Self-Adaptive Genetic Algorithms with Simulated Binary Crossover

Kalyanmoy Deb deb@iitk.ac.in
Kanpur Genetic Algorithms Laboratory, Department of Mechanical Engineering, Indian Institute of Technology Kanpur, PIN 208 016, India

Hans-Georg Beyer beyer@zappa.cs.uni-dortmund.de
Systems Analysis Group, Department of Computer Science XI, University of Dortmund, D-44221 Dortmund, Germany

Abstract

Self-adaptation is an essential feature of natural evolution. However, in the context of function optimization, self-adaptation features of evolutionary search algorithms have been explored mainly with evolution strategy (ES) and evolutionary programming (EP). In this paper, we demonstrate the self-adaptive feature of real-parameter genetic algorithms (GAs) using a simulated binary crossover (SBX) operator and without any mutation operator. The connection between the working of self-adaptive ESs and real-parameter GAs with the SBX operator is also discussed. Thereafter, the self-adaptive behavior of real-parameter GAs is demonstrated on a number of test problems commonly used in the ES literature. The remarkable similarity in the working principle of real-parameter GAs and self-adaptive ESs shown in this study suggests the need for emphasizing further studies on self-adaptive GAs.

Keywords

Self-adaptation, genetic algorithms, simulated binary crossover, blend crossover, real-coded GAs, evolution strategies.

1 Introduction

Self-adaptation is a phenomenon that makes evolutionary algorithms flexible and closer to natural evolution. Among the evolutionary methods, self-adaptation properties have been explored with *evolution strategies* (ESs) (Bäck, 1997; Beyer, 1995b; Hansen and Ostermeier, 1996, 1997; Rechenberg, 1973; Saravanan et al., 1995; Schwefel, 1987, 1988) and *evolutionary programming* (EP) (Fogel et al., 1995), although there exist some studies of self-adaptation in genetic algorithms (GAs) with a mutation operator (Bäck, 1992). Despite such studies, there exists no formal definition of self-adaptation or description of properties an algorithm should have in order for it to qualify to be a self-adaptive algorithm. In this paper, we do not try to answer this question, rather recognize the importance of such a study in the near future. When applied to function optimization, there are a number of reasons why evolutionary algorithmists should pay attention to self-adaptation:

1. Knowledge of lower and upper bounds for the optimal solution may not be known *a priori*,

2. It may be desired to know the optimal solution with arbitrary precision,

3. The objective function and the optimal solution may change with time.

In many problems, the lower and upper bounds for the optimal solution may not be known a priori. Some evolutionary search algorithms such as binary-coded GAs require information about lower and upper bounds on each problem variable, so that a linear mapping of the decoded values of a binary string-coding can be used. This forces the search to concentrate only within the chosen lower and upper bounds. If the true optimal solution does not lie within this range, the fixed coding scheme of binary-coded GAs will not be able to find the true optimal solution. In this respect, ES and EP are alternatives where precise knowledge of such range is not required. Real-parameter GAs, where problem variables are not coded in any string structure, rather are used directly, can eliminate the rigidity of fixed coding used in binary-coded GAs.

Some search and optimization problems require the optimal solution to be found with arbitrary precision. In such cases, the fixed-length coding scheme has another disadvantage. Since a fixed-length coding is used, the algorithm offers a lower bound on the precision that can be achieved in a GA. For example, if ℓ_i bits are used to code a problem variable in the range $[x_i^l, x_i^u]$ in a binary-coded GA, the maximum attainable precision is $(x_i^u - x_i^l)/(2^{\ell_i} - 1)$. Although the precision in the optimal solution can be increased by increasing the string length ℓ_i, it has been shown elsewhere (Goldberg et al., 1992) that even for simple problems the required population size is of the order of the string length. One other approach to achieve more precision is to use a variable-length coding or a coarse-to fine-grained coding, both of which make the algorithm more complex and subjective to the way decisions are made in switching from coarse-to fine-grained coding (Shaefer, 1987; Schraudolph and Belew, 1990). Once again, real-parameter GAs with direct use of problem variables can practically achieve any precision in the problem variables, simply because the real numbers are used directly.

One of the challenging optimization problems, and more often found in real-world search and optimization, is a problem with an objective function that changes with time. When such problems are to be solved for optimality, the search procedure needs to be flexible enough to adapt to the new function landscape as quickly as it changes. A difficulty of the population-based optimizers is that once the search has narrowed near the previous optimal solution, the diversity in the population may not be enough for the search to get out of there and proceed towards the new optimal solution. Often, in these cases, diversity preserving mechanisms (large mutation rate, inclusion of a niching operator, and others) must be used. However, the maintenance of diversity does not come free; a portion of the total function evaluations is always spent in areas of non-interest in the current iteration as an investment for diversity needed at a later iteration when the function landscape changes. ES or EP without self-adaptation cannot tackle such problems and is not flexible enough to respond to changing landscapes. However, the inclusion of self-adaptive features in both ES and EP allowed such problems to be solved with an addition of extra strategy parameters that control the degree of search power in their major mutation-based search operators. Although not obvious, such self-adaptive behavior is also possible to achieve with real-parameter GAs with specialized crossover operators.

In this paper, we show the self-adaptive behavior of real-parameter GAs with one such crossover operator on a number of different fitness landscapes commonly used in self-adaptive ES studies. The *simulated binary crossover operator* (SBX) (Deb and Agrawal, 1995) uses a probability distribution around two parents to create two children solutions. Unlike other real-parameter crossover operators, SBX uses a probability distribution that is similar in principle to the probability of creating children solutions in crossover operators used in binary-coded GAs. There are two aspects that give real-parameter GAs with SBX their self-adaptive power: (i) children solutions closer to parent solutions are more likely to be created, and (ii) the span of children solutions is proportional to the span of parent solutions.

In the remainder of the paper, we briefly discuss the working principle of the SBX operator. Thereafter, we mention different variants of self-adaptive ESs and show that there is a similarity in the working of real-parameter GAs with SBX and at least one variant of self-adaptive ESs. The self-adaptive behavior of real-parameter GAs with the SBX operator is then demonstrated by applying them to a number of test problems. Finally, based on the current study, a number of plausible extensions are suggested.

2 Real-Parameter Genetic Algorithms and Simulated Binary Crossover (SBX)

There exists a number of real-parameter GA implementations, where crossover and mutation operators are applied directly to real parameter values. One of the early implementations was by Wright (1991), where a linear crossover operator created three solutions $(x_i^{(1,t)}+x_i^{(2,t)})$, $(1.5x_i^{(1,t)}-0.5x_i^{(2,t)})$, and $(-0.5x_i^{(1,t)}+1.5x_i^{(2,t)})$ from two parent solutions $x_i^{(1,t)}$ and $x_i^{(2,t)}$ at generation t and chose the best two solutions as children solutions. Goldberg (1991) introduced the concept of virtual alphabets in the context of real-coded GAs. Eshelman and Schaffer (1993) have introduced the notion of *interval* schemata for real-coded genetic algorithms and suggested a *blend crossover operator* (BLX-α). For two parent solutions $x_i^{(1,t)}$ and $x_i^{(2,t)}$ (assuming $x_i^{(1,t)} < x_i^{(2,t)}$), the BLX-$\alpha$ randomly picks a solution in the range $\{x_i^{(1,t)}-\alpha(x_i^{(2,t)}-x_i^{(1,t)}), x_i^{(2,t)}+\alpha(x_i^{(2,t)}-x_i^{(1,t)})\}$. Thus, if u_i is a random number between 0 and 1, the following is a child solution:

$$x_i^{(1,t+1)} = (1-\gamma_i)x_i^{(1,t)} + \gamma_i x_i^{(2,t)}, \tag{1}$$

where $\gamma_i = (1+2\alpha)u_i - \alpha$. If α is zero, this crossover creates a random solution in the range $(x_i^{(1,t)}, x_i^{(2,t)})$. In a number of test problems, they have reported that BLX-0.5 (with $\alpha = 0.5$) performs better than BLX operators with any other α value. However, it is important to note that the factor γ_i is uniformly distributed for a fixed value of α. However, BLX-α has an interesting property: the location of the child solution depends on the difference in parent solutions. This will be clear if we rewrite Equation 1 as follows:

$$\left(x_i^{(1,t+1)} - x_i^{(1,t)}\right) = \gamma_i\left(x_i^{(2,t)} - x_i^{(1,t)}\right). \tag{2}$$

If the difference between the parent solutions is small, the difference between the child and parent solutions is also small. We believe that this is an essential property for any search algorithm to exhibit self-adaptation. This is because the spread of the current population dictates the spread of the solutions in the resulting population. In problems where self-adaptation should cause the population to either converge or diverge for better regions in the search space, assignment of children population proportional to parent population may help achieve the task.

Ono and Kobayashi (1997) suggested a *unimodal normally distributed crossover operator* (UNDX), where three parent solutions are used to create two or more children solutions. Children solutions are created from an ellipsoidal probability distribution with one axis formed along the line joining two of the three parent solutions, and the extent of the orthogonal direction is decided by the perpendicular distance of the third

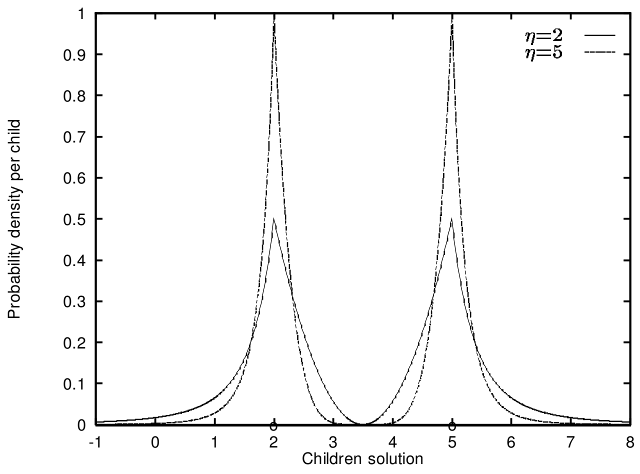


Figure 1: Probability distribution for creating children solutions of continuous variables. Parents are marked with an "o."

parent from the axis. Unlike the BLX operator, this operator assigns more probability for creating solutions near the center of the first two parents than near the parents. Recently, a multi-parental UNDX operator with more than three parents is also suggested (Kita et al., 1999). Like the BLX operator, this operator also assigns children solutions proportional to the spread of parent solutions, thereby giving a GA with this operator the potential to exhibit self-adaptation.

In 1995, the first author and his students developed the simulated binary crossover (SBX), which works with two parent solutions and creates two children solutions (Deb and Agrawal, 1995; Deb and Kumar, 1995). As the name suggests, the SBX operator simulates the working principle of the single-point crossover operator on binary strings. In those studies, authors showed that this crossover operator respects the interval schemata processing, in the sense that common interval schemata between parents are preserved in children. The procedure of computing the children solutions $x_i^{(1,t+1)}$ and $x_i^{(2,t+1)}$ from parent solutions $x_i^{(1,t)}$ and $x_i^{(2,t)}$ is described as follows. A spread factor β_i is defined as the ratio of the absolute difference in children values to that of the parent values:

$$\beta_i = \left| \frac{x_i^{2,t+1} - x_i^{1,t+1}}{x_i^{2,t} - x_i^{1,t}} \right|. \tag{3}$$

First, a random number u_i between 0 and 1 is created. Thereafter, from a specified probability distribution function, the ordinate β_{q_i} is found so that the area under the probability curve from 0 to β_{q_i} is equal to the chosen random number u_i. The probability distribution used to create a child solution is derived to have a similar search power as that in a single-point crossover in binary-coded GAs and is given as follows (Deb and Agrawal, 1995):

$$\mathcal{P}(\beta_i) = \begin{cases} 0.5(\eta+1)\beta_i^{\eta}, & \text{if } \beta_i \leq 1; \\ 0.5(\eta+1)\frac{1}{\beta_i^{\eta+2}}, & \text{otherwise.} \end{cases} \tag{4}$$

Figure 1 shows the above probability distribution with $\eta = 2$ and 5 for creating children solutions from two parent solutions ($x_i^{(1,t)} = 2.0$ and $x_i^{(2,t)} = 5.0$) in the real space. In the above expressions, the distribution index η is any nonnegative real number. A large value of η gives a higher probability for creating near parent solutions, and a small value of η allows distant solutions to be selected as children solutions. Using Equation 4, we calculate β_{q_i} by equating the area under the probability curve equal to u_i as follows:

$$\beta_{q_i} = \begin{cases} (2u_i)^{\frac{1}{\eta+1}}, & \text{if } u_i \leq 0.5; \\ \left(\frac{1}{2(1-u_i)}\right)^{\frac{1}{\eta+1}}, & \text{otherwise.} \end{cases} \tag{5}$$

After obtaining β_{q_i} from the above probability distribution, the children solutions are calculated as follows:

$$x_i^{(1,t+1)} = 0.5\left[(1+\beta_{q_i})x_i^{(1,t)} + (1-\beta_{q_i})x_i^{(2,t)}\right], \tag{6}$$

$$x_i^{(2,t+1)} = 0.5\left[(1-\beta_{q_i})x_i^{(1,t)} + (1+\beta_{q_i})x_i^{(2,t)}\right]. \tag{7}$$

Thus, the following step-by-step procedure is followed to create two children solutions ($x_i^{(1,t+1)}$ and $x_i^{(2,t+1)}$) from two parent solutions ($x_i^{(1,t)}$ and $x_i^{(2,t)}$):

Step 1: Choose a random number $u_i \in [0, 1)$.

Step 2: Calculate β_{q_i} using Equation 5.

Step 3: Compute children solutions using Equations 6 and 7.

Note that two children solutions are symmetric about the parent solutions. This is deliberately used to avoid any bias towards any particular parent solution in a single crossover operation. Another interesting aspect of this crossover operator is that for a fixed η, the spread of children solutions is proportional to that of the parent solutions:

$$\left(x_i^{(2,t+1)} - x_i^{(1,t+1)}\right) = \beta_{q_i}\left(x_i^{(2,t)} - x_i^{(1,t)}\right). \tag{8}$$

This has an important implication. Let us consider two scenarios: (i) two parents are far away from each other, and (ii) two parents are closer to each other. For illustration, both these cases (with parent solutions $x_i^{(1,t)} = 2.0$ and $x_i^{(2,t)} = 5.0$ in the first case and with parent solutions $x_i^{(1,t)} = 2.0$ and $x_i^{(2,t)} = 2.5$ in the second case) and the corresponding probability distributions with $\eta = 2$ are shown in Figures 2 and 3, respectively. For an identical random number u_i, the parameter β_{q_i} takes the same value in both cases. From Equation 8 it is clear that in the first case the children are likely to be more widely spread than in the second case. Figures 2 and 3 also show the corresponding children solutions (marked with a box) for $u_i = 0.8$ or $\beta_{q_i} = 2.5^{1/3}$. Figures clearly show that if the parent values are far from each other (the first case), solutions away from parents can be created. Compare the child solution $x_i^{(1,t+1)} = 1.464$ with parent $x_i^{(1,t)} = 2.0$. But if the parent values are close by (the second case), distant children solutions are not likely. Compare the child solution $x_i^{(1,t+1)} = 1.911$ to parent $x_i^{(1,t)} = 2.0$ that creates using the same random number as in the first case. In initial populations, where the solutions are randomly placed (like the first case), this allows almost any value to be created as a child solution. But when the solutions tend to converge due to the action of genetic operators (like the second case), distant solutions are not allowed, thereby focusing the search to a narrow region.

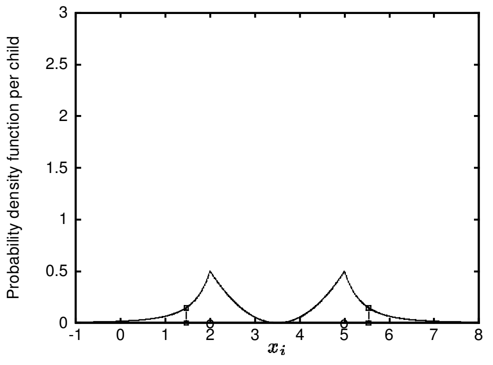


Figure 2: Probability distribution of children solutions with distant parents.

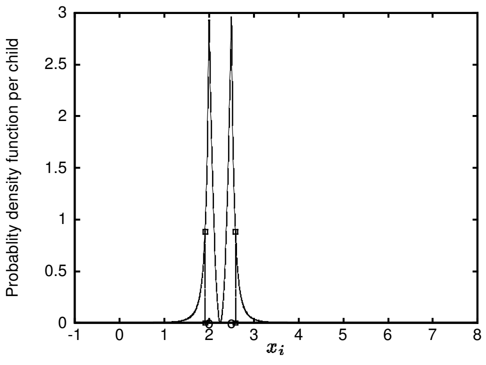


Figure 3: Probability distribution of children solutions with closely spaced parents.

It is interesting to note that both Equations 6 and 7 can be written in the form of Equation 1 with the following relationship: $\gamma_i = 0.5(1 \mp \beta_{q_i})$. However, it is important that, unlike in the BLX-α operator, the equivalent γ_i term in the SBX operator is not uniformly distributed (refer to Equation 5). The SBX operator biases solutions near each parent more favorably than solutions away from parents. Essentially, the SBX operator has two properties:

1. The extent of children solutions is in proportion to the parent solutions, and
2. Near parent solutions are monotonically more likely to be chosen as children solutions than solutions distant from parents.

3 Self-Adaptive Evolution Strategies

There are three different ways self-adaptation is used in ES: (i) a hierarchically organized population-based meta-ES (Herdy, 1992), (ii) adaptation of the *covariance matrix* (CMA) determining the probability distribution for mutation (Hansen and Ostermeier, 1996), and (iii) explicit use of self-adaptive control parameters (Rechenberg, 1973; Schwefel, 1987). The meta-ES method of self-adaptation uses two levels of ESs. The top level optimizes the strategy parameters (such as mutation strengths), a solution of which is used to optimize the true objective function in the lower level ES. The second method (CMA) records the population history for some number of iterations to calculate covariance and variance information among object variables. Subsequent search effort is influenced by these variance values. In this paper, we use the third type of self-adaptive ES, where the strategy parameters are explicitly coded and updated using the log-normal update rule in each generation. There are basically three different implementations that are in use.

3.1 Isotropic Self-Adaptation

In this self-adaptive ES, a single mutation strength σ is used for all variables. In addition to N object variables, the strategy parameter σ is also used in a population member. Here are the update rules:

$$\sigma^{(t+1)} = \sigma^{(t)} \exp(\tau_0 N(0,1)), \tag{9}$$

$$x_i^{(t+1)} = x_i^{(t)} + \sigma^{(t+1)} N_i(0,1), \tag{10}$$

where $N(0,1)$ and $N_i(0,1)$ are realizations of a one-dimensional normally distributed random variable with mean zero and standard deviation one. The parameter τ_0 is the learning parameter which must be set as $\tau_0 \propto N^{-1/2}$, where N is the dimension of the variable vector (Schwefel, 1987). Beyer (1996) has shown that, for the sphere model, the optimal learning parameter for (1,λ)-ES is $\tau_0 = c_{1,\lambda}/\sqrt{N}$, where $c_{1,\lambda}$ is the progress coefficient. We use $c_{\mu,\lambda}$ or $c_{\mu/\mu,\lambda}$ as constant of proportionality in corresponding ES, although they may not be optimal. The above update rule for σ requires an initial value. In all simulations here, we choose a $\sigma^{(0)} = (x_i^u - x_i^l)/\sqrt{12}$ which assumes a uniform distribution of solutions within the specified range of x_i values.

3.2 Non-Isotropic Self-Adaptation

Here, a different mutation strength σ_i is used for each variable. Thus this type of self-adaptive ES is capable of learning to adapt to problems where each variable has unequal contribution to the objective function. In addition to N object variables, N other strategy parameters. The update rules are as follows:

$$\begin{aligned} \sigma_i^{(t+1)} &= \sigma_i^{(t)} \exp\left(\tau' N(0,1) + \tau N_i(0,1)\right), & (11)\\ x_i^{(t+1)} &= x_i^{(t)} + \sigma_i^{(t+1)} N_i(0,1), & (12) \end{aligned}$$

where $\tau' \propto (2n)^{-1/2}$ and $\tau \propto (2n^{1/2})^{-1/2}$. Due to lack of any theoretical results on this self-adaptive ES, we use the progress coefficient of the (μ,λ)-ES or $(\mu/\mu,\lambda)$-ESs as the constant of proportionality of both τ' and τ. Similar initial values for $\sigma_i^{(0)}$ as discussed for isotropic self-adaptive ESs are used here.

3.3 Correlated Self-Adaptation

Here, different mutation strengths and rotation angles are used to represent the covariances for pair-wise interactions among variables. Thus, in addition to N object variables, there are a total of N mutation strengths and $N(N-1)/2$ rotation angles used explicitly in each population member. The update rules are as follows:

$$\begin{aligned} \sigma_i^{(t+1)} &= \sigma_i^{(t)} \exp\left(\tau' N(0,1) + \tau N_i(0,1)\right), & (13)\\ \alpha_j^{(t+1)} &= \alpha_j^{(t)} + \beta_\alpha N_j(0,1), & (14)\\ \vec{x}^{(t+1)} &= \vec{x}^{(t)} + \vec{N}\left(\vec{0}, C(\vec{\sigma}^{(t+1)}, \vec{\alpha}^{(t+1)})\right), & (15) \end{aligned}$$

where $\vec{N}\left(\vec{0}, C(\vec{\sigma}^{(t+1)}, \vec{\alpha}^{(t+1)})\right)$ is a realization of a normally distributed correlated mutation vector with a zero mean vector and covariance matrix C. The parameter β_α is fixed as 0.0873 (or 5^o) (Schwefel, 1987). The parameters τ', τ, and $\sigma_i^{(0)}$ are set as before. We initialize the rotation angles within zero and 180 degrees at random.

4 Connection Between GAs with SBX and Self-Adaptive ESs

Without loss of generality, we try to argue the similarity in the working of GAs with SBX and self-adaptive ESs by considering only isotropic self-adaptive ESs. Under isotropic self-adaptive ES, the difference (say Δ_i) between the child ($x_i^{(t+1)}$) and its parent ($x_i^{(t)}$) can be written from Equations 9 and 10:

$$\Delta_i = \left(\sigma^{(t)} \exp(\tau_0 N(0,1))\right) N(0,1). \qquad (16)$$

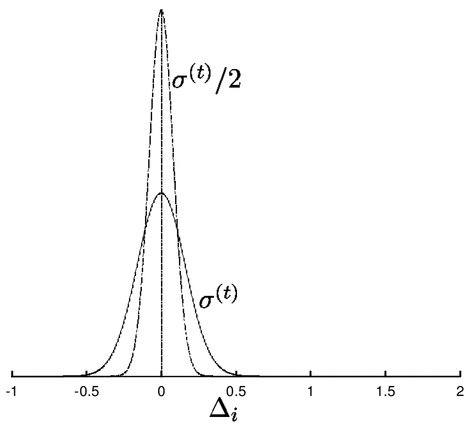


Figure 4: Effect of mutation strength $\sigma^{(t)}$ on the probability distribution of Δ_i under the self-adaptive ES.

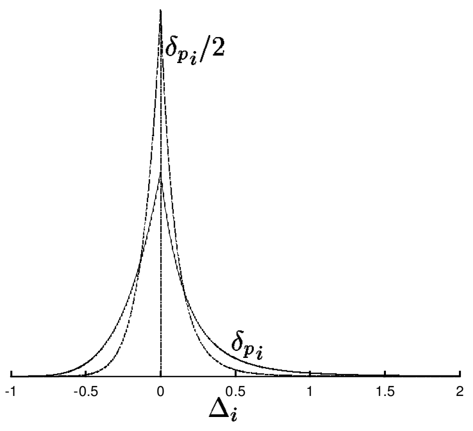


Figure 5: Effect of δ_{p_i} on the probability distribution of Δ_i under the SBX operator.

Thus, an instantiation of Δ_i is a normal distribution with zero mean and a variance that depends on $\sigma^{(t)}$, τ_0, and the instantiation of the the log-normal distribution. For our argument, there are two aspects of this procedure:

1. For a particular realization of log-normal distribution, the difference Δ_i is normally distributed with zero mean. That is, children solutions closer to parents are monotonically more likely to be created than children solutions away from parents.

2. The standard deviation of Δ_i is proportional to the mutation strength $\sigma^{(t)}$, which signifies, in some sense, the population diversity.

In Figure 4, we show the effect of 50% reduction in $\sigma^{(t)}$ on the distribution of Δ_i under a self-adaptive ES.

Under the SBX operator, we write the term Δ_i using Equations 6 or 7 as follows:

$$\Delta_i = \frac{\delta_{p_i}}{2}(\beta_{q_i} - 1), \tag{17}$$

where δ_{p_i} is the absolute difference in two parent solutions. There is a similarity between Equations 16 and 17. The above equation suggests that the distribution of Δ_i depends on the distribution of $(\beta_{q_i} - 1)$ for a particular pair of parents. The distribution of β_{q_i} has its mode at $\beta_{q_i} = 1$ (refer to Equation 5) thus, the distribution of $(\beta_{q_i} - 1)$ will have its mode at zero. Although we have not used a normal distribution for $(\beta_{q_i} - 1)$ here, Figure 2 or 3 suggests that a small Δ_i has a higher probability of creation than a large Δ_i and that this distribution is monotonic to the distance from a parent. The variance of this distribution depends on δ_{p_i}, which signifies the population diversity. Thus, there is a remarkable similarity in the way children solutions are assigned in both isotropic self-adaptive ES and in GAs with SBX. In both cases, the children solutions closer to parent solutions are assigned more probability of creation than solutions away from parents, and the variance of this probability distribution depends on the current population diversity. Figure 5 shows the effect of 50% reduction in δ_{p_i} on the distribution of Δ_i under the SBX operator. There is a remarkable similarity between the working of the two algorithms.

In a self-adaptive ES, the mutation strength gets continuously updated depending on the fitness landscape. For example, if the fitness landscape is such that the population needs to concentrate in a narrow region in the search space for improvement in the fitness, a self-adaptive ES usually evolves mutation strengths to become smaller and smaller, so that search concentrates near the parents rather than away from parents. The outcome of continuously reducing mutation strength is that most population members come closer and closer. When population members come closer in a real-parameter GA, the effective variance of probability distribution under the SBX operator also reduces. This, in turn, creates children solutions that are also not far away from each other. This helps to produce population members closer and closer to each other, thereby producing the effect of increased precision like that observed in the self-adaptive ES. A similar phenomenon occurs when a fitness function demands the population to diverge to get to the optimal region or demands other kind of variations in the search process.

5 Simulation Results

In this section, we present simulation results of real-parameter GAs with the SBX operator on a number of different test problems borrowed from the ES literature. For handling multi-variable problems, SBX is used variable-by-variable with a variable-wise probability of 0.5. We would like to mention here that since variable-wise crossover is used, GAs with the current implementation of SBX may face difficulty in problems where variable interactions are important. This so-called linkage issue is an important matter in search and optimization problems and has been recognized by many researchers (Goldberg et al., 1989; Harik, 1997; Kargupta, 1996; Schwefel, 1987). A tournament selection operator and no mutation operator is used in GAs with the SBX operator. In the context of self-adaptive ES, a (μ, λ)-ES signifies a non-recombinative ES and a $(\mu/\rho, \lambda)$-ES signifies a recombinative ES. Wherever recombinative ES is used, the dominant crossover is used on object variables and intermediate recombination is used on strategy parameters, as suggested by Bäck and Schwefel (1993). In all methods, no special effort is spent to find the best parameter settings; instead, a reasonable set of parameter values is used. In all test problems, we use $N = 30$ variables.

5.1 Sphere Model

This is the most commonly-used test function chosen for studying self-adaptation properties of ES. We consider several variants of this function in the following subsections.

5.1.1 Function F1-1: Quadratic Function

First, we consider the sphere model, where the objective is to minimize the following N-variable function:

$$\text{F1-1:} \quad \text{Minimize} \sum_{i=1}^{N} (x_i - x_i^*)^2, \tag{18}$$

where x_i^* is the optimal value of the ith variable. In the simulations presented first, we have chosen $x_i^* = 0.0$. Populations are initialized in $x_i \in [-1.0, 1.0]$. Real-parameter GAs with SBX are used to find the optimal solution for this function. Tournament size $s = 2$ and distribution index $\eta = 1$ for the SBX operator are used. A population size of 100 is used. The Euclidean distance $R = \sqrt{\sum_{i=1}^{N} x_i^2}$ of the best solution in a population from the minimum is plotted with generation number in Figure 6. The ordinate axis is drawn in logarithmic scale and the figure shows that real-parameter GAs with SBX (solid line) are able to maintain increased precision with generation number. Here,

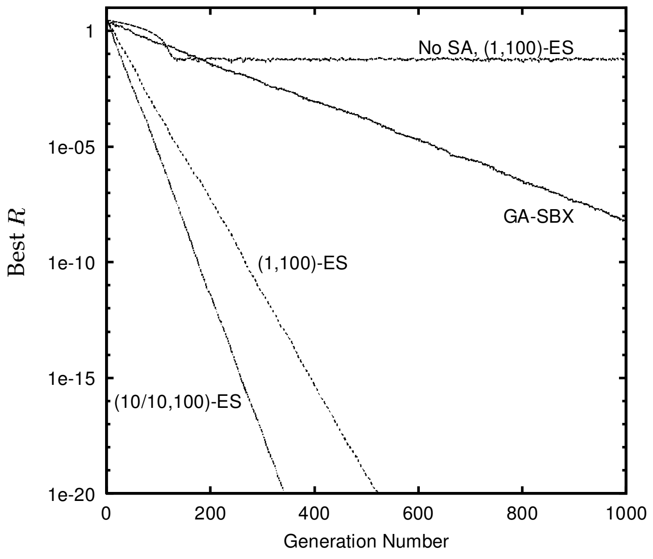


Figure 6: Population-best distance R from the minimum obtained with several evolutionary algorithms on test function F1-1.

it is worth mentioning that the straight lines refer to linear convergence, that is, an exponential decrease of the residual distance to the optimum. The figure also shows the performance of several other evolutionary algorithms. The performance of a $(1,100)$-ES with no self-adaptation (marked as No SA) is shown next. We have used a fixed mutation strength of 0.01 here. The children population size $\lambda = 100$ is used to keep the number of function evaluations the same as that in the real-parameter GAs. The figure re-confirms an already established (Beyer, 1995b) fact that a comma-ES without self-adaptation cannot find continuously increased precision. From a theoretical analysis on the sphere model, it was found that a non-self-adaptive ES gets stuck at a distance R_∞ given by (Beyer, 1995b):

$$R_\infty = \frac{\sigma N}{2c_{\mu,\lambda}}.$$

For (1,100)-ES, $c_{1,100} = 2.51$ and the above equation yields $R_\infty = 0.0598$. An average of population-best R values from generation 800 till 1000 generations is calculated from the simulation data, and it is found to be 0.0585, which is in good agreement (within 2%) with the theory.

Next, we apply isotropic self-adaptive ESs ((1,100)-ES and (10/10,100)-ES). In all self-adaptive ES runs, we have used $\tau = c_{\mu,\lambda}/\sqrt{N}$ to update the mutation strength. The figure also shows an already established fact that with proper learning parameter update, self-adaptive ESs can find continuously improved precision in the sphere model. The slope of the (1,100)-ES performance plot is 0.1011, which is about 3% from the theoretical estimate $(c_{1,\lambda}^2/(2N)$ (Beyer, 1995b).

There are two aspects to notice in Figure 6. First, the introduction of crossover enhances the performance of self-adaptive ES in this problem (Beyer, 1995a). Second, the performance of the self-adaptive ES is much better than that of real-parameter GAs with the SBX operator. This test function is unimodal, and an isotropic ES uses this

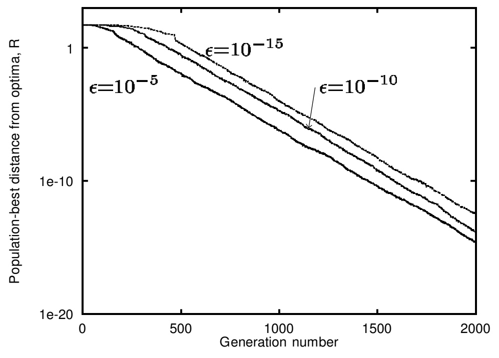


Figure 7: Population-best distance from optimum (R) for populations initialized at different ranges $[10-\epsilon, 10+\epsilon]$ for the function F1-2.

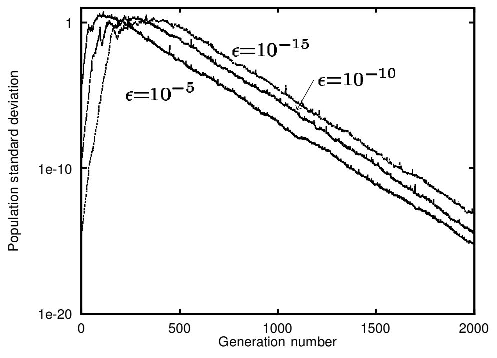


Figure 8: Population standard deviation with generation number for populations initialized at different ranges $[10-\epsilon, 10+\epsilon]$ for the function F1-2.

problem knowledge by using only one mutation strength parameter for all variables. On the other hand, a real-parameter GA with SBX does not use any such information, and hence, the progress rate comparison between the two algorithms is not proper. Moreover, there is a mismatch of effective selection pressure in both algorithms, with more selection pressure for the self-adaptive ESs. However, it is important to note here that real-parameter GAs with the SBX operator are able to maintain increased precision (linear convergence rate) with the generation number.

5.1.2 Function F1-2: Biased Population

Here, we use the same sphere model but initialize the population far away from the optimum and in a narrow range $x_i \in [10-\epsilon, 10+\epsilon]$, where ϵ is a small positive number. However, the minimum solution is at $x_i^* = 0.0$. The average distance from initial population from the minimum solution is thus $R_0 = 10\sqrt{N}$ (or, 54.772 with $N = 30$). We choose three different values of $\epsilon = 10^{-5}$, 10^{-10}, and 10^{-15}. Figures 7 and 8 show the population-best distance from optimum (R) and the population standard deviation[1] in the variable vectors (averaged over all $N = 30$ variables). Identical GA parameter settings as before are used here. With real-parameter GAs and the SBX operator, the initial population begins with a standard deviation of the order of ϵ and grows to a large value. Thereafter, the population standard deviation reduces as in test function F1-1, and GAs converge to the true optimum. Notice how GAs require a larger number of generations to bring the population standard deviation to a reasonable limit with a smaller ϵ value. This behavior of GAs is very similar to that observed with self-adaptive ESs (Bäck, 1997). Once the population has the correct population variance and it is near the optimum, the rate of convergence to the optimum with increasing precision is independent of how the population was initialized.

[1]This quantity is calculated by first finding the mean $\bar{x}$ of all population members. Thereafter, the standard deviation is computed as $\sqrt{\sum_{j=1}^{n}(x^{(j)}-\bar{x})^T(x^{(j)}-\bar{x})/(n-1)}$, where n is the population size and $x^{(j)}$ is the x-vector of the jth population member.

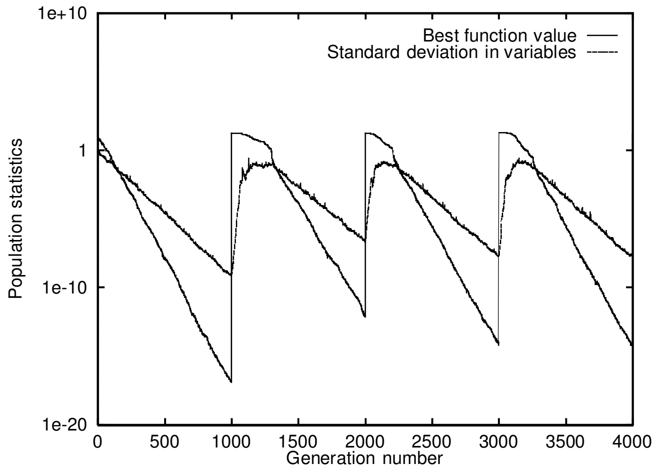


Figure 9: Population-best function value and average of population standard deviation in variables are shown with generation number, as test function F1-3 changes its optimum after every 1,000 generations.

5.1.3 Function F1-3: Time-Varying Function

In order to investigate the performance of real-parameter GAs with SBX on time-varying functions, we now choose the same function as F1-1, but x_i^* now varies with generation number in the range $[-1.0, 1.0]$ at random. The optimum is changed after every 1,000 generations so that at the time of a change in the optimum the population diversity has reduced substantially. The best function value and average population standard deviation (as defined earlier) in all variables are plotted versus generation number in Figure 9. GA parameter settings the same as that in F1-1 are used here. The figure shows that even though all population members are all within a small range (in the order of 10^{-10}) at the end of 999 generations, the population with the SBX operator can diverge and can get adapted to a changed optimum. This happens not only once, but as many times as there is a change in the function.

When the objective function changes (at generations 1,000, 2,000, and so on), the population standard deviation in all variables is of the order of 10^{-8}, suggesting that the population has lost diversity with respect to the new function. Even then, GAs with the SBX operator can quickly increase their population diversity and converge to the new optimum. First, the population gets more diverse to come near the current optimum and then decreases diversity to converge closer and closer to the current optimum. This is exactly how a self-adaptive ES is expected to behave in a time-varying function (Bäck, 1997), and we find here similar self-adaptive behavior of real-parameter GAs with the SBX operator.

5.1.4 Function F1-4: Multi-Modal Function

We now choose one test problem that is multi-modal in the search space. Moreover, in the range $[-1.0, 1.0]$, where the population is initialized, the function is unimodal, but just outside this range the function has other local attractors. We simply add a periodic

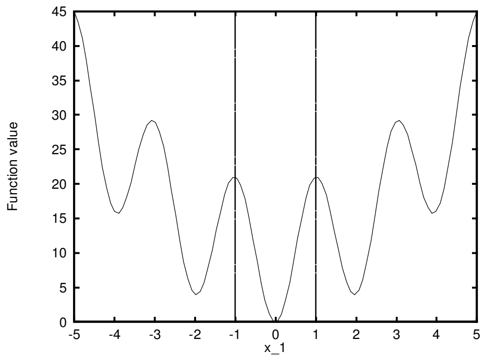


Figure 10: An one-dimensional version of F1-4.

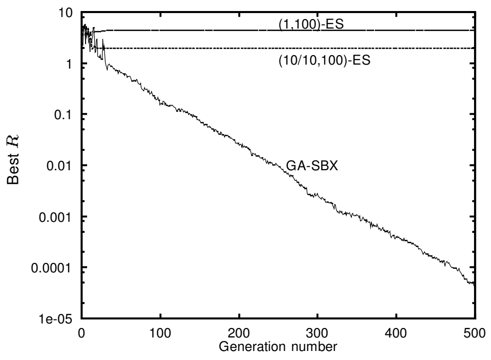


Figure 11: Population-best distance R from optimum for F1-4.

term to the sphere model:

$$\text{F1-4:} \quad \text{Minimize} \sum_{i=1}^{N} \left(x_i^2 + 10(1 - \cos(\pi x_i)) \right). \tag{19}$$

The interesting aspect is that the function has a global optimum (all $x_i = 0$) in the range where the population is initialized. But beyond this range, there exists a number of local optima – the nearest one for each variable is at $x_i = -2$ and $x_i = 2$. Figure 10 shows a one-dimensional version of this function. The range where the population is initialized is shown by drawing two vertical dashed lines. Figure 11 shows the performance of real-parameter GAs with SBX. The figure also shows the performance of isotropic self-adaptive (1,100)-ES and (10/10,100)-ES with identical parameter settings as used in Function F1-1. Now, self-adaptive ES seems unable to converge to the global attractor, although the initial population was placed in the global basin. In self-adaptive ESs, the mutation strength for each parameter needs an adaptation time within which the update of mutation strengths and corresponding fitness landscape should make a suitable agreement. If this agreement does not happen, either due to improper use of learning rate or other ES parameters, or due to a complex fitness, the mutation strength does not get adapted properly. Since in this function, the landscape just outside $[-1, 1]$ has a non-agreeing landscape compared to that inside the region $[-1, 1]$ for each variable, self-adaptive ES gets confused whether to increase or decrease mutation strengths for variables.

However, as suggested in Beyer (1995b, 335), if lower and upper bounds on variables are known with confidence, self-adaptive ES may be used with a small initial mutation strength $\sigma^{(0)}$. It is intuitive that when a small initial mutation strength $\sigma^{(0)}$ is used, the mutated solutions are not likely to be outside $[-1, 1]$, and thus self-adaptive ES will be confined to the global basin. When an initial mutation strength $\sigma^{(0)}$ is used (one-tenth of that used in the above runs), the ES converges to the correct optimum. However, a small initial mutation strength may not have the desired properties in other functions where divergence from the initial population is necessary to get to the true optimum (such as test function F1-2, F1-3, or ridge functions). Nevertheless, the study of this multi-modal test function suggests the importance of the initial mutation strength in successful working of self-adaptive ES, particularly in multi-modal problems, a matter that is not adequately addressed in the ES literature.

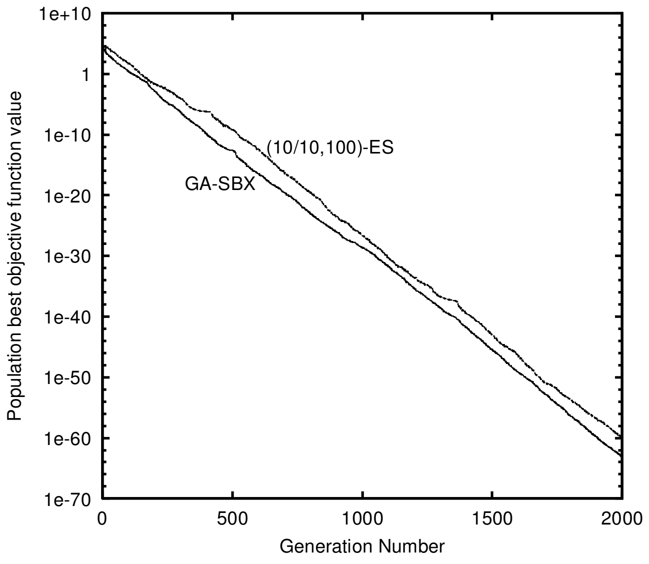


Figure 12: The best objective function value in the population is shown for two evolutionary algorithms: real-parameter GAs with SBX and self-adaptive (10/10,100)-ES on function F2-1.

In real-parameter GAs with SBX, there is an equal probability of creating solutions inside or outside the region bounded by two parent solutions. Thus, many children solutions will be created outside $[-1,1]$. But since the generation of children solutions is mainly governed by the distance between parent solutions, and the above function has an overall quadratic structure with its attractor at the global optimum, real-parameter GAs do not face much difficulty in converging to the true optimum. Notice how similar the performance of GAs with SBX in this figure is with respect to that in Figure 6, where the simple sphere model was considered.

5.2 Elliptic Model

In this function, every variable has an unequal contribution to the objective function. We consider a few variants of the elliptic function.

5.2.1 Function F2-1: Elliptic Function

It is similar to the sphere model, but not every variable has equal contribution to the objective function:

$$\text{F2-1:}\quad \text{Minimize} \sum_{i=1}^{N} 1.5^{i-1} x_i^2. \tag{20}$$

Since each variable has unequal contribution to the objective function, self-adaptive ESs with isotropic mutation strength is not adequate to solve this problem. Figure 12 shows the objective function value of the best solution in the population with a GA and with a (10/10,100)-ES. The population is initialized in $x_i \in [-1.0, 1.0]$. Once again, we use the same GA parameters as before but use tournament size 3 to compare the performance with self-adaptive ES.

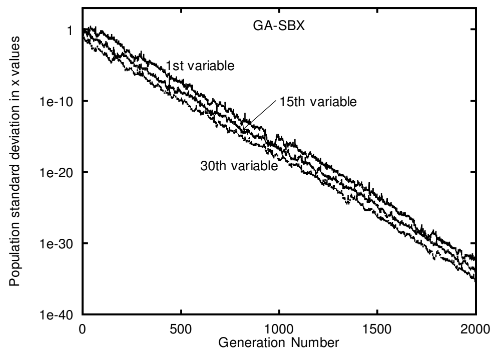


Figure 13: Population standard deviation in variables x_1, x_{15}, and x_{30} are shown with real-parameter GAs with the SBX operator for the function F2-1.

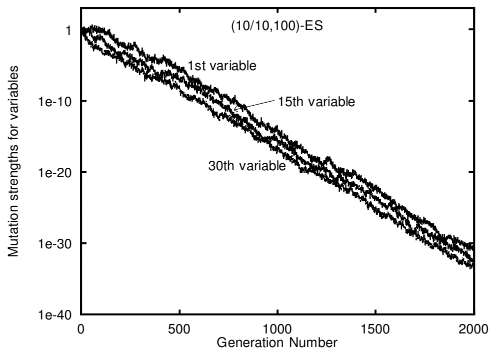


Figure 14: Mutation strengths for variables x_1, x_{15}, and x_{30} are shown with self-adaptive (10/10,100)-ES for the function F2-1.

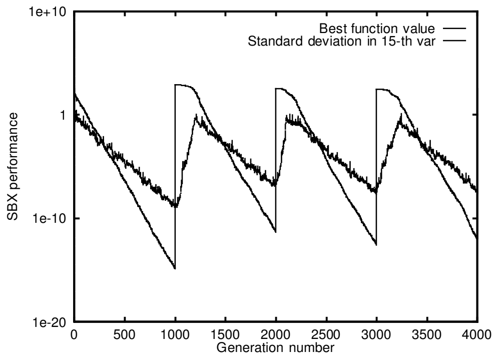
Figure 13 plots the population standard deviation in x_1, x_{15}, and x_{30} variables in the population for the real-parameter GAs with the SBX operator. Since they are scaled as 1.0, $1.5^{14} \approx 292$, and $1.5^{29} \approx 127,834$, respectively, the 30th variable is likely to have smaller population variance than the 1st variable. The figure shows this fact clearly. Since, ideal mutation strengths for these variables are also likely to be inversely proportionate to 1.5^{i-1}, we find similar ordering with non-isotropic self-adaptive ES as well (Figure 14). Thus, there is a remarkable similarity in how both the both real-parameter GAs with SBX and self-adaptive ES work. In the former case, the population diversity gets adapted based on the need of the fitness landscape, which in turn helps the SBX operator to create x_i values of children solutions proportionately. In the latter case, the population diversity gets controlled by independent mutation strengths, which get adapted based on the fitness landscape.

5.2.2 Function F2-2: Time Varying Elliptic Function

Like in the sphere model, we construct a test problem where the elliptic function changes its optimum solution occasionally with generation. We use the following function:

$$\text{F2-2:} \quad \text{Minimize} \sum_{i=1}^{N} r_i (x_i - x_i^*)^2 \ , \tag{21}$$

where r_i is a randomly shuffled array of integers between 1 and N. After every 1,000 generations, this array is changed to another permutation of integers from 1 to N. In addition, the optimum (x_i^* values) of the function is also changed to a random value in the range $[-1.0, 1.0]$. The parameter setting of a tournament size of 3 for real-parameter GAs and $\mu = \rho = 10$ for self-adaptive ES (which are used in the previous experiment) make the corresponding algorithm too sluggish to adapt to the changes made at every 1,000 generations. Thus, in this experiment, we use a tournament size of 2 for GAs and $\mu = \rho = 15$ with $c_{15/15,100} = 1.558$ (Beyer, 1995a) for ESs. Figures 15 and 16 show the population-best objective function value with GAs with SBX and non-isotropic ES, respectively. It is clear that, although the variation in solution vectors in the population is quite small at the end of 999 generations, the population can adapt to the change in the function landscape. The variation of population standard deviation in x_{15} under GAs



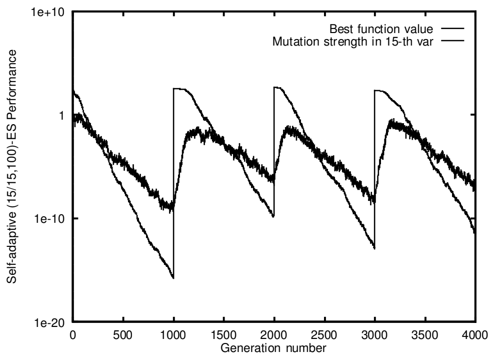


Figure 15: Population-best objective function value and the standard deviation of x_{15} variable are shown for real-parameter GAs with the SBX operator on function F2-2.

Figure 16: Population-best objective function value and the mutation strength for x_{15} variable are shown for self-adaptive (15/15,100)-ES on function F2-2.

with SBX and variation of mutation strength in x_{15} under ESs show remarkable similarity in their working principles. In this figure, the population-best objective function value and the mutation strength for x_{15} variable are shown. The remarkable similarity in both figures suggests that for chosen parameter settings, both real-parameter GAs with the SBX operator and the self-adaptive ES have very similar working principles.

5.3 Correlated Function

Next, we consider a function where pair-wise interactions of variables exist. Schwefel's function is chosen:

$$\text{F3-1:} \quad \text{Minimize} \sum_{i=1}^{N} \left(\sum_{j=1}^{i} x_j \right)^2 . \tag{22}$$

The population is initialized at $x_i \in [-1.0, 1.0]$. Figure 17 shows the performance of real-parameter GAs with SBX ($\eta = 1$) and tournament size 3, non-isotropic (4,100)-ES, and correlated (4,100)-ES. Although all methods have been able to find increased precision in obtained solutions, as expected, the rate of progress for the real-parameter GAs with SBX is slower compared to that of the correlated self-adaptive ESs. However, GAs with SBX make steady progress towards the optimum and perform better than the non-isotropic (4,100)-ES. The reason for the slow progress of real-parameter GAs is as follows. In the SBX operator, variable-by-variable crossover is used with a probability of 0.5. Correlations (or linkage) among the variables are not explicitly considered in this version of SBX. Although some such information comes via the population diversity in variables, it is not enough to progress faster towards the optimum in this problem compared to the correlated ES, where pairwise interactions among variables are explicitly considered. Clearly, a better real-parameter crossover operator handling the linkage issue is needed to solve such problems faster. One such implementation is suggested in Section 6.

Besides the linkage issue, there is another mismatch between the GAs with SBX and the correlated self-adaptive ESs used above. In GAs, a selection pressure of 3 (best solution in a population gets a maximum of three copies after the tournament selection

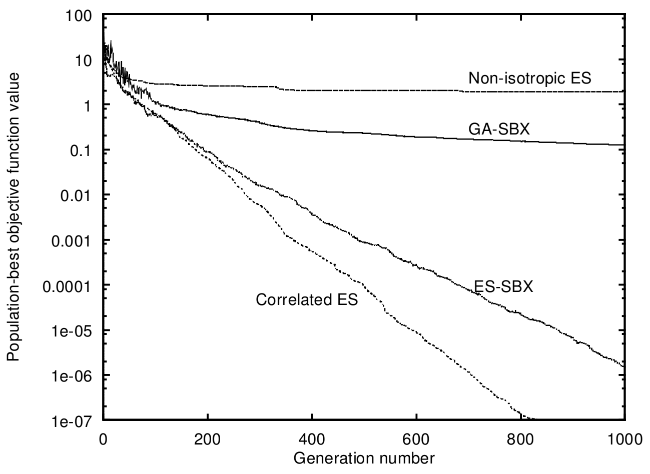


Figure 17: The best objective function value in the population is shown for four evolutionary algorithms on function F3-1: real-parameter GAs with SBX, non-isotropic (4,100)-ES, correlated self-adaptive (4,100)-ES, and (4,100)-ES-SBX.

operation), whereas in the correlated self-adaptive (4,100)-ESs, only 4 best solutions are picked from 100 children solutions. In order to alleviate this mismatch in selection pressures, we use a different algorithm (we call ES-SBX), where a (μ, λ)-ES is used, but λ children solutions are created from μ parent solutions only by the action of the SBX operator alone. Each parent $(x^{(1,t)})$ mates with other parents in exactly λ/μ crossovers, every time creating one child solution $(x^{(1,t+1)})$ using Equation 6. Like the SBX operator used in GAs, every variable is crossed with a probability 0.5. If a variable is not to be crossed, its value $(x_i^{(1,t)})$ in the first parent is directly passed on to the child. No mutation operator is used. As shown in Figure 17, this new (4,100)-ES-SBX algorithm is able to achieve much better performance than GAs with the SBX operator, although no explicit pair-wise correlations among variables are used in any operator. However, we recognize that the linkage issue may still be a problem in general, but this simple change in the algorithm seems to fair well in a problem where finding pair-wise linkage information is important.

5.4 Ridge Model

Above problems have tested the ability of algorithms to converge near the true optimum with increased precision. We now test algorithms for an opposing characteristic. We consider the following function, which is largely known as the ridge function in the ES literature (Oyman et al., 1998):

$$\text{Maximize} \quad v^T x - d\left(||(v^T x)v - x||\right)^{\alpha}, \tag{23}$$

where x is the N-dimensional variable vector and v is the ridge axis (or the direction vector specifying the ridge axis). Since the objective is to maximize the overall function, there are two subgoals: (i) maximize the distance along the ridge axis $(v^T x)$ and (ii) minimize the distance to the ridge axis $(||(v^T x)v - x||)$. The parameters d and α govern the shape of ridge functions. Higher values of d make the second term more prominent compared to the first term and therefore make an algorithm difficult to progress along

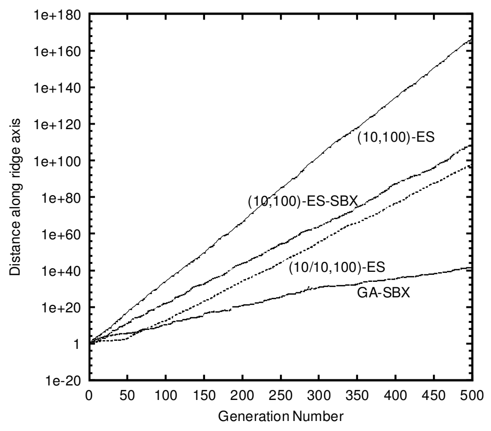


Figure 18: Performance of real-parameter GAs with SBX, non-isotropic self-adaptive ESs, and (10,100)-ES-SBX are shown on the ridge function F4-1.

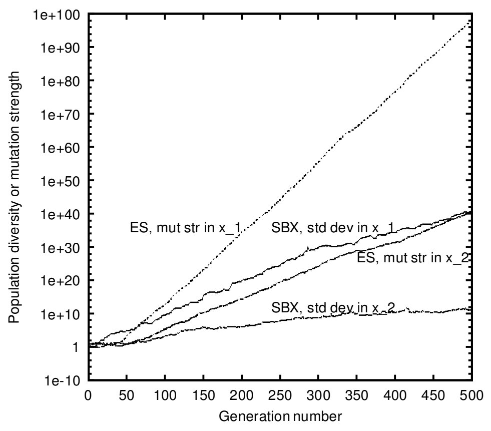


Figure 19: Population standard deviation in x_1 and x_2 for GAs with SBX and population average mutation strengths in x_1 and x_2 for (10/10,100)-ES are shown for the ridge function F4-1.

the ridge axis. In all simulations here, we use a comparatively large d ($= 1$). The parameter α has a direct effect on the fitness landscape. Usually, two values of α are commonly used: $\alpha = 2$ is known as the parabolic ridge function and $\alpha = 1$ is known as the sharp ridge function. Here, we shall consider the parabolic ridge function only.

The ridge functions have their theoretical maximum solution at infinity along the ridge axis in the search space. Since the maximum solution lies on the ridge axis, this function tests two aspects: converging ability on the axis and diverging ability in the direction of the ridge axis. We test the performance of real-parameter GAs with SBX and the non-isotropic self-adaptive ES.

5.4.1 Function F4-1: Parabolic Ridge

Here, we choose $v_1 = 1$ and all other $v_i = 0$ for $i = 2, \ldots, N$. This makes the first coordinate axis as the ridge axis. In all simulations, we use $N = 30$ and the population is initialized in $x_i \in [-2, 2]$. Real-parameter GAs with SBX ($\eta = 1$) and with tournament selection ($s = 2$) are used. A population of size 100 is used. Figure 18 plots the distance along the ridge axis (the term $v^T x$) with generation number. The figure shows that real-parameter GAs are able to progress towards infinity along the ridge axis exponentially with generation number (the ordinate axis is plotted in logarithmic scale). Runs with non-isotropic self-adaptive (10,100)-ES and (10/10,100)-ES with one independent mutation strength parameter for each variable show a similar behavior, but with much better progress rate. Since the optimum is at infinity along the ridge axis, an algorithm with faster divergence characteristics is desired. The figure shows that the recombination of problem variables and mutation strengths has an adverse effect on the performance of ES in this function.

Figure 19 shows the population standard deviation in x_1 and x_2 for real-parameter GAs with the SBX operator. The figure also shows the population average mutation strengths in x_1 and x_2 for (10/10,100)-ES as they evolve for the parabolic ridge function. It is clear that in both algorithms the population diversity or mutation strength

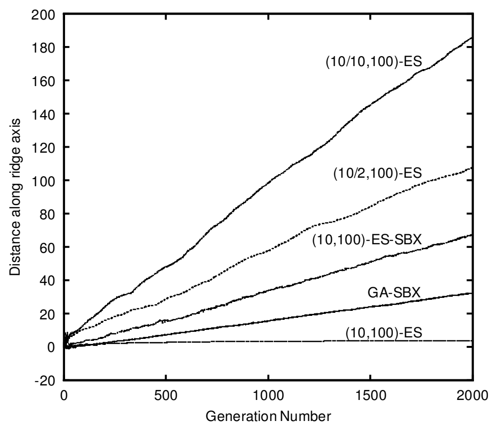


Figure 20: Performance of real-parameter GAs with SBX, non-isotropic self-adaptive ESs, and (10,100)-ES-SBX are shown on the rotated ridge function F4-2.

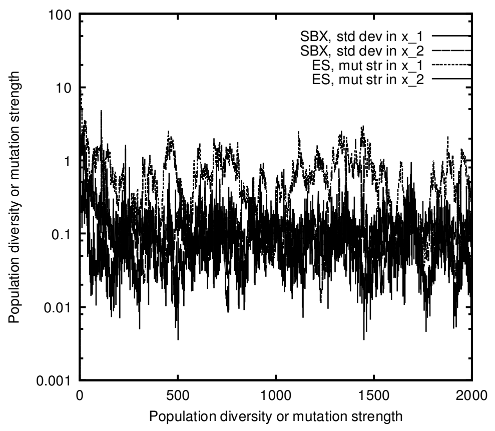


Figure 21: Population standard deviation in x_1 and x_2 in GAs with SBX and population average mutation strengths in x_1 and x_2 in (10/10,100)-ES are shown for the rotated ridge function F4-2.

for x_1 grows much faster than that in x_2. This is what we expected. Increasing the population diversity or mutation strength at a faster rate will allow the corresponding algorithm to move quicker along the ridge axis. On the other hand, since the secondary goal is to come closer to the ridge axis, an ideal situation would be to reduce the population diversity or mutation strengths of other variables (x_2 to x_N) as small as possible. However, both algorithms resorted to increasing these quantities with generation but at a rate much smaller than the growth in x_1 variable.

Next, we apply (10,100)-ES-SBX (where SBX is used as the only search operator in an ES, as described in the previous subsection). Figure 18 shows that a better progress compared to GA-SBX is obtained with ES-SBX algorithm. This is mainly due to the effect of larger selection pressure in ES-SBX.

5.4.2 Function F4-2: Parabolic Ridge with Rotated Ridge Axis

In this case, we use a random $\vec{v}$, so that the ridge axis is now not along any coordinate direction. Parameters as that used in F4-1 for real-parameter GAs with SBX and self-adaptive ES are chosen here, and an identical $\vec{v}$ is used for all algorithms. Figure 20 shows the progress towards the ridge axis with real-parameter GAs, non-isotropic self-adaptive ES, and the ES-SBX algorithm. The self-adaptive ES has a faster progress rate. However, notice that due to the parameter interactions, the progress along the ridge axis is now not exponential to the generation number, rather the progress is linear. Since to improve along the ridge, all N variables need to be changed in a particular way, the progress slows down. However, both algorithms have been able to maintain a steady progress. Unlike in function F5-1, recombinations in problem variables and mutation strengths help achieve better progress. Notice that a (10,100)-ES does not perform well in this function. Since the main goal is to progress along the ridge axis, an intermediate recombination operator helps in creating solutions close to the ridge axis from solutions around the ridge axis. Thus, recombinative ESs are advantageous in problems where the goal is to move along a fixed direction. Like in F4-1, ES-SBX performs better than GA-SBX in this function as well.

Figure 21 shows the population average standard deviation in x_1 and x_2 for real-parameter GAs with the SBX operator and the population average mutation strength in x_1 and x_2 with (10/10,100)-ES. Contrary to their evolution in the parabolic ridge F4-1 (Figure 19), here, these quantities in both algorithms reduce and fluctuate in a certain range. More importantly, these quantities for variables x_0 and x_1 now vary in the same range. This can also be explained as follows. In the rotated ridge function, the ridge axis $\vec{v}$ is a random direction, other than any coordinate direction. For the same reason, any orthogonal direction to the ridge axis is also a non-coordinate direction. In order to simultaneously satisfy both subgoals of maximizing progress towards the ridge axis and minimizing the distance to the ridge axis, the best an algorithm can do is to have a compromise in the rate of growth in each variable direction. In this case, both algorithms make careful changes to variables by keeping the population diversity or the mutation strength small so that a compromise in both subgoals is achieved. Naturally, this reduces the overall progress rate along the ridge axis. However, it may be mentioned here that such a linear behavior of rotated ridge functions is inevitable for any self-adaptive strategies that work with adaptations in variable directions independently. For a strategy that has the capability to independently adapt variations in arbitrary directions, an exponential progress towards optimum with generation number may be possible. For this matter, the correlated self-adaptive ES or the CMA algorithm (Hansen and Ostermeier, 1998) can be tried. However, the superior performance of non-isotropic recombinative ES in this function compared to GAs is interesting and needs further investigation.

6 Future Studies

This study suggests a number of extensions, which are outlined in the following:

1. Real-parameter GAs with the SBX operator can be compared with other self-adaptive ES implementations.
2. Other probability distributions, such as log-normal probability distribution, can also be investigated for self-adaptive behavior in real-parameter GAs.
3. Other existing real-parameter crossover operators can be investigated for their self-adaptive behavior.
4. Properties for an efficient crossover operator can be developed for real-parameter GAs.
5. A generic real-parameter crossover operator with more than two parents can be investigated for a faster progress rate.
6. A real-parameter crossover efficient for problems having correlated interactions among object variables can be investigated.
7. Real-parameter GAs can be compared with self-adaptive ESs in real-world complex search and optimization problems.
8. Properties for an algorithm to exhibit self-adaptation and test suites can be developed for testing the self-adaptive feature of an algorithm.
9. Discrete programming with real-parameter GAs with a modified SBX operator can be investigated for self-adaptation.

We discuss the above extensions in more detail.

In this study, we have shown that the real-parameter GA with the SBX operator has the self-adaptive behavior, similar to that in a self-adaptive ES with log-normal update of self-adaptive parameters. Other self-adaptive ES implementations also exist, and it will be interesting to compare the performance of real-parameter GAs with the SBX operator with them. Of them, the ES implementations by Hansen and Ostermeier (1996) can be investigated.

It is intuitive that the exact form of the polynomial probability distribution used in the SBX operator is not crucial in exhibiting self-adaptive behavior of GAs. However, it is important that the properties of the SBX operator described in Section 2 must be preserved in a probability distribution for a crossover. Thus, other more standard probability distributions used in ES, such as the log-normal distribution, can be used. In this regard, the following probability distribution as a function of β_i can be investigated instead of Equation 4:

$$\mathcal{P}(\beta_i) = \frac{1}{\sqrt{2\pi}\tau}\frac{1}{\beta_i}\exp\left(-\frac{1}{2}\frac{(\ln\beta_i)^2}{\tau^2}\right), \quad \beta_i > 0. \tag{24}$$

This probability distribution has its mode (maximum) at $\beta_i = \exp(-\tau^2)$, mean at $\bar{\beta}_i = \exp(\tau^2/2)$, exactly 50% probability of finding a $0 < \beta_i \leq 1$, and the rest 50% probability of finding $\beta_i > 1$. The above distribution has a variance $\sigma^2_{\beta_i} = \exp(\tau^2)(\exp(\tau^2) - 1)$. Since all statistical properties for this distribution are known, it may be easier to compare the performance of such crossover operators with self-adaptive ES that also uses the log-normal update rule.

Besides the SBX operator, there exist other real-parameter crossover operators such as BLX-α and UNDX operators that can be investigated for self-adaptive behavior. Both these operators create children solutions in proportion to the spread of parent solutions and are likely candidates for showing self-adaptive behavior. This study will show the importance of the actual form of the probability distribution used in exhibiting self-adaptive properties.

Studies of self-adaptation with various real-parameter crossover operators will reveal and allow us to find properties that are needed in an efficient real-parameter crossover operator. Such properties will help us create problem specific crossover operators, if needed. The evidence of self-adaptive behavior of real-parameter GAs with the SBX operator in this study suggests that, besides the properties mentioned in Section 2, the following are also important properties that a self-adaptive real-parameter GA should have in their search operator:

- The crossover operator must produce a children population that has the same mean as that in the parent population.
- The variance of the resulting children population may be larger than that of the parent population.

The first property distinguishes the crossover operator from the selection operator. The primary task of a crossover operator is to search the region represented by the parent solutions. There is no reason why a crossover operator should have any bias towards any particular region in the search space. It is precisely the task of the selection operator to guide the search towards better regions in the search space. The second property helps to maintain a balance between the spread of solutions under selection and crossover operators. Since selection emphasizes good solutions by eliminating bad solutions in a population, it may, in general, reduce the variance of the population. If the crossover operator also has a tendency to reduce the variance of the population, the overall search algorithm may not have adequate power to adapt to any function landscape. Thus, it may be desirable to have a crossover operator that, in general, increases the variance of the parent population.

In self-adaptive ES studies, it has been observed that using multi-parent crossovers provides better local convergence properties. In this study, we have only confined crossovers having two parents. However, a suitable extension to the SBX operator with more than two parents may lead to an algorithm with a faster progress rate. In the following, we outline one such possible algorithm.

It has been discussed earlier that in the SBX operator the variance of children distribution depends on the distance between the two parents. It is also important to use a distribution that assigns more probability for creating near-parent solutions than solutions away from the parents. It may not be of significant importance what distribution is actually used – whether the polynomial distribution used in this study or any other distribution with the above property. In order to be closer with mutation operators used in ES studies, a normal distribution with zero mean and a standard deviation proportional to the distance between two parents can be used, instead of the polynomial distribution. In such a scheme, a parent is first identified. A second parent is randomly chosen to compute the distance (either variable-by-variable or vector-wise) between both parents. Thereafter, a child is created with a normal distribution having its mean at the first parent and standard deviation proportional to the distance. In principle, such a crossover operator (it will still be a crossover operator, because more than one parent will be used to create a child solution) should also have the self-adaptive power. Once such a principle is established, a multi-parent (more than two parents) crossover can be designed. The distance measure used to define the standard deviation for the normal distribution can be computed as a weighted sum of the distances multiple parents have from the parent being mutated.

In solving the correlated functions and generalized ridge functions, it is observed that progress towards the optimum is linear, instead of exponential, to the generation number. One way to speed up the progress would be to use a correlated SBX operator that exploits the pair-wise interactions among variables. One such procedure would be to first use variable-by-variable SBX crossover. Thereafter, the resulting children solutions can be modified further by performing a line SBX operator on pairs of variables. In a line SBX operator, children solutions are found along the line joining the two parents. Such a pair-wise crossover for variables can allow progress of solutions in directions where correlated interactions exist.

The primary reason for emphasizing the research in real-parameter optimizations using evolutionary optimization techniques is their use in real-world search and optimization problems of science and engineering. With the indication of self-adaptive behavior of real-parameter GAs in this study, such GAs can be tried to solve real-world search and optimization problems, where self-adaptation is an essential feature needed in an algorithm to solve a problem.

In this connection, it is important to note that there exists no study identifying properties that a search and optimization algorithm should have in order for it to be qualified as a self-adaptive algorithm. Research effort in this direction is needed to develop better algorithms. Such a study may also help develop a suite of test problems for identifying self-adaptation properties in an algorithm. So far, self-adaptive evolutionary methods are applied only to a few simple functions. We have observed in this study that existing self-adaptive ES algorithms are vulnerable to changes in these test problems. A recent study (Grünz and Beyer, 1999) has shown that the theoretically optimal parameter settings for a non-recombinative self-adaptive ES do not work well for recombinative ESs. Thus, there is a need for studies developing problems that will test various aspects of self-adaptation: convergence with arbitrary precision, adaptation to non-stationary functions, finding true optimum in functions with inadequate knowledge of the location of optimum, and others.

Since a discrete version of the SBX operator can be used to solve discrete programming problems (Deb and Goyal, 1996, 1998) efficiently, GAs can be investigated for their self-adaptiveness in discrete search space problems. The extension of SBX to discrete search space is simple and straightforward. Instead of using a continuous probability density function, a discrete probability distribution (allowing non-zero probabilities only to acceptable solutions, and keeping the shape of the distribution similar to that in the SBX operator used in continuous search space) can be used to create children solutions. Rudolph (1994) used a self-adaptive ES on discrete search space problems. GAs with a discrete-version of the SBX operator can be tried to check if GAs can also exhibit self-adaptation in those problems.

7 Conclusions

In this paper, we have demonstrated that real-parameter genetic algorithms (GAs) with simulated binary crossover (SBX) exhibit self-adaptive behavior on a number of test problems. In this respect, a connection between the self-adaptive evolution strategies (ESs) with the log-normal update method and real-parameter GAs with the SBX operator has been discussed. A non-rigorous analysis has shown that both methods use similar probability distributions in creating children solutions, although a self-adaptive ES uses a single parent, and a GA with SBX uses two parents. Applications of both methods in a number of test problems borrowed from the ES literature, including sphere models and ridge models, reveal the remarkable similarity in their performances. Based on this study, a number of extensions to this study have also been suggested.

Acknowledgments

The first author acknowledges the support provided by the Alexander von Humboldt Foundation, Germany, and the Department of Science and Technology, Government of India, during the course of this study. The second author acknowledges support as a Heisenberg Fellow of the Deutsche Forschungsgemeinschaft under grant Be1578/4-1.

References

Bäck, T. (1992). The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm. In Männer, R. and Manderick, B., editors, *Proceedings of Parallel Problem Solving from Nature II*, pages 85–94, Springer Verlag, Berlin, Germany.

Bäck, T. (1997). Self-adaptation. In Bäck, T. et al., editors, *Handbook of Evolutionary Computation*, pages C.7.1:1–C.7.1:15, Oxford University Press, New York, New York.

Bäck, T. and Schwefel, H.-P. (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1):1–23.

Beyer, H.-G. (1995a). Toward a theory of evolution strategies: On the benefit of sex—the $(\mu/\mu, \lambda)$-theory. *Evolutionary Computation*, 3(1):81–111.

Beyer, H.-G. (1995b). Toward a theory of evolution strategies: Self-adaptation. *Evolutionary Computation*, 3(3):311–347.

Deb, K. and Agrawal, R. B. (1995). Simulated binary crossover for continuous search space. *Complex Systems*, 9:115–148.

Deb, K. and Goyal, M. (1996). A combined genetic adaptive search (GeneAS) for engineering design. *Computer Science and Informatics*, 26:30–45.

Deb, K. and Goyal, M. (1998). A robust optimization procedure for mechanical component design based on genetic adaptive search. *Transactions of the ASME: Journal of Mechanical Design*, 120:162–164.

Deb, K. and Kumar, A. (1995). Real-coded genetic algorithms with simulated binary crossover: Studies on multi-modal and multi-objective problems. *Complex Systems*, 9:431–454.

Eshelman, L. J. and Schaffer, J. D. (1993). Real-coded genetic algorithms and interval schemata. In Whitley, D., editor, *Proceeding of Foundations of Genetic Algorithms II*, pages 187–202, Morgan Kaufmann, San Mateo, California.

Fogel, L. J., Angeline, P. J., and Fogel, D. B. (1995). An evolutionary programming approach to self-adaptation on finite state machines. In McDonnell, J. R. et al., editors, *Proceedings of the Fourth International Conference on Evolutionary Programming*, pages 355–365, Morgan Kaufmann, San Mateo, California.

Goldberg, D. E. (1991). Real-coded genetic algorithms, virtual alphabets, and blocking. *Complex Systems*, 5:139–168.

Goldberg, D. E., Deb, K., and Clark, J. H. (1992). Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, 6:333–362.

Goldberg, D. E., Korb, B., and Deb, K. (1989). Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 3:93–530.

Grünz, L. and Beyer, H.-G. (1999). Some observations on the interaction of recombination and self-adaptation in evolution strategies. *Proceedings of the Congress on Evolutionary Computation*, pages 639–645, IEEE Press, Piscataway, New Jersey.

Hansen, N. and Ostermeier, A. (1996). Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 312–317, IEEE Press, Piscataway, New Jersey.

Hansen, N. and Ostermeier, A. (1997). Convergence properties of evolution strategies with the derandomized covariance matrix adaptation: The $(\mu/\mu_I, \lambda)$-CMA-ES. In Zimmermann, H.-J., editor, *European Congress on Intelligent Techniques and Soft Computing*, pages 650–654, Verlag Mainz, Aachen, Germany.

Hansen, N. and Ostermeier, A. (1998). Personal communication.

Harik, G. (1997). Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms. IlliGAL Report No. 97005, Illinois Genetic Algorithms Laboratory, University of Illinois, Urbana, Illinois.

Herdy, M. (1992). Reproductive isolation as strategy parameter in hierarchically organized evolution strategies. In Männer, R. and Manderick, B., editors, *Parallel Problem Solving from Nature II*, pages 207–217, Morgan Kaufmann, San Mateo, California.

Kargupta, H. (1996). The gene expression messy genetic algorithm. *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 814–819, IEEE Press, Piscataway, New Jersey.

Kita, H., Ono, I., and Kobayashi, S. (1999). The multi-parental extension of the unimodal normal distribution crossover for real-coded genetic algorithms. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 1581–1588, IEEE Press, Piscataway, New Jersey.

Ono, I. and Kobayashi, S. (1997). A real-coded genetic algorithm for function optimization using unimodal normal distribution crossover. In Eshelman, J. L., editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 246–253, Morgan Kaufmann, San Mateo, California.

Oyman, A. I., Beyer, H.-G., and Schwefel, H.-P. (1998). Where elitists start limping: Evolution strategies at ridge functions. In Eiben, A. E. et al., editors, *Parallel Problem Solving from Nature V*, pages 34–43, Springer, Berlin, Germany.

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, Frommann-Holzboog Verlag, Stuttgart, Germany.

Rechenberg, I. (1994). *Evolutionsstrategie'94*, Frommann-Holzboog Verlag, Stuttgart, Germany.

Rudolph, G. (1994). An evolutionary algorithm for integer programming. In Davidor, Y. et al., editors, *Parallel Problem Solving from Nature III*, pages 139–148, Springer, Berlin, Germany.

Saravanan, N., Fogel, D. B., and Nelson, K. M. (1995). A comparison of methods for self-adaptation in evolutionary algorithms, *BioSystems*, 36:157–166.

Schraudolph, N. N. and Belew, R. K. (1990). Dynamic parameter encoding for genetic algorithms. Technical Report No. LAUR90-2795, Los Alamos National Laboratory, Los Alamos, New Mexico.

Schwefel, H.-P. (1987). Collective phenomena in evolutionary systems. In Checkland, P. and Kiss, I., editors, *Problems of Constancy and Change—the Complementarity of Systems Approaches to Complexity*, pages 1025–1033, 31st Annual Meeting of the International Society for General System Research, Budapest, Hungary.

Schwefel, H.-P. (1988). Collective intelligence in evolving systems. In Wolff, W. et al., editors, *Ecodynamics—Contributions to Theoretical Ecology*, pages 95–100, Springer, Berlin, Germany.

Shaefer, C. G. (1987). The ARGOT strategy: Adaptive representation genetic optimizer technique. In Grefenstette, J. J., editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 50–58, Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Wright, A. (1991). Genetic algorithms for real parameter optimization. In Rawlins, G. J. E., editor, *Foundations of Genetic Algorithms*, pages 205–220, Morgan Kaufmann, San Mateo, California.