1}^{m}\delta_v(j), \quad j = 1, \ldots, n,$$

where m is the number of the accepted offspring. Then, the lower bound of η for the next generation is

$$\eta_-^{\kappa+1} = \text{median}\{\overline{\delta}(j), \quad j = 1, 2, \ldots, n\}. \tag{5}$$

This method regards the whole population as an aggregation point. The movement from generation to generation is like one point moving to another point. The $\overline{\delta}$ can be viewed as an n-dimensional vector approximating the movement of the generation. Using the median value of $\overline{\delta}(j)$ as the lower bound encourages, on average, a half of the components perform long-range exploration while the other half perform fine-tuned search.

4. Experimental Results

In order to evaluate the effectiveness of the two dynamic lower bound updating schemes, we compare them empirically with the fixed lower bound scheme which has been proven to be superior to EP without the lower bound.

4.1. Experimental Setup

Six benchmark functions were used, as shown in Table 2, in our experiments. The functions were numbered as in [5, 6] in order to facilitate further comparison with previous results. There are three

Table 2. The 6 benchmark functions used in our experimental studies, where n is the dimension of the function, $f_{\min}$ is the minimum value of the function, and $S \subseteq R^n$.

Test function	n	S	$f_{\min}$
$f_1(x) = \Sigma_{i=1}^n x_i^2$	30	$[-100, 100]$	0
$f_2(x) = \Sigma_{i=1}^n \lvert x_i\rvert + \prod_{i=1}^n \lvert x_i\rvert$	30	$[-10, 10]$	0
$f_5(x) = \Sigma_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	$[-30, 30]$	0
$f_9(x) = \Sigma_{i=1}^n [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	$[-5.12, 5.12]$	0
$f_{10}(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\Sigma_{i=1}^n x_i^2})$ $-\exp(\frac{1}{n}\Sigma_{i=1}^n \cos(2\pi x_i)) + 20 + e$	30	$[-32, 32]$	0
$f_{11}(x) = \frac{1}{4000}\Sigma_{i=1}^n x_i^2 - \prod_{i=1}^n \cos(\frac{x_i}{\sqrt{i}}) + 1$	30	$[-600, 600]$	0

Table 3. Comparison among IFEP with FLB, DLB1 and DLB2 on functions f_1, f_2, f_5, f_9, f_{10}, f_{11}.

F	Func. eval.	DLB2 Mean best	DLB1 Mean best	FLB Mean best	DLB2–DLB1 t-test	DLB2–FLB t-test	DLB1–FLB t-test
f_1	150,000	0	9.23E–30	5.85E–7	−2.02[a]	−12.69[a]	−12.69[a]
f_2	200,000	4.52E–26	9.13E–21	2.30E–3	−1.89	−106.35[a]	−106.35[a]
f_5	2,000,000	8.12E–4	1.61E–1	5.57E–1	−1.45	−3.75[a]	−2.14[a]
f_9	500,000	29.33	21.03	2.89	4.68[a]	20.48[a]	14.21[a]
f_{10}	150,000	7.69E–15	1.03E–14	6.33E–4	−4.47[a]	−11.58[a]	−11.58[a]
f_{11}	200,000	1.24E–2	9.60E–3	1.27E–1	1.05	−4.27[a]	−4.37[a]

"Mean best" indicates the mean best function values found in the last generation. "Func Eval" means the number of function evaluations.

[a]The value of t with 49 degrees of freedom is significant at $\alpha = 0.05$ by a two-tailed test.

unimodal functions: f_1 is the sphere function, f_2 is the test problem numbered 2.22 from [9, p. 341], and f_5 is the extended Rosenbrock function. The other three are multimodal functions with many local minima: f_9 is the Rastrigin function [16], f_{10} is a modified version of the Ackley function [17], and f_{11} is the Griewank function [18].

The EP algorithm used in our study was the improved fast evolutionary programming (IFEP) [6, 19]. The difference between IFEP and classical EP (CEP) is in Step 3 of the algorithm described in Section 1. Instead of generating one offspring using Gaussian mutation, IFEP creates two offspring, one by Gaussian mutation and the other by Cauchy mutation. The better one is then chosen as the offspring. Therefore, each mutation will use two function evaluations. Three IFEP experiments were conducted with different lower bound schemes: fixed lower bound (FLB), DLB1 and DLB2. The tournament size $q = 10$ for selection and the initial standard deviations 3.0 were used. For FLB, the population size $\mu = 50$ and the lower bound $\eta_- = 0.0001$ were set. For DLB1 and DLB2, $\mu = 10$ and the lower bounds were initialized to 0.1. The reason for using smaller population sizes for DLB is that they can indirectly perform more global searches by raising the lower bound. Therefore, using the population size to support more search diversity is not really necessary. The reference parameter of DLB1 is set as $A = 0.3$. The DLB1 and DLB2 schemes were updated every 5 generations using Eq. (3) and Eq. (5), respectively. The reason for not updating the lower bound in every generation is due to our belief that one or two generations are not really enough to obtain statistically useful information about the search.

4.2. *Results and Discussion*

Table 3 summarizes the experimental results of IFEP with and without a dynamic lower bound. All results have been averaged over 50 runs. It is clear that IFEP with a dynamic lower bound performed significantly better than IFEP with a fixed lower bound for five out of six functions. IFEP with a dynamic lower bound produced substantially better solutions for these five functions. The only exception is function f_9, which will be analyzed in more detail in the next subsection.

Comparing the results of DLB1 and those of DLB2, we discover that DLB2 has a better performance on f_1 and f_{10}, but worse on f_9. This can be explained by DLB2 being more greedy than DLB1. Thus, DLB2 has a better progress rate on finer exploitation, but worse ability on global exploration. Another advantage of DLB2 is that its design does not introduce any new parameters. For DLB1, the reference parameter A is a new parameter. However, the results of our IFEP were not very sensitive to the change of A values.

Figures 5–10 show the average evolutionary processes for the six benchmark functions. These processes show that IFEP with the DLB schemes have better convergence rates than IFEP with FLB for all functions except for f_9 where IFEP with the DLB schemes were trapped in the local optima and stagnated. Comparing the two DLB schemes, DLB2 performed better than DLB1 for most functions. The only exception was again f_9.

The experimental results in Figs. 5–10 showed that IFEP with FLB usually stagnated as the whole population moved close to the global optimum. On the other hand, the DLB schemes were able to control the search

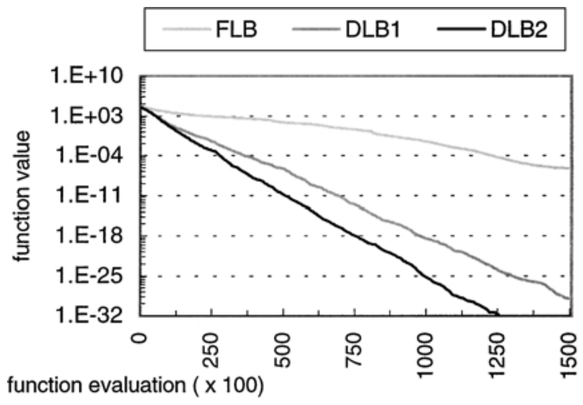

Figure 5. Comparison among IFEP with the FLB, DLB1 and DLB2 schemes on f_1.

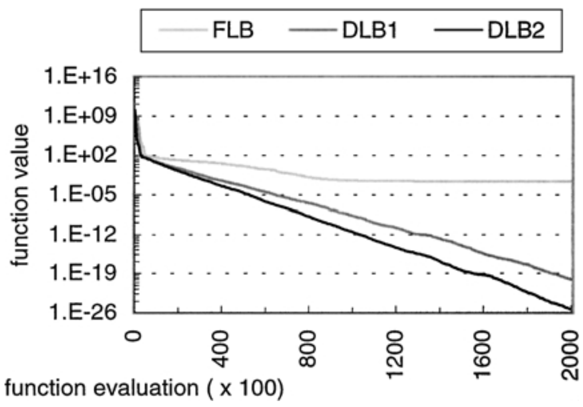

Figure 6. Comparison among IFEP with the FLB, DLB1 and DLB2 schemes on f_2.

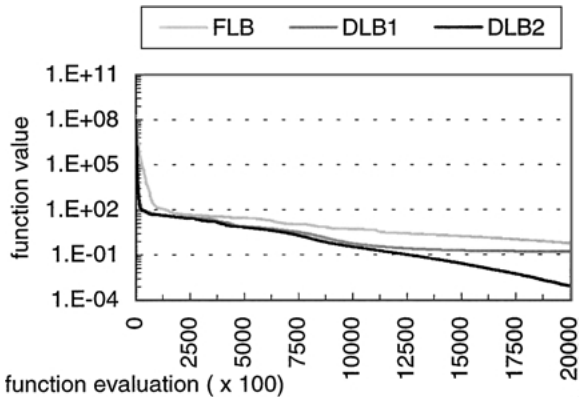

Figure 7. Comparison among IFEP with the FLB, DLB1 and DLB2 schemes on f_5.

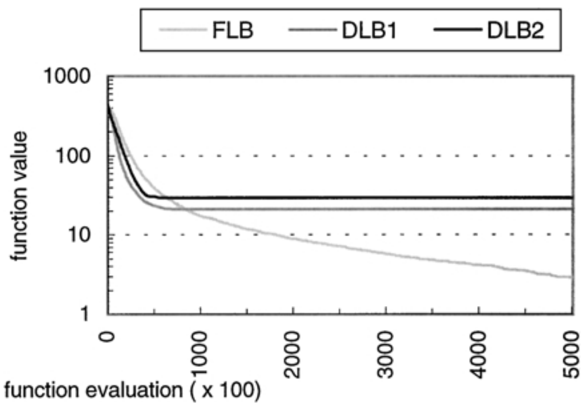

Figure 8. Comparison among IFEP with the FLB, DLB1 and DLB2 schemes on f_9.

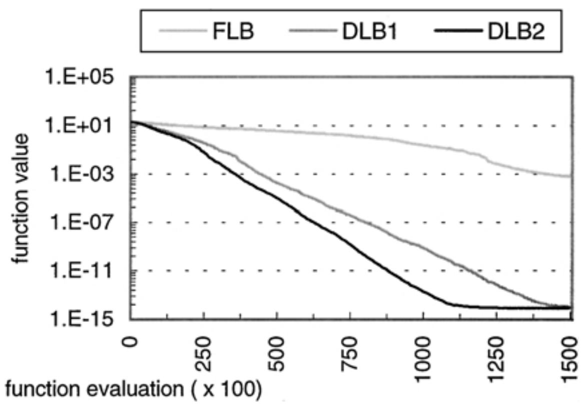

Figure 9. Comparison among IFEP with the FLB, DLB1 and DLB2 schemes on f_{10}.

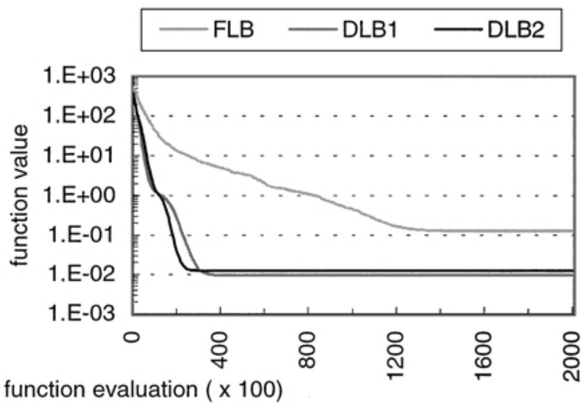

Figure 10. Comparison among IFEP with the FLB, DLB1 and DLB2 schemes on f_{11}.

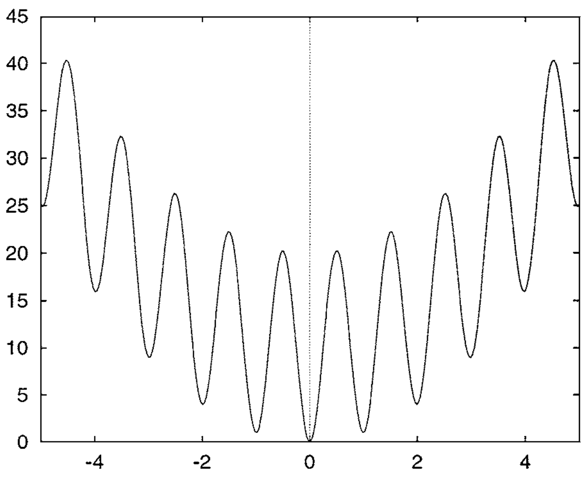

Figure 11. One dimension landscape for the Rastrigin function (f_9).

step size and quite effective and efficient in finding a near optimal solution.

4.3. *Why Is f_9 Difficult for IFEP with DLB*

The major reason for DLB's poor performance on function f_9 is the small η value. The DLB schemes are rather greedy in the sense that they try to imitate good mutations. Both DLB schemes make use of success rate which depends on successful mutations. However, successful mutations in one region (neighborhood) may not be successful anymore in a different region in the search space. Hence, using successful mutations to adjust lower bounds may fall into traps.

Figure 11 shows the one dimension landscape of the Rastrigin function (f_9). In this case, if individuals in a population fall into one of the deep valleys, the lower bound will be reduced gradually since individuals are more likely to go down the valley than to jump to a better point in the next valley. Large jumps may not pay off unless the point jumped to is better than the current one. Hence there is no incentive of increasing the lower bound.

5. Conclusions

The lognormal self-adaptation in EAs without a lower bound does not work very well. A lower bound for the search step size is needed in order for EAs to work effectively and efficiently. However, the optimal lower bound is problem dependent. A trial-and-error process often has to be used to find a good lower bound. This paper proposes two dynamic lower bound schemes where the lower bound changes dynamically during evolution. The first scheme, i.e., the success rate based dynamic lower bound scheme, combines population-level adaptation of the success rate with the component-level self-adaptation of the adaptive parameters to optimize evolutionary performance. The second scheme, i.e., the mutation step size based dynamic lower bound scheme, uses the median of the average mutation step sizes to set the lower bound, so that both long-range exploration and small-region exploitation are considered. Both schemes compare favorably with the fixed lower bound method on the set of benchmark functions we tested. However, when tackling a problem whose fitness landscape consists of many deep and narrow valleys, the dynamic lower bound schemes may not work well. They tend to reduce the lower bound to an overly small value. More work needs to be done in this area.

References

1. X. Yao, "An overview of evolutionary computation," *Chinese Journal of Advanced Software Research*, vol. 3, no. 1, pp. 12–29, 1996.
2. T. Bäck and H.-P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary Computation*, vol. 1, no. 1, pp. 1–23, 1993.
3. T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford University Press: New York, 1996.
4. H. Mühlenbein and D. Schlierkamp-Voosen, "Predictive models for the breeder genetic algorithm I. Continuous parameter optimization," *Evolutionary Computation*, vol. 1, no. 1, pp. 25–49, 1993.
5. X. Yao and Y. Liu, "Fast evolutionary programming," in *Evolutionary Programming V: Proc. of the Fifth Annual Conference on Evolutionary Programming*, edited by L. Fogel, P. Angeline, and T. Bäck, MIT Press: Cambridge, MA, pp. 451–460, 1996.
6. X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, pp. 82–102, July 1999.
7. K.-H. Liang, X. Yao, Y. Liu, C. Newton, and D. Hoffman, "An experimental investigation of self-adaptation in evolutionary programming," in *Evolutionary Programming VII: Proc. of the Seventh Annual Conference on Evolutionary Programming*, edited by V. Porto, N. Saravanan, D. Waagen, and A. Eiben, vol. 1447 of Lecture Notes in Computer Science, pp. 291–300, Springer: Berlin, 1998.
8. D. Fogel, "A comparison of evolutionary programming and genetic algorithms on selected constrained optimization problems," *Simulation*, vol. 64, no. 6, pp. 397–404, 1995.

9. H.-P. Schwefel, *Evolution and Optimum Seeking*, Wiley: New York, 1995.
10. D. Fogel, L. Fogel, and J. Atmar, "Meta-evolutionary programming," in *Proc. of the 25th Asilomar Conference on Signals, Systems and Computers*, edited by R. Chen, Maple Press: San Jose, CA, pp. 540–545, 1991.
11. K. Chellapilla, "Combining mutation operators in evolutionary programming," *IEEE Transactions on Evolutionary Computation*, vol. 2, no. 3, pp. 91–96, 1998.
12. H.-G. Beyer, "Toward a theory of evolution strategies: Self-adaptation," *Evolutionary Computation*, vol. 3, no. 3, pp. 311–348, 1995.
13. H. Larson, *Introduction to Probability Theory and Statistical Inference*. 3rd ed., Wiley: New York, 1982.
14. W. Feller, *An Introduction to Probability Theory and Its Applications*, vol. 1, 3rd ed., Wiley: New York, 1968.
15. P. Angeline, "Adaptive and self-adaptive evolutionary computation," in *Computation Intelligence: A Dynamic System Perspective*, edited by Y. Palaniswami, R. Attikiouzel, R. Marks, D. Fogel, and T. Fukuda, IEEE Press: Piscataway, NJ, pp. 152–163, 1995.
16. A. Törn and A. Žilinskas, *Global Optimization*, vol. 350 of Lecture Notes in Computer Science, Springer-Verlag: New York, 1989.
17. D. Ackley, *A Connectionist Machine for Genetic Hillclimbing*, Kluwer: Boston, MA, 1987.
18. A. Griewank, "Global optimization by controlled random search," *Journal of Optimization Theory and Application*, vol. 34, no. 1, pp. 11–39, 1981.
19. X. Yao, G. Lin, and Y. Liu, "An analysis of evolutionary algorithms based on neighbourhood and step sizes," in *Evolutionary Programming VI: Proc. of the Sixth Annual Conference on Evolutionary Programming*, edited by P. Angeline, R. Reynolds, J. McDonnell, and R. Eberhart, vol. 1213 of Lecture Notes in Computer Science, Springer: Berlin, pp. 297–307, 1997.

Ko-Hsin Liang is a Ph.D. candidate in the School of Computer Science, University College, University of New South Wales, ADFA, Canberra, Australia. His research interests are evolutionary algorithms and optimization.

Xin Yao received his BSc from the University of Science and Technology of China (USTC), Hefei, P.R. China, in 1982, his MSc from the North China Institute of Computing Technologies (NCI), Beijing, P.R. China, in 1985, and his PhD from USTC in 1990.

He is a professor of computer science in the School of Computer Science, the University of Birmingham, UK. He was an associate professor in the School of Computer Science, University College, the University of New South Wales, Australian Defence Force Academy (ADFA), before joining Birmingham. He held post-doctoral fellowships in the Australian National University (ANU) and the Commonwealth Scientific and Industrial Research Organisation (CSIRO) in Australia in 1990–1992.

He is/was a chair/co-chair of many international conferences, including PPSN'2000, IEEE ECNN'2000, CIEF'2000, CEC'99, ICCIMA'99, IEEE ICEC'98, SEAL'98, etc., an associate editor of *IEEE Transactions on Evolutionary Computation* and *Knowledge and Information Systems: An International Journal* (Springer), and a member of the editorial board of four other international journals. He is a senior member of IEEE, chair of the IEEE NNC Technical Committee on Evolutionary Computation, and the current president of the Evolutionary Programming Society. His research interests include evolutionary computation, evolvable hardware, neural network ensembles, global optimisation, computational time complexity of stochastic algorithms, and data mining.

Charles Newton is a Professor and Head of the School of Computer Science, University College, University of New South Wales, ADFA. His major research interests are in group decision support systems, simulation and optimization.