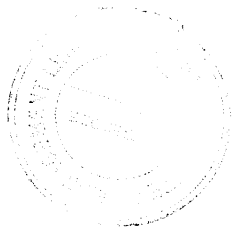
IIMK/WPS/13/IT/2007/02

Performance Analysis of Self-Adaptive Evolutionary Computation Methods

Anjan Kumar Swain[1],



[1] Associate Professor, Indian Institute of Management Kozhikose, IIMK Campus PO-673570, Kerala, India
Email: akswain@iimk.ac.in

Performance Analysis of Self-Adaptive Evolutionary Computation Methods[1]

This paper concerns with the detailed analysis of the performance of self-adaptive evolutionary computation algorithms. Various causes of premature convergence in these methods have been established. Subsequently, formulation of two new evolutionary algorithms has been discussed. The potentiality of these methods has been verified on eight popular test functions.

Keywords: Evolutionary Computation; Self-adaptive; Fast Evolutionary Computation; Dynamic Lower Bound

1. Introduction

The popular self-adaptive evolutionary computation (EC) algorithms are associated with many inherent problems such as premature convergence, slow speed, low accuracy etc. Thus, the effective use of these algorithms necessitates a thorough analysis why and under what conditions they fail. Further, the analysis also provides new ideas to develop improved algorithms to cater with the problems. In addition, this knowledge helps in the development of novel and efficient EC algorithms to solve various problems where accuracy and speed play important roles. In this paper, the causes of premature convergence in the prominent self-adaptive EC methods have been established and then, substantiated through computer simulation.

One of the important reasons of premature convergence may be the lack of system landscapes information in the learning equations of self-adaptive EC algorithms. Further, intuitively it can be realised that the introduction of as much system information as possible into the learning process can effectively guide the search to reach quickly its destination. In addition, two system behaviours have been identified for incorporation into the learning process. The first one is the fitness of the system, as used extensively in evolutionary programming (EP) [1] based methods in the early nineties. The other one is the distance information, i.e., the distance of a particular individual from other individuals, especially from the global optimum. Incorporation of distance and fitness information into the learning process has been extensively used in GA-related work. However, the use of distance information in EP and evolution strategy (ES) is not a mandatory.

Usually, distance related information is used in GAs and ESs inside crossover operators. The commonly used intermediate or generalised intermediate crossover operator generates offspring from two parent individuals by an amount related to the *genotypic distance* between the position vectors of each of the object variables of the respective parents. Here, genotypic distance is defined as the distance measured in the genotypic space or solution space.

Also, the unimodal normal distribution crossover proposed by Ono and Kobayashi [2] uses the distance among three parents to guide the search process.

In any global optimisation problem, the distance of an individual from the global optimum is known then the problem becomes more deterministic and easy to solve. Unfortunately, the global optimum in a problem is not known in advance.

[1] Referencing style followed: Journal of Information and Management, Elsevier B.V.

2. Self Adaptive Evolutionary Computation Methods

2.1 Self-Adaptation of Strategy Parameters

One of the important facets of recent EC methods is the use of self-adaptation. Self-adaptive methods modify the individual representation by incorporating strategy parameters into them. Thus, the ith individual p_i in a population pool P can now be redefined as:

$$p_i = \{(p_{ij}, \sigma_{ik}, \alpha_{il}) \mid j = 1, ..., n_o; k = 1, ..., n_\sigma; l = 1, ..., n_\alpha\}, \forall i \in \{1, ..., \mu\} \quad (1)$$

where n_o, n_σ, and n_α are the number of object variables, number of standard deviations and number of angles, respectively, and σ_{ik} and α_{il} are the standard deviations and rotation angles, respectively.

The working principle of self-adaptation can be described as follows: (i) evolve (using mutation and/or recombination) the strategy parameters; (ii) evolve the object variables using the already evolved strategy parameters. Hence, only those object variable values associated with good strategy parameter values will survive.

The use of self-adaptation in EC methods started in the ES community [3]. Thereafter, a plenitude of publications on this facet established its importance in the broad field of EC [4-8]. In evolutionary programming (EP), the use of self-adaptation was started by Fogel in the early nineties [1], and now it is almost mandatory [9-10] for all EP-based methods. Whereas, the importance of self-adaptation in GA is less established. A very few publications in GA addressed this vital issue [11-16].

In ESs, three types of self-adaptive methods are in use [16]: (i) hierarchically organised population-based meta-ES[17]; (ii) adaptation of covariance matrix (CMA) determining the probability distribution for mutation [6-18]; and (iii) explicit use of self-adaptive control parameters [7-19]. Out of these three categories, the third variety is extensively used by ES and EP researchers. In this research work, emphasis has also been placed on this third category of self-adaptation method. This is primarily because this method is well tested and studied from both empirical and theoretical standpoints, and needs the least computational time amongst the three. As such, no further studies on meta-ES and CMA methods have yet been pursued.

Now, the most prominent methods of self-adaptation in practice will be described. Referring to the generalised representation of an individual in Eq.(1), the mutation operator works by adding an n-dimensional normally distributed random variable $\mathbf{N} = \mathbf{N}(\mathbf{0}, \breve{\mathbf{C}})$ with expectation of $\mathbf{0}$ and covariance matrix $\breve{\mathbf{C}}$:

$$\breve{\mathbf{C}} = [c_{ij}] = \begin{cases} \mathrm{cov}(x_i, x_j) & i \neq j \\ \mathrm{var}(x_i) & i = j \end{cases} \quad (2)$$

and the probability density function is

$$\mathbf{f}_x(\mathbf{p}_1, ..., \mathbf{p}_{n_o}) = \frac{\exp\left(-\frac{1}{2}\mathbf{p}_i^T \breve{\mathbf{C}}^{-1} \mathbf{p}_i\right)}{\sqrt{\left((2\pi)^{n_o} \det(\breve{\mathbf{C}})\right)}} \quad (3)$$

where the covariance matrix is described by the mutated strategy parameters. The strategy parameters needed are included in the individual representation. Now, the self-adaptive methods can be categorised into four major classes [16]:

(i) Isotropic self-adaptation

Here, $n_\sigma = 1$, $n_\alpha = 0$, and $\mathbf{N} \sim \sigma\,\mathbf{N}(0,1)$. Hence, one standard deviation σ per individual is included in the representation and there is no rotation. Now, the update rule for the ith individual and $\forall j \in \{1, \ldots, n_o\}$ are:

$$\sigma_{ij}(t+1) = \sigma_{ij}(t)\,\exp(\tau_0 N_i(0,1)) \qquad (4)$$

$$p_{ij}(t+1) = p_{ij}(t) + \sigma_{ij}(t+1)\,N_{ij}(0,1) \qquad (5)$$

where the learning parameter $\tau_0 \propto n_0^{-\frac{1}{2}}$, $N_i(0,1)$ and $N_{ij}(0,1)$ are one-dimensional normally distributed random variates with expectation zero and standard deviation one. Here, $N_i(0,1)$ serves as a global factor allowing an overall change of the mutability in an individual-level and $N_{ij}(0,1)$ represents a local factor, thus allowing adjustment of each component of the individual acting at a component-level. Recently, Beyer (1996) showed mathematically that, for an $(1, \lambda)$-ES, the optimal value of the learning parameter $\tau_0 = \frac{C_{1,\lambda}}{\sqrt{n_o}}$, where $C_{1,\lambda}$ is the progress coefficient. The self-adaptation scheme, known as Gaussian self-adaptation as originally proposed by[1], is a particular case of Eq.(6.4) [5]. Now, for a small τ_0 we have

$$\sigma_{ij}(t+1) = \sigma_{ij}(t) + \sigma_{ij}(t)\,\tau_0 N_i(0,1) \qquad (6)$$

This relationship for obtaining the self-adaptive feature was used extensively in the works by Fogel in the early nineties [1]. Hence, Fogel's Gaussian self-adaptive method is the self-adaptive rule in Eq.(5) for small settings of the strategy parameter. Bäck and Schwefel [20] experimentally verified these observations on time varying sphere models.

(ii) Non-isotropic self-adaptation

Here, $n_\sigma = n_o$, $n_\alpha = 0$, and $\mathbf{N} \sim \mathbf{N}(0, \sigma_{ik}\,1)$. Thus, all the object variables p_{ij}, $\forall i \in \{1, \ldots, \mu\}$ and $\forall j \in \{1, \ldots, n_o\}$ have their own mutation strength σ_{ik} with $k = j$. Now, the update rules are:

$$\sigma_{ij}(t+1) = \sigma_{ij}(t)\exp(\tau' N_{ij}(0,1) + \tau N_i(0,1)) \qquad (7)$$

$$p_{ij}(t+1) = p_{ij}(t) + \sigma_{ij}(t+1) N_{ij}(0,1) \qquad (8)$$

where the exogenous parameters τ and τ' are set to $\left(\sqrt{2\sqrt{n_o}}\right)^{-1}$ and $\left(\sqrt{2n_o}\right)^{-1}$, respectively [21]. This non-isotropic self-adaptation method is universally followed to solve most of the problems in recent variants of EP and ESs.

(iii) Correlated self-adaptation

Here, $n_\sigma = n_o$, $n_\alpha = n_o(n_o-1)/2$, and $\mathbf{N} \sim \mathbf{N}(0, \check{\mathbf{C}})$. Thus, all the object variables p_{ij}, $\forall i \in \{1, \ldots, \mu\}$ and $\forall j \in \{1, \ldots, n_o\}$ are associated with distinct mutation strength σ_{ik} with $k = j$, and rotation angles α_{il}, $\forall l \in \{1, \ldots, n_o(n_o-1)/2\}$. The vectors σ_i and α_i are used in calculating the elements of the covariance matrix of the n-dimensional normal distribution with the covariances c_{ij}, $\forall i \in \{1, \ldots, (n_o-1)\}$ and $\forall j \in \{i+1, \ldots, n_o\}$ that are represented by rotation angles α_k (where $k = \left(\frac{1}{2}(2n_o - i)(i+1) - 2n_o + j\right)$) that describe the co-ordinate rotation necessary to transform an uncontrolled mutation vector into a correlated one. Here, α_k characterises the rotation angle with respect to the co-ordinate axes i and j. Now, the rotation angles and covariances are related by the following expression:

$$\tan(2\alpha_k) = \frac{2c_{ij}}{\sigma_i^2 - \sigma_j^2} \qquad (9)$$

Hence, a total of n_o mutation strengths and $n_o(n_o-1)/2$ rotation angles are used in each individual to update the object variables:

$$\sigma_{ij}(t+1) = \sigma_{ij}(t)\exp(\tau N_{ij}(0,1) + \tau' N_i(0,1)) \qquad (10)$$

$$\alpha_{ij}(t+1) = \alpha_{ij}(t) + \beta N_{ij}(0,1) \qquad (11)$$

$$P(t+1) = P(t) + N(0, \breve{C}(\sigma(t+1), \alpha(t+1))) \qquad (12)$$

where $N(0, \breve{C}(\sigma(t+1), \alpha(t+1)))$ represents the correlated mutation vectors with a zero mean and covariance matrix $\breve{C}$, and $\beta \approx 0.0873$ (i.e., 5 degrees).

(iv) Mixed self-adaptation

Here, $1 < n_\sigma < n_o$. Thus, the number of mutation strengths are less in number than the object variables. In this case, the mutation strength for the object variables after n_σ can be set to the last mutation strength σ_{n_σ}, and the rest of the process is similar to either non-isotropic or correlated self-adaptation schemes described above.

2.2. Advances in Self-adaptive Evolutionary Computing Methods

Recently, Cauchy mutation instead of Gaussian or normal mutation for continuous parameter optimisation has been gaining more interest. Yao and Liu [22-23], and Yao et al. [24] used a non-isotropic self-adaptation method, with the only change made being to replace the normally distributed random variable $N_{ij}(0,1)$ in Eq.(8) by an one-dimensional standard Cauchy random variable $\Delta_{ij}(0,1)$. The probability density function (pdf) of a Cauchy variate can be described as [25]

$$(\pi\varphi)^{-1}\left[1+\left(\frac{x-\theta}{\varphi}\right)^2\right]^{-1} \qquad (13)$$

$$\frac{1}{2} + \pi^{-1}\tan^{-1}\left(\frac{x-\theta}{\varphi}\right) \qquad (14)$$

This distribution is symmetrical about $x = \theta$. Also, it does not possess a finite expected value or standard deviation; its finite moments only exist for order less than one. Hence, a standard form for the Cauchy distribution $\Delta(0,1)$ can only be obtained by substituting zero for θ and one for φ. So, the standard pdf and cdf are represented as

$$\text{pdf} = \pi^{-1}\left(1+x^2\right)^{-1} \qquad (15)$$

$$\text{cdf} = \frac{1}{2} + \pi^{-1}\tan^{-1}x \qquad (16)$$

Hence, the standard Cauchy variate $\Delta(0,1)$ is nothing but simply a problem independent unimodal, symmetric random variate with heavier tails than the normal variate.

These Cauchy-based modified algorithms are known as fast EP (FEP) or fast ES (FES) in the context of EP and ES research, respectively. FEP and FES perform much better on multimodal functions with many local minima, while being comparable to CEP in most cases for unimodal and multimodal functions with few local optima. This better performance is attributed to the much flatter tail of the Cauchy variate that helps it to escape local optima very easily. Yao et al. [22]-[24] reported that the long jumps that the Cauchy variate perform compared to normal variates accelerate the convergence speed during initial period of search and are detrimental towards the final convergence. This is responsible for the poor performance of the Cauchy variate-based methods on certain functions. Hence, Yao et al. showed that a

blend of Cauchy and normal variate might help to find an overall better performance. This motivated subsequent empirical works by Sarvanan and Fogel [25] and Chellapilla [26] on hybrid Cauchy and Gaussian–based algorithms. The initial success of empirical results lead Rudolph [27] to analyse theoretically the relative behaviours of Cauchy and normal mutations. He reported that the normal variates provide faster local convergence on convex functions, whereas Cauchy variates are more helpful in escaping local optima. Hence, the value of empirical test results should not be under estimated as they ultimately help also in the development of the theory. Further, Yao and Liu [23] reported that updating standard deviations before or after the object variables essentially do not make much difference. In the FEP algorithm the offspring generated are:

$$p_{ij}(t+1) = p_{ij}(t) + \sigma_{ij}(t+1)\Delta_{ij}(0,1) \qquad (17)$$

The major advantages of non-isotropic self-adaptation mechanisms are evident when optimising multi-parameter tasks with each parameter having its own search bound remarkably distinct from each other. Thus, in practical applications, this is much useful. It also avoids the demerits of fitness dependent variation operators. The fitness dependent variation operators may cause the following problems [28]: (i) for the same standard deviation, two optimization problems differing by a scale factor give rise to a fitness function which also differs by the same scale factor; (ii) in the optimization problems, where the fitness value increases rapidly with increase in the dimensions of the problem, this increases the standard deviation, which in turn drags the process to instability. As these methods do not depend upon the process knowledge, hence acts independent of the problem. The only tie up with problem knowledge is inside the selection operation, which derives the knowledge from the phenotype to remove unfit individuals from the population pool.

2.3. Analysis of Self-Adaptive Evolutionary Computation Methods

The potential demerits of self-adaptive methods that can be observed are: (i) only the selection process guides these methods toward the optimal solution. Without selection, it is no more than a random search. Hence, this process-blind variation operator cannot know the nature of the fitness landscape. Hence, this may prove fatal in the presence of many local optima. Thus, the effectiveness of this method largely depends on the selection method; (ii) because the fitness depends on the object variable values only, two individuals with same object variables and with substantially different standard deviations are treated as equivalent. This may help the solutions to loose diversity and thereby become trapped in prominent local optimum; (iii) because of the use of the multiplicative lognormal metaheuristics strategy to mutate the standard deviations, which helps the standard deviations to fall sharply to very low values. These low values of standard deviations effectively provide no modifications of the object variables, which therefore stagnates at a fixed value and behave as if struck at some local minima or wandering on a flat plateau; (iv) a problem may arise due to the use of normal variates with a standard deviation that remains at a constant value of one. Thus, for functions having a global optimum amidst many local optima, it is very probable that a very small initial search bound will fail to provide adequate search to locate the optimal solution. Hence, self-adaptive methods are likely to perform better with large initial search domain, and when there are not many local optima at the very close proximity of the global optimum; (v) again if the initial search bound is very large and the scale factor of the random variate is one with lognormal strategy for the variation of the strategy parameters, then mutation steps drop to such a low value well before reaching the global optimum thereby leading to premature convergence. Hence, this can be seen that large initial search bounds always drag the solution to suboptimal points. This problem further aggravates with higher dimensions of the problem and many local optima. It is clear from the above, and from the reasons to be discussed in the next paragraph, that it is highly prone to fall in local optima.

Hence, the problems inherent in self-adaptive methods described above usually enhance the probability of such methods becoming trapped at some local optima, thereby leading to premature convergence. This was first reported by Liang et al. [30]. They observed that self-adaptive evolutionary algorithms are not even able to find a global optimum for simple functions and suggested the use of a fixed lower bound on

each of the mutated parameters to improve the overall performance of these algorithms. Subsequently, Liang et al. [31] proposed a dynamic lower bound on σ_{ij} that resulted in performance improvement on some test functions. Due to the lack of any concrete method to avoid the premature convergence, researchers normally use a fixed lower bound on the mutation variables [27]. Recently, Glickman and Sycara [32] presented three conditions that may be the possible causes of the premature convergence of all self-adaptive evolutionary algorithms. Of course, all these fall within the demerits of self-adaptive methods discussed above. However, their experimentation and assertions are based on training recurrent artificial neural networks (RANN). The avoidance of this problem is of much important in EC research community, and is an open research problem.

3. Modified Self-Adaptive EC Algorithms

3.1. Differential Step Recombination Based EC Methods

In this section, the concept of distance as discussed earlier has been used to design a recombination or crossover operator. The recombination operator depends on the differential distance or step of the object variables of an individual from the corresponding object variables of the fittest individual.

In this paper, the differential step based EC method uses a Cauchy distribution. Of course, any other type of slowly varying continuous distributions including normal distribution can be used. Here, the standard deviation or the scaling factor to be used has been made proportional to the distance d_{ij}:

$$\sigma_{ij} \propto d_{ij} = k\, d_{ij} \qquad (18)$$

where $k = \beta\sqrt{\frac{w}{\pi}}$ is a proportionality constant, with $\beta = 0.1$ and w as the width of the user defined search domain. Hence, σ_{ij} is represented by

$$\sigma_{ij} = k\,|p_{ij} - p_{kj}| \qquad (19)$$

In Eq.(19), the value of k is proportional to the square root of the width of the feasible region. Thus, k is always a non-zero positive number; its value will be one when the search width w = 314.16, which in turn mean that $\sigma_{ij} = d_{ij}$; and its value will be two for w = 1256.64. Hence, it is obvious that k > 2 will rarely be used in practice due to its extremely large search bounds. Now, effectively the action of k can be judged for three distinct sets of values such as $0 < k < 1$, $k = 1$ and $k > 1$.

Now, the properties of a standard Cauchy distribution around zero with a scale factor σ_{ij} can be understood as follows: (i) it has much longer and heavier tails than a normal distribution, thus generates large values more often; (ii) approximately 50% of the values lie within σ_{ij} on either side of zero, 80% of values will be within $3\sigma_{ij}$ and 98% will be within $31\sigma_{ij}$ on either side of zero.

Then, the three cases of k can be analysed as follows:

(i) For $k < 1$, the standard deviation is reduced to a lower value from the value equal to its distance d_{ij} from the best individual. Hence, the smaller the width, the larger the exploitation. Thus, for a very narrow search bound, it can work perfectly and enhance the convergence rate.

(ii) For $k = 1$, $\sigma_{ij} = d_{ij}$, and thus, the search progress exactly depends on the genotypic distance of the object variables from the respective object variables of the best individual.

(iii) For $k > 1$, the σ_{ij} value increases, allowing the search to progress more openly to cover a large part of the feasible region. Thus, in this case, possibility of exploration of the search space is much greater.

Then, to speed up the process of convergence, a directionality feature associated with each of the object variables of the parent individual, determined with respect to the object variables corresponding to the best individual, has been introduced. The direction (or the sign) of the generated absolute Gaussian random variable is decided as per the position of the particular object variables of the parent individual with respect to the object variables of the best individual in that generation. This directionality feature can be expressed as

$$dir(p_{ij}) = sgn(p_{ij} - p_{kj}) \qquad (20)$$

where "sgn" calculates the sign of the argument within the bracket.

In Eq.(20), the genotypic distance $|p_{ij} - p_{kj}|$ for the best quality individual in the population pool is zero, which in turn forces σ_{ij} to zero. This does not update the object variables of the fittest individual.

This is undesirable in the context that the fittest should have more opportunity to exploit its own neighborhood to generate better offspring (local search). To circumvent this, a constant offset z_c has been introduced. Hence, for the fittest individual $\sigma_{ij} = z_c$, this small standard deviation increases the probability of producing offspring in a very close vicinity of the parent. Whereas, with the increase in genotypic distance for other individuals, the exploration of new and unknown areas of the search domain increases, which progressively decreases the exploitation capability. Now, σ_{ij} can be expressed as

$$\sigma_{ij} = k\,|p_{ij} - p_{kj}| + z_c \qquad (21)$$

The Cauchy random variables are used to guide the search process to generate offspring. Hence, the jth object variable of the ith offspring generated from the corresponding object variable of the parent, can be represented as

$$p_{ij} = p_{ij} - \sigma_{ij}\, dir(p_{ij})\, C_{ij}(0,1) \qquad (22)$$

It has been shown that the crossover operators used in practice by GAs and ESs research can at best generate offspring within a defined initial rectangular hyper body. Thus, this effectively generates offspring constrained to lie within the parent object boundaries. However, in the proposed method without the directional feature, the offspring can be anywhere in the search space. The incorporation of the directional feature imposes a restriction on the generated offspring such that the offspring will no longer be generated in the direction quite opposite to the fittest individual. Thus, it adds the ability to explore the search space more than that of the conventional recombination operators used in GAs and ESs, in addition to enhance the convergence speed of the process.

Hybrid Evolutionary Algorithm Development

The EC algorithms based on differential distance d_{ij} avoids all the demerits associated with the fitness dependent variation operators. Whereas, it lacks the advantages due to fitness dependence, and importantly the fitness landscape information. Hence, in order to take full advantage of both the contradictory features of fitness dependency and non-dependency, it is essential to include fitness information into the algorithm in some way or other, and then to counteract the demerits by some means. One of the possibilities that have been used here is the hybridisation of both the concepts within a single algorithmic framework.

Now, the concept of fitness can be introduced into the remaining mutation operator. This could be achieved very easily, as in the standard canonical EP (CEP) where the use of a fitness-based mutation operator is a common practice. The mutation operator in CEP varies directly with the square root of the fitness value of the individual. Thus, this concept has been used to develop the mutation operator directly. The only change made in BEP is that the normal distribution has been replaced by a Cauchy distribution. This can be described in the following paragraph.

This mutation operator uses a problem dependent deterministic factor ζ, which is selected such that it is directly proportional to the square root of the fitness score and inversely proportional to the problem dimensions, and is defined for the ith individual as

$$\zeta_i \propto \frac{1}{n}\sqrt{f(p_i)} = \frac{\alpha}{n}\sqrt{f(p_i)} \qquad (23)$$

where $f(p_i)$ is the fitness score associated with the ith individual and $\alpha = 0.01$ is a proportionality constant. Here, the fitness should always be positive, but if it happens to be negative then it should be set to an arbitrarily low value (Fogel, 1995), which has been set to 0.01 in all the simulation studies performed in this thesis. Then, it is used along with the andomness of the standard Cauchy distribution C(0,1) to escape from the local optima so that there is increased probability of directing the solution process toward the global optimum. Hence, the offspring can be represented as

$$p_{ij}(t+1) = p_{ij}(t) + C_{ij}(0,1)\,\frac{\alpha}{n}\sqrt{f(p_i)} \qquad (24)$$

where $C_{ij}(0,1)$, $\forall i \in \{1, 2, ..., \mu\}$ and $\forall j \in \{1, 2, ..., n_o\}$, represents a one-dimensional Cauchy random variate for the jth variable of the ith individual. As this mutation operator is related to the individual fitness score, it will suffer from the problems of solution instability for higher dimensional tasks [29,33]. This problem has been addressed here by limiting the variation to lie within twice the user-supplied search-width when the standard deviation score exceeds ten times the value of the search width. The factor ten has been chosen as it is very unlikely that the object variables will go beyond this relatively large limit. But, if this large limiting value is exceeded, this means the search process is wandering randomly away from the initially specified feasible region. Thus, it indicates that this happens due to the exceedingly high fitness values and as such the intention of the algorithm is not to search a long way from the present knowledge of the feasible region. Limiting the corresponding object variables to a value twice the magnitude of the upper bound ensures that no portion of the feasible search space is being excluded.

Here, the proposed recombination method generates one offspring from parents and fittest among them replaces the weak individual. Hence, the differential step recombination follows a short of elitist strategy. Thus, the μ individuals after recombination generate the final μ individuals to undergo the mutation process. The mutation operator produces λ offspring (here, $\lambda = \mu$) as per a $(\mu + \lambda)$ scheme of ES and EP. Then, a stochastic tournament selection, which is typically used in all EP-based algorithms, has been used to select μ individuals from $(\mu + \lambda = 2\mu)$ individuals consisting of both parents and offspring. These become the new parents in the population pool. This process is repeated until a prefixed termination criterion is satisfied.

The complete algorithm, which effectively encompasses both fitness and distance information into it, has been used in this paper and is named as the *hybrid evolutionary algorithm* (HEA). This algorithm, apart from the hybridisation of both fitness dependent and non-dependent features, borrows the concept of elitist recombination philosophy from GAs and uses other features from EP- or ES-based algorithms.

3.2. EC with a Dynamic Lower Bound

One of the major reasons for premature convergence is attributed to the rapid drop in the value of σ_{ij} to an extremely low value while the solutions or the object variables are still far away from the global optimum. Hence, this small value of σ_{ij} effectively does not add any knowledge to the search process, causing the solution to stagnate at a point other than the global optimum. This shows that the effect is likely to be more prominent if the feasible region or simply the search width is very large. Further, it confirms that the premature convergence has got some kind of relationship with the distance of objective variables from the global minimum. If the distance is large and σ_{ij} is very small then it leads to premature convergence. Hence, the concept of genotypic distance from the optimum would be helpful for avoiding

the problem of premature convergence. Keeping this in mind, dynamic lower bounds designated by b_{ij} on each of the standard deviation expressions as represented in Eq.(7) have been set in proportion to the distance d_{ij}:

$$\sigma'_{ij}(t+1) = \sigma_{ij}(t+1) + b_{ij} \qquad (25)$$

Hence, it is proposed that the lower bound b_{ij} on σ_{ij} corresponding to the object variable p_{ij} should vary in proportion to its distance from the jth object variable of the fittest individual in that population pool. In this way, the distance information is incorporated into the adaptation equations of strategy parameters. In addition, this lower bound is active at the component level of an individual. This serves the purpose of not allowing σ_{ij} to fall below its distance from the optimum in that generation. The operation of this lower bound can be explained as that when σ_{ij} is very small but the corresponding p_{ij} is far from the true global optimum, then the lower bound may be dominant and thus it may have greater control on the convergence.

Now, mathematically this lower bound can be represented as

$$b_{ij} \propto |p_{ij} - p_{kj}| \qquad (26)$$

The basic philosophy of real number mutation works by utilising a Gaussian random distribution such that small variations are more likely to occur compared to large variations. By extending this concept to b_{ij}, we have

$$b_{ij} = \gamma |p_{ij} - p_{kj}| N_{ij}(0,1) \qquad (27)$$

where $N_{ij}(0,1)$ is a normal distribution with zero mean and unity standard deviation, and the proportionality constant $\gamma = \frac{1}{\sqrt{n_o}}$ for CEP and for FEP $\gamma = \frac{1}{n_o}$. Clearly, the factor γ is system dependent, i.e., it is very likely to change for other variants of CEP.

It can be seen that, when the object variables are situated far from the subglobal and the initial search domain is very large, then the probability of b_{ij} being large is greater. Hence, this makes the mutated parameters to be larger than the initial σ_{ij} values. This can be avoided with the following heuristics:

$$\sigma_{ij}(k+1) = \begin{cases} \sigma_{ij} + b_{ij}; & \sigma_{ij} + b_{ij} < (\sigma_{ij})_{initial} \\ \sigma_{ij}; & \text{otherwise} \end{cases} \qquad (28)$$

With the progress in the evolution, the strategy parameter σ_{ij} reduces to a low value. Hence, at the start of the evolution process, the mutation is very large and so it does mostly exploration of the search space. Then, at later generations, the search domain narrows down and so it is most likely to exploit the search space. This concept of varying the mutation parameter to avoid the premature convergence leads to a novel dynamically-varying lower bound.

4. Simulation and Result Discussion

4.1. Test-Functions

For the experimental verification of the performance of the algorithms to find the global minimum, typical 8-benchmark functions have been considered in Yao et al. [24]. These functions are typically combinations of low dimensional and high dimensional, unimodal and multimodal, continuous and discontinuous, and stochastic and deterministic parts. These functions are shown in Fig.1.

4.2. Set-up for Computer Simulation

All the experiments on the CEP and FEP method have been performed under exactly the same conditions with initial standard deviation $\eta_{ij} = 3$, $\forall i \in \{1, ..., \mu\}$ and $\forall j \in \{1, ..., n_o\}$, and the same

initial population size μ = 100. The proposed HEA is conceptually altogether different from CEP and FEP. As such, HEA is not a self-adaptive method. Rather it is a heuristic approach with only one tuning parameter z_c, $0 \leq z_c \leq 1$. Here, z_c is an additive parameter and in general, it is very easy to tune additive parameters compared with tuning multiplicative parameters. The value of z_c mostly controls the final accuracy of the results. The population size for HEA is fixed at 50, so that the number of function evaluations is the same as that of CEP and FEP. In HEA, a population size of 50 with two stages of variation means that a total of 100 function evaluations are needed per generation. The tournament size for all the experiments with HEA, CEP and FEP were fixed at c = 10. For CEP and FEP, a constant lower bound $b_{ij} = 0.0001$, $\forall i \in \{1, ..., \mu\}$ and $\forall j \in \{1, ..., n_o\}$, on the strategy parameters has been included. At this lower bound, Chellapilla [26] reported better performance of FEP and CEP over that of no lower bound. All the results have been averaged over 50 runs. The statistical t-test has been used to study the statistical significance of the obtained results. In the case of HEA, the fitness score cane become negative for some functions and, to counteract this, the fitness score is set to a small positive value of 0.01 at the moment that it goes negative. Further, in the simulation of f_5, the object variables are constrained to stay within the initial search domain. This is because, its true optimum is at ∞ where the function value is $-\infty$. However, for all other functions, no such restrictions have been imposed, and the initial search domain is only used once while initializing the population pool.

4.3. Result Discussion for HEA

The proposed HEA performs better than CEP and FEP on all the unimodal functions f_1 - f_4. The behaviors of the functions are shown in Fig. 2 (a), (b), (c) and (d). Average best results are plotted to show the learning capability of HEA method. As shown in Fig 2(a), both the convergence speed and accuracy of the results of HEA are consistently better than those of FEP and CEP on all the high dimensional unimodal functions. On function f_1, which is a simple sphere model, HEA converges much faster and produces better results than FEP and CEP. This is because, once the optimum has been located, the differential step recombination operation tries to direct all other individuals in the population pool to reach the optimum thereby producing much faster convergence. Further, a small z_c at the optimum further exploits the neighbouring search space to generate increasingly accurate results. This reasoning can be extended to analyse the better performance of HEA on most of the test functions.

Function f_2 is the Rosenbrock's function with a very narrow and steep banana-shaped valley surrounding the global minimum. The thirty-dimensional function f_2 is a really difficult problem for all these methods. Although the performance of HEA on f_2 is better than FEP and CEP, it was not able to locate the global optimum. From the results, it can be seen that the standard deviations of the FEP method are better than those of HEA, which shows that HEA results are widely dispersed around the mean value. For step function f_3, CEP clearly stalls at the flat surface, exhibiting an inability to cope with functions having a flat objective function landscape. However, FEP performs better than CEP due to the long jumps of the Cauchy distribution, which largely prevents the search process from becoming stuck on the flat surface. Nevertheless, its convergence rate towards the global optimum is still not very fast. In contrast, HEA finds the global minimum very promptly. This superb performance may be attributed to the use of the two-staged variation operation along with Cauchy distribution. On the quartic function, f_4, which is a noise-based function, the overall performance of FEP is the worst. It is to be noted that CEP outperforms FEP on this function. The better performance of HEA suggests its effectiveness for noisy functions. Hence, on all the unimodal, high dimensional functions the performance of HEA is far better than CEP and FEP.

Fig. 2(e) shows that the performance of HEA is again much better than CEP and FEP on the multimodal functions f_5 with many local minima. Function f_5 consists of an extremely large number of local minima, where, within the specified search boundary of width one thousand spanning both sides of

the zero axis, the minimum is located at (420.9687, ..., 420.9687) with a function value of –12569.5. In this problem the upper and lower bounds of the search domain have to be strictly respected, otherwise its minimum will only be restricted by the minimum value of the computing machine.

HEA's performance is quite similar to FEP and CEP on most of the multimodal functions f_6 to f_8 that have few local minima. The corresponding performance curves are presented in Figs. 2(f), (g) and (h). Function f_6, the Shekel's foxholes, which contains a moderate number of local optima, is usually a difficult problem to solve. Interestingly, HEA finds the exact global optimum, whereas FEP and CEP remained away from the optimum. On function f_7, all the three methods yield equally satisfactory results. For the function f_8, the results after the stated number of generations remain almost same for all the methods. However, the convergence characteristics of HEA, as indicated in Fig.2, clearly show its much better performance over CEP and FEP. HEA achieves the desired results within only a few generations, whereas both CEP and EFP take a much longer duration. It can be observed from these figures that the rate of convergence of the mean fitness value of HEA is much faster than FEP and CEP on all the functions f_6 to f_8. This effectively provides sufficient evidence that HEA outperforms FEP and CEP on almost all the functions f_1 to f_8.

4.4. Result Discussion for EC Algorithm with Dynamic Lower bound

The results of CEP and FEP with and without lower bound were compared for all the 8-benchmark functions. Figs. 3(a), (b), (c) and (d) show the progress of the average best values of the population by CEP over 50 runs for the unimodal functions f_1 to f_4. It can be observed that, on all the functions, CEP with differential step lower bound (CEPDSLB) performs consistently better than CEP with no lower bound (CEPNLB) and CEP with fixed lower bound (CEPFLB). For functions f_2 and f_4, the results of CEPDSLB and CEPFLB are comparable. This shows that a fixed lower bound of 10^{-4} serves adequately on these functions. The results for f_6 show that the CEPFLB performs almost identically with that of CEPNLB. This indicates that CEPFLB does not yield better results on all the functions.

The performance of CEPNLB, CEPFLB, and CEPDSLB for high dimensional multimodal function with many local minima f_5 is shown in Fig. 3(e). Here, the performance of CEPFLB and CEPNLB do not differ. However, on all the function f_5, CEPDSLB outperforms CEPFLB and CEPNLB.

On the low-dimensional functions f_6 to f_8, the performance of all the methods is very similar. This is shown in Figs. 3(f), (g) and (h) and particularly the performance of CEPDSLB is impressively better than CEPFLB and CEPNLB. On function f_8, the convergence of CEPDSLB is faster than both CEPNLB and CEPFLB.

5. Conclusion

The major focus of this paper was on finding the possible causes of premature convergence of self-adaptive methods, and a fundamental analysis of the reasons thereof has been provided.

In this paper, the concept of distance and fitness has been used to develop better algorithms. The potential of this concept has been verified by applying it in the development of a hybrid evolutionary algorithm for global optimization and a differential step lower bound on the strategy parameters of a self-adaptive evolutionary algorithm. The HEA essentially behaves as a two stage variational algorithm. The first stage uses a new recombination operator, which adjusts the object variables in proportion to their distances from the corresponding object variables of the fittest individual. The second stage of variation is accomplished with a typical basic EP-style mutation operation. Both of the variational operators used

Cauchy distribution to provide the necessary random variation. The performance of the HEA method has been studied extensively on a test-suite of 8 benchmark problems of varied complexities. The results of HEA are then compared with that of FEP and CEP. On multimodal functions with many local minima, and unimodal functions of any complexity, HEA outperformed both FEP and CEP. For multimodal functions with few local minima, the performance of all these methods is quite similar.

This concept has been extended to employ a differential step lower bound on each of the strategy parameters has been formulated. This differential step lower bound has been designed to vary in proportion to the distance of the object variable from the corresponding object variable of the optimum. This very concept has been tested on CEP, and the corresponding method was named as CEP with differential step lower bound (CEPDSLB). The results were also tested on a 8-function test-bed and observed that, on all the functions, CEPDSLB outperformed both CEP with no lower bound and also CEP with fixed lower bound.

Function Description

Three-Dimensional Plots

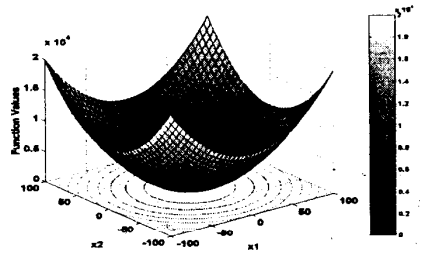
Sphere Model

$$f_1(x) = \sum_{i=1}^{no} x_i^2$$

$-100 \leq x_i \leq 100$

$n_o = 30$

$\min(f_1) = f_1(0,\ldots,0) = 0.$



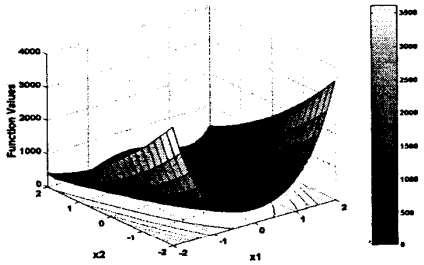
Generalised Rosenbrock's Function

$$f_2(x) = \sum_{i=1}^{n-1} \{100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\}$$

$-30 \leq x_i \leq 30$

$n_o = 30$

$\min(f_2) = f_2(1,\ldots,1) = 0.$



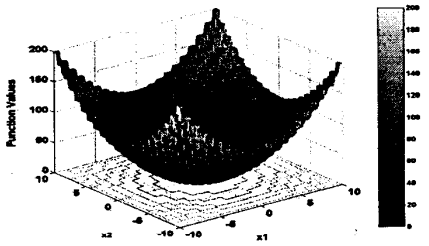
Step Function

$$f_3(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$$

$-100 \leq x_i \leq 100$

$n_o = 30$

$\min(f_3) = f_3(0,\ldots,0) = 0.$



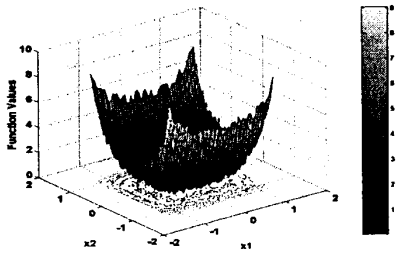
Quartic Function

$$f_4(x) = \sum_{i=1}^{n} i x_i^4 + random(0,1)$$

$-1.28 \leq x_i \leq 1.28$

$n_o = 30$

$\min(f_4) = f_4(0,\ldots,0) = 0.$



Generalised Schwefel's Problem 2.26

$$f_5(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{|x_i|}\right)$$

$-500 \leq x_i \leq 500$

$n_o = 30$

$\min(f_5) =$

$f_5(420.9687,\ldots, 420.9687) = -12569.5.$

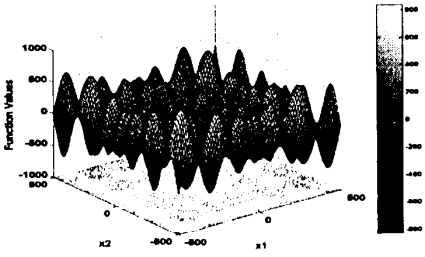


Fig. 1. (a) First 5-benchmark functions, where n_o is the function dimension, $f_{\min}$ is the minimum function value and SD is the user supplied search domain.

Shekel's Foxholes

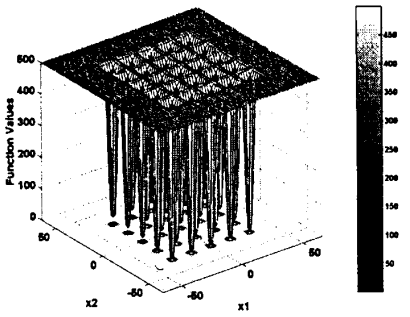
$$f_6(x) = \left[.002 + \sum_{j=1}^{25} \left(j + \sum_{i=1}^{n} (x_i - a_{ij})^6 \right)^{-1} \right]^{-1}$$

$-65.536 \leq x_i \leq 65.536$

$n_r = 2$

$\min(f_6) = f_6(-32,\ldots,-32) = 0.980004.$

a1 = [{-32.0,-16.0,0.0,16.0,32.0,-32.0,-16.0, 0.0,
16.0,32.0,-32.0,-16.0,0.0,16.0,32.0,-32.0,
-16.0,0.0,16.0,32.0,-32.0,16.0,0.0,16.0,32.0};
{-32.0,-32.0,-32.0,-32.0,-32.0,-16.0,-16.0,
-16.0,-16.0,-16.0, 0.0, 0.0, 0.0,0.0,0.0,16.0,
16.0,16.0,16.0,16.0,32.0,32.0,32.0,32.0, 32.0}].



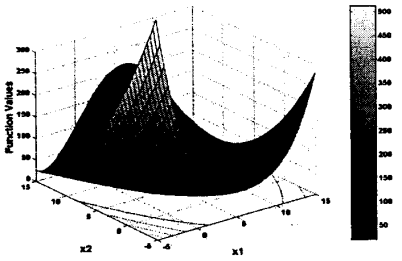
Branin Function

$$f_7(x) = \left(x_2 - (5.1/(4\pi^2))x_1^2 + (5/\pi)x_1 - 6\right)^2 + 10\left(1 - (1/8\pi)\right)\cos x_1 + 10$$

$-5 \leq x_1 \leq 10, 0 \leq x_2 \leq 15$

$n_r = 2$; $\min(f_7) = f_7(-3.142, 12.275) =$

$f_7(3.142,2.275) = f_7(9.425,2.425) = 0.398.$



Hartman's Family 1

$$f_8(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{3} a_{ij}\left(x_j - p_{ij}\right)^2 \right]$$

$0 \leq x_i \leq 1$

$n_r = 3$

$\min(f_8) = f_8(0.114,0.556,0.852) = -3.86$

a = [{3,10,30},{.1,10,35},{3,10,30},{.1,10,35}]
c = [1,1.2,3.0,3.2],
p = [{0.3689,0.1170, 0.2673}, {0.4699,0.4387,
0.7470},{0.1091,0.8732,0.5547},{0.038150,
0.5743,0.8828}].

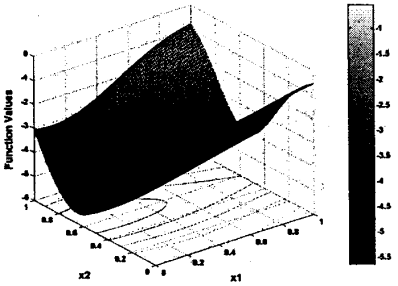
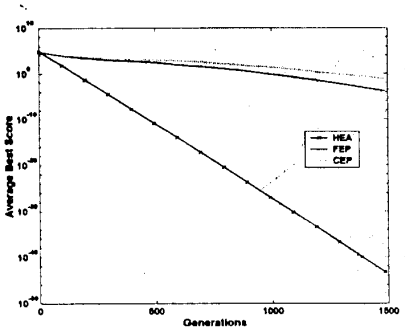
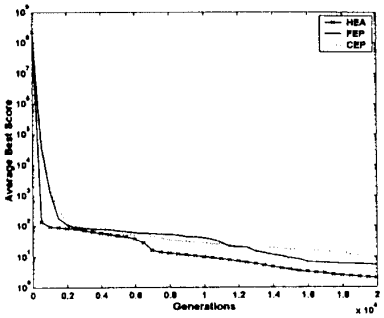


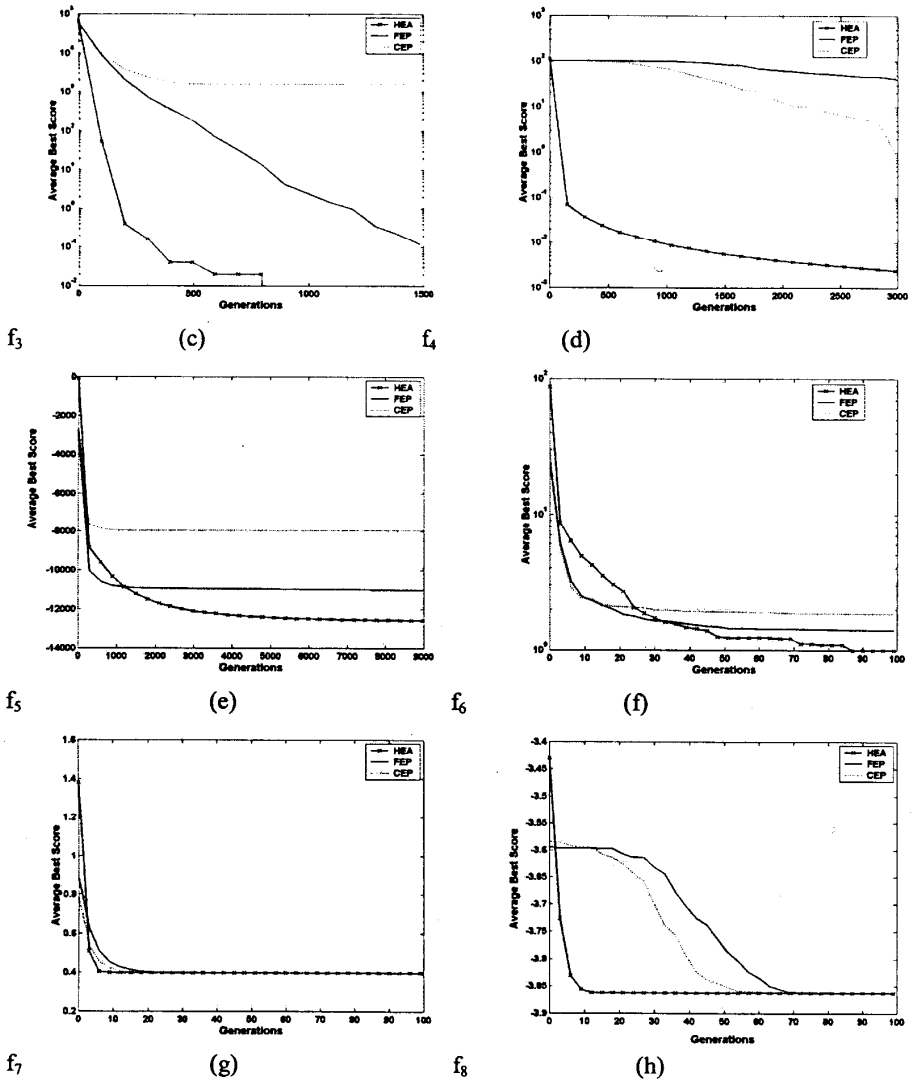
Fig. 1. (b) Last 3-benchmark functions, where n_o is the function dimension, $f_{\min}$ is the minimum function value and SD is the user supplied search domain.



f_1 (a)



f_2 (b)

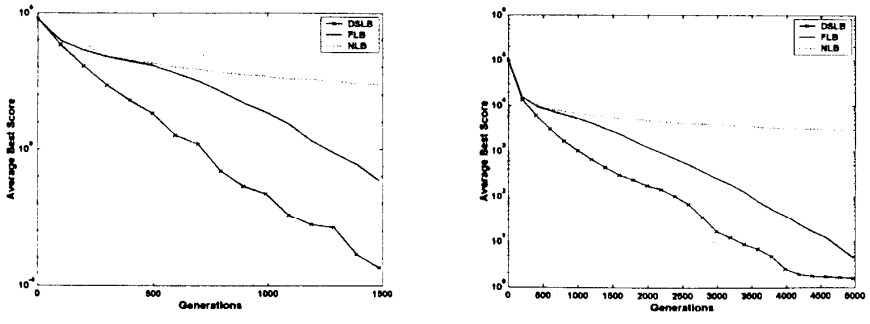


f_3 (c) f_4 (d)

f_5 (e) f_6 (f)

f_7 (g) f_8 (h)

Fig.2. Comparison of HEA, FEP and CEP on multimodal functions f_1 to f_8. The results are averaged over 50 runs.



f_1 (a) f_2 (b)

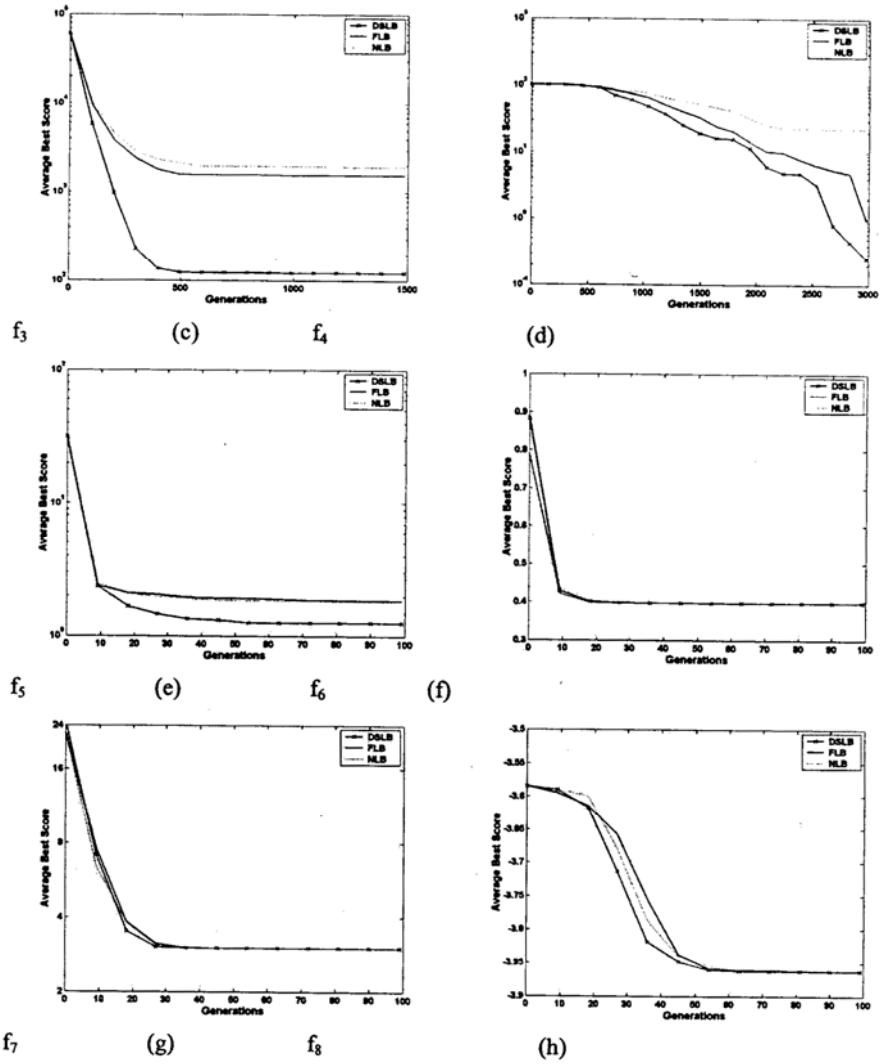


Fig.3. Comparison of CEPDSLB, CEPFLB and CEPNLB on functions f_1 to f_8. The results are averaged over 50 runs.

REFERENCES

[1] D.B. Fogel, *Evolutionary computation: towards a new philosophy of machine intelligence*, IEEE press, Piscataway, N.J., USA, 1995.

[2] I. Ono, and Kobayashi, "A real-coded genetic algorithm for function optimization using unimodal normal distribution crossover," *Proceedings of the Seventh International Conference on Genetic Algorithms*, 1997, pp.246-253.

[3] H.-P. Schwefel, *Numerical optimization of computer models*, John Wiley, 1981.

[4] H.-G. Beyer, "Toward a theory of evolution strategies: On the benefits of sex-the $(\mu/\mu, \lambda)$-theory," *Evolutionary Computation*. vol. 3, no. 1, 1995, pp.81-111.

[5] H.-G. Beyer, "Toward a theory of evolution strategies: The (μ, λ)-theory," *Evolutionary Computation*. vol. 3, no. 4, 1995, pp. 165-188.

[6] N. Hansen, and A. Ostermeier, "Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, 1996, pp.312-317.

[7] T. Bäck, *Evolutionary algorithms in theory and practice*, Oxford University Press, Inc., NY, 1996.

[8] T. Bäck, "Self-adaptation," *Handbook of Evolutionary Computation*, editors, Bäck, T., Fogel, D.B. and Michalewicz, Z., Oxford University Press, Inc., NY, 1997.

[9] N. Sarvanan, D.B. Fogel, and K.M. Nelson, A comparison of methods for self-adaptation in evolutionary algorithms. *BioSystems*, vol. 36, 1995, pp.157-166.

[10] D.B. Fogel, *Evolutionary computation: towards a new philosophy of machine intelligence*, 2nd ed, IEEE press, 2000.

[11] T. Bäck, "The interaction of mutation rate, selection rate, and self-adaptation within a genetic algorithm," in *Parallel Problem Solving from Nature II*, Männer, R. and Manderick, B. Eds., Amsterdam: North Holland, 1992, pp.85-94.

[12] T. Bäck, "Self-adaptation in genetic algorithms," in *Toward a practice of Autonomous Systems: Proceedings First European Conference on Artificial Life*, Varela, F.J. and Bourgine, P., Eds., Cambridge, MA: MIT Press, 1992, pp.263-271.

[13] M. Srinivas, and L.M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," *IEEE Transactions on System, Man and Cybernetics*, vol. 24, no. 4, 1994, pp.17-26.

[14] R. Hinterding, "Gaussian mutation and self-adaptation in numeric genetic algorithms," in *Proceedings Second IEEE Conference on Evolutionary Computation*, Piscataway, NJ: IEEE Press, 1995, pp.384-389.

[15] J. Smith, and T.C. Fogarty, "Self-adaptation of mutation rates in a steady state genetic algorithm," in *Proceedings Third IEEE Conference on Evolutionary Computation*, Piscataway, NJ: IEEE Press, 1996, pp.318-323.

[16] K. Deb, and H.-G. Beyer, "Self-adaptation in real-parameter genetic algorithms with Simulated binary crossover," in *GECCO-1999: Proceedings of the Genetic and Evolutionary Computation Conference*, Orlando, Florida, 1999, pp.172-179.

[17] M. Herdy, "Reproductive isolation as strategy parameter in hierarchically organized evolutionary strategies," in *Parallel Problem Solving from Nature II*, Männer, R. and Manderick, B. Eds., Amsterdam: North Holland, 1992, pp.207-217.

[18] N. Hansen, and A. Ostermeier, "Convergence properties in evolution strategies with the derandomized covariance matrix adaptation: The (μ / μ_1, λ)-CMA-ES," in *Proceedings of European Congress on Intelligent Techniques and Soft Computing*, 1997, pp.650-654.

[19] T. Bäck, *Evolutionary algorithms in theory and practice*, Oxford University Press, Inc., NY, 1996.

[20] B. Bäck, and H,-P. Schwefel, "Evolutionary computation: an overview," in Proceeding of Third IEEE Conference on Evolutionary Computation, Piscataway, NJ: IEEE, 1996, pp.20-29.

[21] B. Bäck, and H,-P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary Computation*. vol. 1, no. 1, 1993, pp. 1-23.

[22] X. Yao, and Y. Liu, "Fast evolutionary programming," in *Evolutionary Programming: Proc. Fifth Annual Conf. Evolutionary Programming*, edited by L.J. Fogel, P.J. Angeline, and T. Bäck, Cambridge MA: MIT Press, 1996, pp. 451-460.

[23] X. Yao, and Y. Liu, " Fast evolution strategies," *Control and Cybernatics*, vol. 26, no. 3, 1997, pp.467-496.

[24] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Trans.Evolutionary Computation*. vol. 3, no. 2, 1999, pp. 82-102.

[25] N.L. Johnson, S. Kotz, and N. Balakrishnan, *Continuous univariate distributions. Vol. 1*. John Wiley & sons, Inc., USA, 1994.

[26] N. Sarvanan, and D.B. Fogel, "Multi-operator evolutionary programming: A preliminary study," in *Proc. Sixth Intel. Conf. Evolutionary Programming*, Spinger, 1997, pp. 215-221.

[27] K. Chellapilla, "Combining mutation operators in evolutionary programming," *IEEE Transaction on Evolutionary Computation*, vol. 2, no. 3, 1998, pp. 91-96.

[28] G. Rudolph, "Local convergence rates of simple evolutionary algorithms with Cauchy mutations," *IEEE Trans. Evolutionary Computation*. vol. 1, no. 4, 1997, pp. 249-258.

[29] A.K. Swain, and A.S. Morris, "A novel hybrid evolutionary programming method for function optimization," *Proc. Congress on Evolutionary Computation (CEC2000)- IEEE/IEE Intl. Conf. on Evolutionary Computation*, San Diego, USA, July, 2000.

[30] K.-H Liang, X. Yao, C. Newton, and D. Hoffman, An experimental investigation of self-adaptation in evolutionary programming. in *Proc. Seventh Intel. Conf. Evolutionary Programming*, Spinger, 1998, pp. 291-300.

[31] K.-H Liang, X. Yao, and C. Newton, Dynamic control of adaptive parameters in evolutionary programming. *Proc. SEAL98*, Spinger-Verlag, 1998, pp.42-49.

[32] M.R. Glickman, and K. Sycara, "Reasons for premature convergence of self-adapting mutation rates," in *Proceedings of Congress on Evolutionary Computation (CEC-2000)*, San Diego, CA, 2000.

[33] A.K. Swain, and A.S. Morris, "A hybrid evolutionary algorithm for multimodal function optimization," *Proc. Genetic and Evolutionary Computation Conference (GEECO-2000) Late-breaking Papers*, July 8-12, 2000, Las Vegas, Nevada.

Indian Institute of Management Kozhikode

Type of Document (Working Paper/Case/ Teaching Note, etc.)	*Ref. No.:*
Working Paper	**IIMK/WPS/13/IT/2007/02**

Title:

Performance Analysis of Self-Adaptive Evolutionary Computation Methods

Author(s)& Contact Details :	*Institution(s):*
Anjan Kumar Swain, Associate Professor, Indian Institute of Management Kozhikode IIMK Campus PO-673570, Kozhikode, Kerala, India Ph: (91-495) 2809122, email:akswain@iimk.ac.in	Indian Institute of Management Kozhikode, Kozhikode, Kerala, India

Subject Areas: **Information Technology**	*Subject Classification Codes, if any:*
Supporting Agencies, if any:	*Research Grant/Project No.(s):*
Supplementary Information, if any:	*Date of Issue:* January 2007
	Number of Pages: 19

Abstract:

This paper concerns with the detailed analysis of the performance of self-adaptive evolutionary computation algorithms. Various causes of premature convergence in these methods have been established. Subsequently, formulation of two new evolutionary algorithms has been discussed. The potentiality of these methods has been verified on eight popular test functions.

Key Words/Phrases: Evolutionary Computation; Self-adaptive; Fast Evolutionary Computation; Dynamic Lower Bound

Referencing Style Followed: Journal of Information and Management, Elsevier B.V.