SIMULATED BINARY CROSSOVER FOR CONTINUOUS SEARCH SPACE

Kalyanmoy Deb and Ram Bhusan Agrawal

IITK/ME/SMD-94027
November, 1994

Convenor, Technical Reports
Department of Mechanical Engineering
Indian Institute of Technology
Kanpur, UP 208 016
India

æ

Simulated Binary Crossover for Continuous Search Space

Kalyanmoy Deb and Ram Bhushan Agrawal
Department of Mechanical Engineering
Indian Institute of Technology
Kanpur, UP 208 016, INDIA
email: deb@iitk.ernet.in

Abstract

The success of binary-coded genetic algorithms (GAs) in problems having discrete search space largely depends on the coding used to represent the problem variables and on the crossover operator that propagates building-blocks from parent strings to children strings. In solving optimization problems having continuous search space, binary-coded GAs discretize the search space by using a coding of the problem variables in binary strings. However, the coding of real-valued variables in finite-length strings causes a number of difficulties—inability to achieve arbitrary precision in the obtained solution, fixed mapping of problem variables, inherent Hamming cliff problem associated with the binary coding, and processing of Holland's schemata in continuous search space. Although, a number of real-coded GAs are developed to solve optimization problems having a continuous search space, the search powers of these crossover operators are not adequate. In this paper, the search power of a crossover operator is defined in terms of the probability of creating an arbitrary child solution from a given pair of parent solutions. Motivated by the success of binary-coded GAs in discrete search space problems, we develop a real-coded crossover (we called the simulated binary crossover or SBX) operator whose search power is similar to that of the single-point crossover used in binary-coded GAs. Simulation results on a number of real-valued test problems of varying difficulty and dimensionality suggest that the real-coded GAs with the SBX operator are able to perform as good or better than binary-coded GAs with the single-point crossover. SBX is found to be particularly useful in problems having multiple optimal solutions with a narrow global basin and in problems where the lower and upper bounds of the global optimum are not known a priori. Further, a simulation on a two-variable *blocked* function shows that the real-coded GA with SBX works as suggested by Goldberg and in most cases the performance of real-coded GA with SBX is similar to that of binary GAs with a single-point crossover. Based on these encouraging results, this paper suggests a number of extensions to the present study.

1 Introduction

The crossover operator is believed to be the main search operator in the working of a genetic algorithm (GA) as an optimization tool (Goldberg, 1989). The purpose of a crossover operator is two-fold. Initial random strings representing the problem variables must be searched thoroughly in order to create good strings. Thereafter, good portions of these strings must be combined together to form better strings. A number of crossover operators exist in the GA literature; however the search power to achieve both the above aspects differs from one crossover to another. The successful operation of GAs also depends on the coding mechanism used to represent the problem variables (Kargupta, Deb, and Goldberg, 1992; Radcliffe,

1993). Unless good building-blocks are coded *tightly*, the crossover operator cannot combine the building-blocks together (Goldberg, Korb, and Deb, 1989). This coding-crossover interaction is important for the successful working of GAs. The success of binary-coded GAs in problems having discrete search space is largely due to the careful choice of a coding and a crossover operator that complements each other's interaction in propagating useful building-blocks in successive generations. The problem of tight or loose coding of problem variables is largely known as the *linkage* problem, which has been recently tackled for binary-coded GAs by using messy genetic algorithms (Goldberg, Korb, Deb, 1989; Goldberg, et. al, 1993), In real-coded GAs, however, the variables are directly used. Thus, the tight or loose coding of bits constituting a variable does not arise, but the tight or loose coding of different variables still exists. Whether there really exists a real-coded GA to solve this linkage problem is a matter of interesting further research. However, in this paper, we recognize the importance of this aspect of coding inherent to problems with multiple variables in the successful working of GAs, but we do not address this issue. Instead, we devise a real-coded crossover operator that addressees some of the other difficulties inherent to binary-coded GAs in tackling problems having continuous search space.

In solving problems having continuous search space using binary-coded GAs, binary strings are usually chosen to code the problem variables. When a binary (discrete) coding is used for continuous search space, a number of difficulties arise. One difficulty is the Hamming cliffs associated with certain strings, from which a transition to a neighboring solution (in real space) requires alteration of many bits. Hamming cliffs in the binary coding cause artificial hindrance to a gradual search in the continuous search space. The other difficulty is the inability to achieve any arbitrary precision in the optimal solution. In binary-coded GAs, the string length must be chosen a priori to enable GAs to achieve a certain precision in the solution. The more the required precision, the larger is the string length. For large strings, the population size requirement is also large (Goldberg, Deb, and Clark, 1992), thereby increasing the computational complexity of the algorithm. Since fixed, mapped coding is used to code the variables, the variable bounds must be such that they bracket the optimum variable value. Since in many problems, this information is not usually known a priori, this may cause some difficulty in using binary-coded GAs in those problems. Further, a careful thinking of the schema processing in binary strings reveals that not all Holland's schemata are equally important in most problems having continuous search space. To a continuous search space, the meaningful schemata are those that represent the contiguous regions of the search space. The schema ($* \ * \ * \ * \ 1$), for example, represents every other point in the discretized search space. Although this schema may be useful in certain periodic or oscillatory functions, the schema ($1 \ * \ * \ * \ *$) signifies a more meaningful schema representing the right-half of the search space in most problems. Thus, the crossover operator used in the binary coding needs to be redesigned in order to increase the propagation of more meaningful schemata pertaining to continuous search space.

However, motivated by the success of binary GAs in discrete search space problem and

realizing the need of an efficient real-coded GA to eliminate the above difficulties of binary-coded GAs in continuous search space problems, we devise a real-coded crossover operator that simply simulates a binary crossover operator but without using the coding of variables. In order to achieve that we define the *search power* of a crossover operator in terms of the probability distribution of any arbitrary child string to be created from any two given parent strings[1] The search power of the single-point crossover operator is first calculated. Later, the simulated binary crossover (SBX) is developed to have a search power similar to that in the single-point crossover. The difference in the implementations of the real-coded GAs with SBX and binary-coded GAs with single-point crossover is that in the former method the coding of variables is eliminated and a child string is created from a probability distribution which depends on the location of the parent strings.

Although a number of other real-coded GAs exist in the literature (Eshelman and Schaffer, 1993; Wright, 1991), the search power, as defined in the paper, of these crossover operators is not adequate. Of them, the study by Eshelman and Schaffer is particularly important in that they introduced the concept of a schema (which they called interval schema) relevant for continuous search space. It may be mentioned here that the interval schemata are conceptually similar to the virtual alphabets introduced by Goldberg (1991) and locality formae defined by Radcliffe (1991). The virtual alphabets, the locality formae, or the interval schemata represent contiguous regions in the search space. Although Eshelman and Schaffer devised a blend crossover (BLX) operator, they did not construct the crossover operator from the interval-schema processing point of view. In a later section, we observe that their best-found crossover BLX-0.5 does not quite satisfactorily propagate interval schemata from parent to the children points. Interestingly, we find that a special case of SBX operator processes interval schemata better than their best-reported BLX-0.5 operator.

In the literature on evolution strategy, real-valued parameters are directly used, but mutation is used as the main search operator in those studies (Rechenberg, 1973). Only recently, crossover operators are being used in evolution strategy studies (Back, Hoffmeister, and Schwefel, 1991). For each variable, the *discrete* crossover chooses one of the parent values as the child, the *intermediate* crossover uses the average of the parent values as the child. Although other types of crossover operators are introduced, these operators are deterministic and can create only one search point. Thus, these operators do not have the adequate search power. However, if the effect of crossover and subsequent mutation operator are considered, their combined search power becomes significant. Since in this paper we mainly discuss and compare crossover operators, we do not consider the combined action of crossover and mutation operators of evolution strategy methods.

In the remainder of the paper, we investigate the efficacy of the real-coded GAs with the SBX operator on a number of carefully chosen test functions. Simulation results are compared

[1]The design of a crossover operator depends on a number of other factors such as respect, assortment, and transmissibility of formae (similarity templates) as suggested by Radcliffe (1991). The crossover operator designed here satisfies Radcliffe's design criteria, but it is not a R^3 (random, respectful, recombination) operator. Instead, the deviation from the random distribution of children and ability to access search points (Radcliffe's ergodicity criterion) is used to define the search power of a crossover operator.

with that of binary-coded GAs with single-point crossover and real-coded GAs with BLX-0.5 operator on the same functions. The test functions also include a two-variable, two-sided active blocked function introduced by Goldberg (1991). Goldberg argued that the working of the real-coded GAs is such that in these functions the population first converges to *virtual alphabets* (above-average points) and then the convergence to the global optimum gets blocked by two local optima. Our simulations using the SBX operator support Goldberg's theory of virtual alphabets, and the simulation results on this simple blocked function show that real-coded GAs with SBX perform similar to that of the binary-coded GAs with a single-point crossover. Simulation results also show that real-coded GAs with the SBX operator can overcome the Hamming cliff problem, precision problem, and fixed mapping problem and perform similar to binary-coded GAs with the single-point crossover (but better than real-coded GAs with BLX-0.5) in difficult problems. These results are encouraging and suggest the use of real-coded GAs with SBX operator to other optimization problems having continuous variables. Based on these results of this paper, a number of extensions to this work are outlined.

2 Binary Crossover Operator

In many applications of binary-coded GAs, a single-point crossover is used. In a single-point crossover, a random cross site along the length of the string is chosen and the bits on one side of the cross site are swapped between two parent strings. In a single-variable optimization problem, the action of the crossover is to create two new children strings from two parent strings. In a multi-variable optimization problem, each variable is usually coded in a certain number of bits (collectively called a substring) and these substrings are combined to form a bigger string. Since, in a single-point crossover operator, only one cross site is chosen from the complete string, the action of the single-point crossover may disrupt one variable or a combination of contiguous variables simultaneously. In the following, we calculate the probability of creating a child point from two chosen points under the single-point crossover.

2.1 Search power

The search power of a crossover operator is defined here as a measure of how flexible the operator is to create an arbitrary point in the search space. Radcliffe (1991) suggests a number of criteria for successful design of a crossover operator. That study shows how different existing crossover operators satisfy those criteria, but the study does not suggest any measure of comparing operators when they satisfy all required criteria. It is worth mentioning here that Radcliffe also defined an ergodicity criterion as the ability to access to any search point through genetic operators from any population. In this paper, we use Radcliffe's ergodicity criterion to define the search power of a crossover operator as the probability of creating an arbitrary point in the search space from two given parent points. To make the analysis simpler, we assume that the function is a single-variable function. Later, we shall discuss how the analysis can be extended for multi-variable functions.

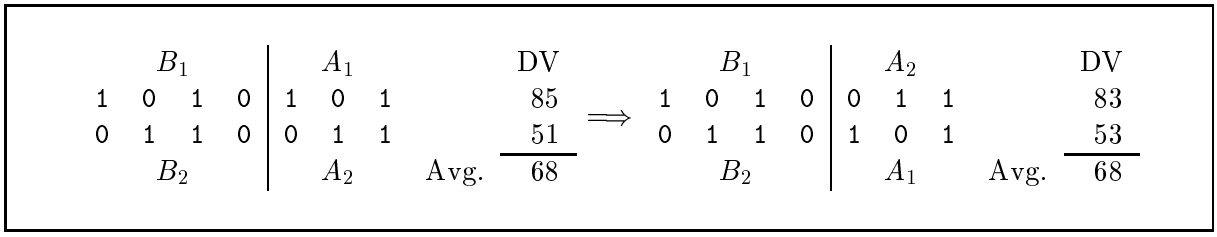

Figure 1: The action of single-point crossover on two random strings are shown. DV stands for the decoded parameter value. Notice that the average of the decoded parameter values is the same before and after the crossover operation.

When two parent binary strings of length ℓ are crossed at an arbitrary site, the contents on either side of the cross site get swapped between the parents. Let us imagine that the cross site is k bits from the right of the string. We also assume that the leftmost bit is the highest significant bit in the underlying binary coding. If the decoded value of the rightmost k bits in the first parent is A_1 and the decoded value of the rest $(\ell - k)$ bits is B_1, then the decoded value of the string is as follows:

$$x_1 = B_1 2^k + A_1. \tag{1}$$

The decoded value of the second parent string can also be calculated by knowing A_2 and B_2 as the decoded values of the right k bits and the rest $(\ell - k)$ bits of the string, respectively: $x_2 = B_2 2^k + A_2$. Figure 1 shows the single-point crossover operation on two arbitrary strings and the resultant children strings obtained by crossing at a particular site. The effect of the single-point crossover operator is to swap the values A_1 and A_2 between the two parent points. Thus, the decoded values of the children strings are as follows:

$$y_1 = B_1 2^k + A_2, \tag{2}$$

$$y_2 = B_2 2^k + A_1. \tag{3}$$

The above equations bring out an important property of the single-point crossover. Notice that the mean of the decoded values of the parent strings $((x_1 + x_2)/2)$ is the same as that of the children strings $((y_1 + y_2)/2)$. This can also be observed in Figure 1, by calculating the decoded parameter values of both parent and children strings. It is interesting to note that this property of decoded parameter values holds good for real parameter values of parent and children points for a linear mapping rule. Assuming a linear mapping rule, the parent points can be obtained as follows:

$$p_i = m x_i + t, \qquad i = 1, 2, \tag{4}$$

where m and t are two constants. The children points can also be computed using the same mapping rule as follows:

$$c_i = m y_i + t, \qquad i = 1, 2. \tag{5}$$

Using the above two equations and the previous observation on y_i, it is trivial to show that mean of the parent points is the same as that of the children points. In other words, the children

points are equidistant from the mid point of the parent points. We shall use this property to develop the real-coded crossover operator later. Further, we observe that the children points may lie inside or outside the region bounded by the parent strings depending on the the strings and the location of the cross site. In order to define the spread of the children points with respect to that of the parent strings, we define a *spread factor* β as the ratio of the spread of children points to that of the parent points:

$$\beta = \left| \frac{c_1 - c_2}{p_1 - p_2} \right|. \tag{6}$$

With the above definition of the spread factor, we classify crossovers in three different classes:

Definition 1 *All crossover operations having a spread factor $\beta < 1$ are contracting crossovers.*

When the children points are enclosed by the parent points, the spread of the children points (the absolute difference in c_1 and c_2 values) is smaller than that in parent points. Thus, the spread factor β is less than one. Since the children points usually replace the parent points in nonoverlapping genetic algorithms, this crossover seems to have an *contracting* effect on the parent points.

Definition 2 *All crossover operations having a spread factor $\beta > 1$ are expanding crossovers.*

When the children points enclose the parent points, the absolute difference in children points is more than that of the parent points. This crossover has an effect of *expanding* the parent points to form the children points.

Definition 3 *All crossover operations having a spread factor $\beta = 1$ are stationary crossovers.*

In stationary crossovers, the children points are the same as the parent points. One interesting feature of the spread factor β is that it *hides* the effect of the true spread of the children points. The spread factor β signifies a spread of children points relative to that of the parent points. For a fixed value of β, two close children points can be obtained if the parent points are close to each other or two distant children points could be obtained if the parent points are far away from each other. In a single-point crossover, the occurrence of a contracting, an expanding, or a stationary crossover depends on the parent strings and the location of the cross site.

With the above definitions of different types of crossovers, let us now compute the probability of occurrence of three different types of crossovers for any two random parent strings. We first consider that two ℓ-bit parent strings are crossed at a site $k \in (0, \ell-1)$ from the rightmost bit. Once again, we assume that the leftmost bit is the most significant bit. Representing two parent strings as $a_i, b_i \in \{0, 1\}$, we write the decoded values as follows:

$$x_1 = \sum_{i=0}^{\ell-1} a_i 2^i, \tag{7}$$

$$x_2 = \sum_{i=0}^{\ell-1} b_i 2^i. \tag{8}$$

Using the property of single-point crossover operator given in equations 7 and 8, we obtain the decoded values of the children strings as follows:

$$y_1 = \sum_{i=0}^{k-1} b_i 2^i + \sum_{i=k}^{\ell-1} a_i 2^i, \tag{9}$$

$$y_2 = \sum_{i=0}^{k-1} a_i 2^i + \sum_{i=k}^{\ell-1} b_i 2^i. \tag{10}$$

For a linear mapping rule (equations 9 and 10), the spread factor defined in equation 6 can be written in terms of decoded values of parent and children strings. Thus, equations 7 through 10 can be used to write the spread factor β. But we simplify the expression for the spread factor by assuming a parameter $u_i = (a_i - b_i)$ for all $i = 0, 1, \ldots, (\ell - 1)$. Note that each u_i can take a value -1, 0, and 1.

$$\begin{aligned}
\beta(k) &= \left| \frac{y_1 - y_2}{x_1 - x_2} \right|, \\
&= \frac{\sum_{i=k}^{\ell-1} u_i 2^i - \sum_{i=0}^{k-1} u_i 2^i}{\sum_{i=k}^{\ell-1} u_i 2^i + \sum_{i=0}^{k-1} u_i 2^i}, \\
&= \frac{1 - \zeta(k)}{1 + \zeta(k)},
\end{aligned}$$

where $\zeta(k) = \sum_{i=0}^{k-1} u_i 2^i / \sum_{i=k}^{\ell-1} u_i 2^i$. The above term for β is valid for any two parent strings (or in other words for any u_i values) and for any crossover site (or for any k). But, let us first consider the contracting crossovers (where $0 \leq \beta \leq 1$). In order to simplify our analysis further, we assume that parent strings are two extreme strings having all ones (`111...1`) and all zeros (`000...0`). In this case, the difference in allele values in each bit is one (or $u_i = 1$ for $i = 0, 1, \ldots, (\ell - 1)$). Substituting $u_i = 1$ in the above expression, we obtain the following expression for β:

$$\beta(k) = 1 - 2(2^k - 1)/(2^\ell - 1) \approx 1 - 2^{(-\ell+1)} 2^k. \tag{11}$$

The above spread factor for two extreme strings of length 20 crossed at different sites is plotted in Figure 2. It is clear from the figure that for most cross sites k, the spread factor $\beta \approx 1$.

With a distribution of the spread factor as a function of the cross site, we can now compute the probability of creating any arbitrary string (a representative β) from two given parent strings. Let us consider Figure 2 again. For a particular value of β, we consider a small range $d\beta$ and find how many crossover operations (dk) would produce children points in that range. This can be achieved by calculating the slope of the above expression at the corresponding β:

$$dk = \frac{1}{\left| \frac{d\beta}{dk} \right|_\beta} d\beta. \tag{12}$$

Denoting the probability of occurrence of β by $\mathcal{C}(\beta)$ and knowing that β lies between zero and one for contracting crossovers, we obtain the distribution for $d\beta \to 0$:

$$\mathcal{C}(\beta) = \frac{1/\left| \frac{d\beta}{dk} \right|_\beta}{\int_0^1 \left(1/\left| \frac{d\beta}{dk} \right|_\beta \right) d\beta}. \tag{13}$$

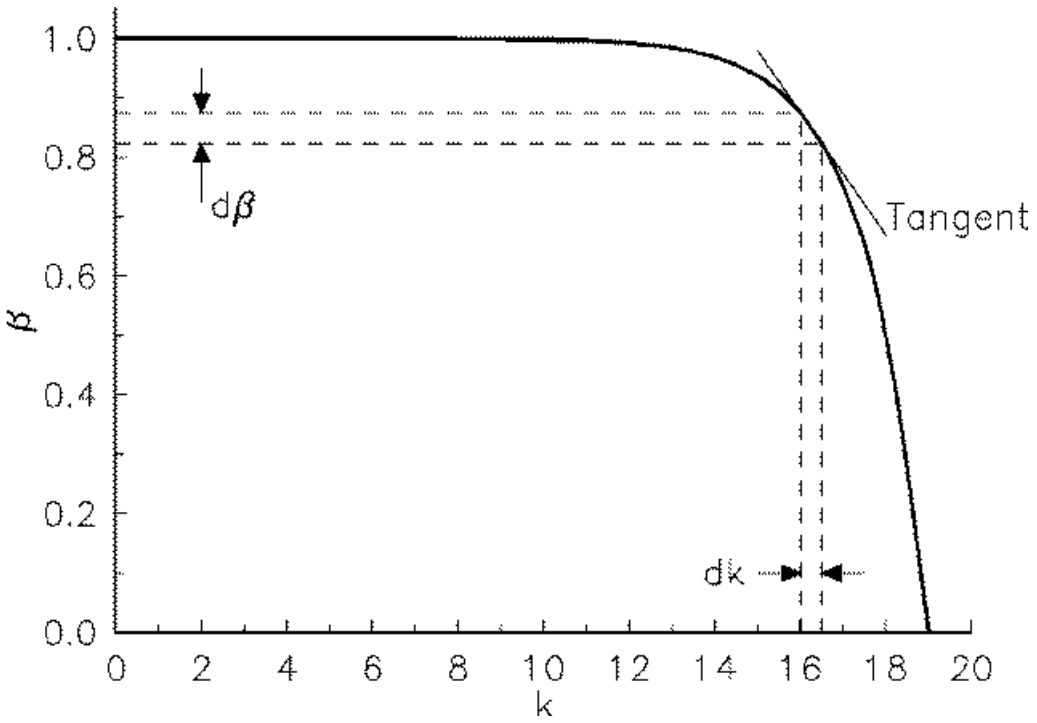

Figure 2: The distribution of $\beta(k)$ versus k for $\ell = 20$ is shown.

The denominator of the right-side expression is a constant quantity that makes the distribution $\mathcal{C}(\beta)$ a probability distribution. Here, we are not interested in computing the above probability distribution exactly, rather we are interested in finding the functional relationship of the probability distribution $\mathcal{C}(\beta)$ with β. Thus, it will suffice here to compute the numerator of the above expression only. By differentiating equation 11 with respect to k and substituting the expression for k in terms of β, we obtain the numerator of the right-side expression in equation 13 as follows:

$$\mathcal{C}(\beta) = \frac{\kappa}{1-\beta}, \tag{14}$$

where κ is a constant term. The functional form of the above distribution is shown in Figure 3 (for $0 \leq \beta \leq 1$). The figure shows that as β is increased towards one, the probability increases. Thus, the probability of creating children points closer to the parent points is larger than that of points far away from the parents.

The probability distribution for expanding crossovers ($\beta > 1$) can be obtained by considering two other parent strings[2] and by using the above procedure. However, the probability distribution can also be obtained from that of the contracting crossovers by realizing a simple probabilistic event. If two children strings found in a contracting crossover are crossed at the same site, the same parent strings are obtained. Thus, for each contracting crossover with a spread factor β, there exists an expanding crossover with a spread factor $1/\beta$. In other words, we can argue that the probability of occurrence of contracting crossovers having a spread fac-

[2]Notice that with two extreme strings considered in the contracting crossovers the expanding crossover is not possible.

tor between β and $(\beta + d\beta)$ would be the same as the probability of occurrence of expanding crossovers with a spread factor between $1/(\beta + d\beta)$ and $1/\beta$. Summing these probabilities for all contracting crossovers in the range $0 \leq \beta \leq 1$, we conclude that the overall probability of contracting crossovers is the same as that of the expanding crossovers[3]. Equating the probability of expanding crossovers ($\mathcal{E}(\beta)$) having spread factor between β and $(\beta + d\beta)$ and that of contracting crossovers ($\mathcal{C}(\beta)$) having spread factor between $1/(\beta + d\beta)$ and $1/\beta$, we obtain the following:

$$\mathcal{E}(\beta)\left[(\beta + d\beta) - \beta\right] = \mathcal{C}(\frac{1}{\beta})\left[\frac{1}{\beta} - \frac{1}{\beta + d\beta}\right]. \tag{15}$$

Rearranging terms and in the limit $d\beta \to 0$, we obtain

$$\mathcal{E}(\beta) = \frac{1}{\beta^2}\mathcal{C}(\frac{1}{\beta}). \tag{16}$$

With this relationship between the probability of occurrence of contracting and expanding crossovers, we only have to know the probability distribution for only one type of crossovers. Since their sum is the overall probability of all crossovers, the overall probability for either contracting or expanding crossovers must be equal to 0.5. For the probability distribution given in equation 14, we obtain a probability distribution for expanding crossover using the above equation:

$$\mathcal{E}(\beta) = \frac{\kappa}{\beta(\beta - 1)}. \tag{17}$$

This distribution is plotted in the region $\beta \geq 1$ in Figure 3. The figure shows that the probability of creating children strings having a large β is small.

Recall that equations 14 and 17 are derived only for two extreme binary strings. In order to investigate the above distributions and probability distributions for other parent pairs, we create all pairs of binary strings of a particular length and cross them at all possible cross sites. For each crossover, the corresponding spread factor β is calculated and arranged in ascending order. The experiment is continued for string lengths of 10, 15, and 30. Figure 4 shows the distribution for $\ell = 15$. The figure shows that the probability of occurrence of $\beta \approx 1$ is more likely than any other β value. If the parents are closer, the spread of the two likely children is also smaller. On the other hand, if the spread of parents is more, the spread of likely children is also large. This aspect of the crossover operator distinguishes it from the mutation operator. In a mutation operator, the probability of a mutated string close to the parent string is higher, but this probability is usually constant and depends only on one string.

3 Simulated Binary Crossover (SBX)

In order to simulate the operation of a single-point binary crossover directly on real variables, we design a crossover operator that uses a similar probability distribution as obtained in the

[3]Of course, the stationary crossovers can be considered as either contracting or expanding. For the sake of argument, we can assume that half of the stationary crossovers are contracting and rest half of the stationary crossovers are expanding.

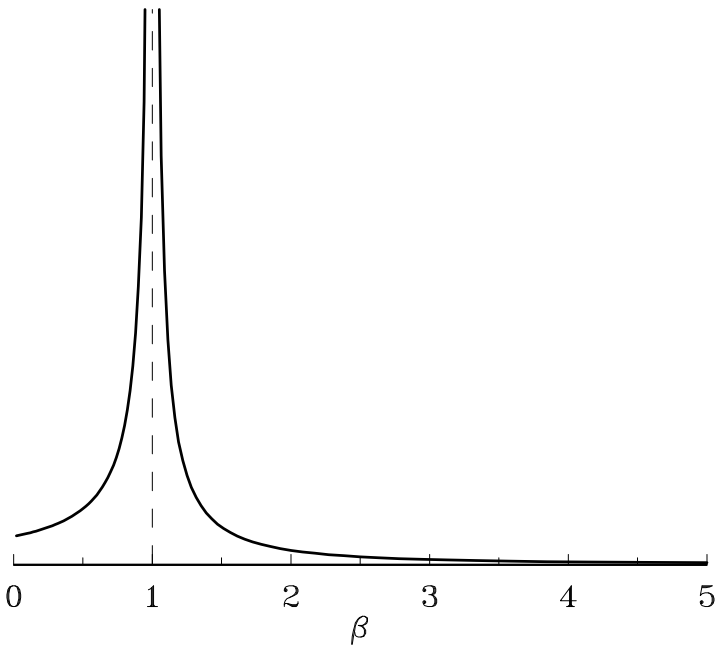

Figure 3: Probability distributions of contracting and expanding crossovers on two extreme binary strings are shown.

previous section. One property of a probability distribution is that the cumulative probability of the all possible states is equal to one. But the integration of the probability distribution in equation 14 over all values of β is not a finite number, this is because at $\beta = 1$ the probability does not exist. Moreover, recall that the distribution obtained in equation 14 is only for two extreme binary strings as parents. For contracting crossovers, we assume a simple polynomial probability distribution which is similar to the distribution in equation 13 and to that shown in Figure 4:

$$\mathcal{C}(\beta) = 0.5(n+1)\beta^n, \tag{18}$$

where n is any nonnegative real number. The factor 0.5 makes the cumulative probability at $\beta = 1$ equals to 0.5. A large value of n allows a better approximation of the above distribution with equation 14 for large values of β. A small value of n makes a more uniform distribution in the range $0 \leq \beta \leq 1$. A value of $n = 0$ makes a uniform distribution in the range (Figure 5). A moderate value of n (2 to 5) matches closely with the simulation results for the single-point crossover in binary-coded GAs.

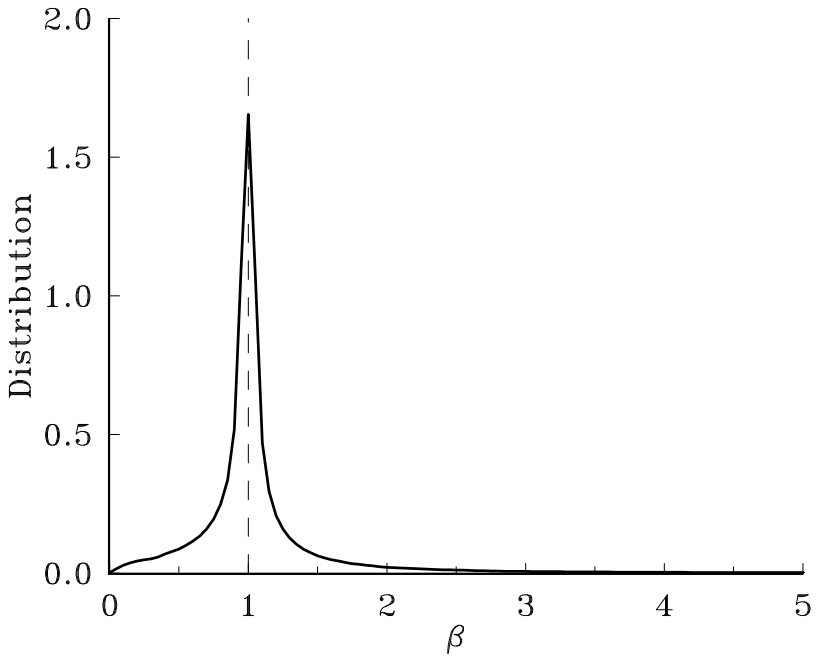

Figure 4: Probability distributions of contracting and expanding crossovers on all pairs of random binary strings of length 15 are shown.

For expanding crossovers, the probability distribution can be derived by using equation 16 and 18:

$$\mathcal{E}(\beta) = 0.5(n+1)\frac{1}{\beta^{n+2}}. \tag{19}$$

It is interesting to note that the probability of all expanding crossovers (can be found by evaluating $\int_{\beta=1}^{\infty} \mathcal{E}(\beta)d\beta$) is found to be 0.5, equal to that of the contracting crossovers. Also the probability of stationary crossovers obtained from both equations are the same.

The probability distribution for contracting and expanding crossovers for various values of n are shown in Figure 5. It is clear that the distribution largely depends on the exponent n. For small values of n, points far away from the parents are likely to be chosen, whereas for large values of n, only points closer to the parents are likely to be chosen. In a sense, the exponent n is similar to the reciprocal to the temperature parameter (T) used in simulated annealing algorithm (Aarts and Korst, 1989). Ideally, a good search algorithm must have a broad search (with large T or small n) in early generations and as the generations proceed the search must be focussed on to a narrow region (with small T or large n) to obtain better precision in the

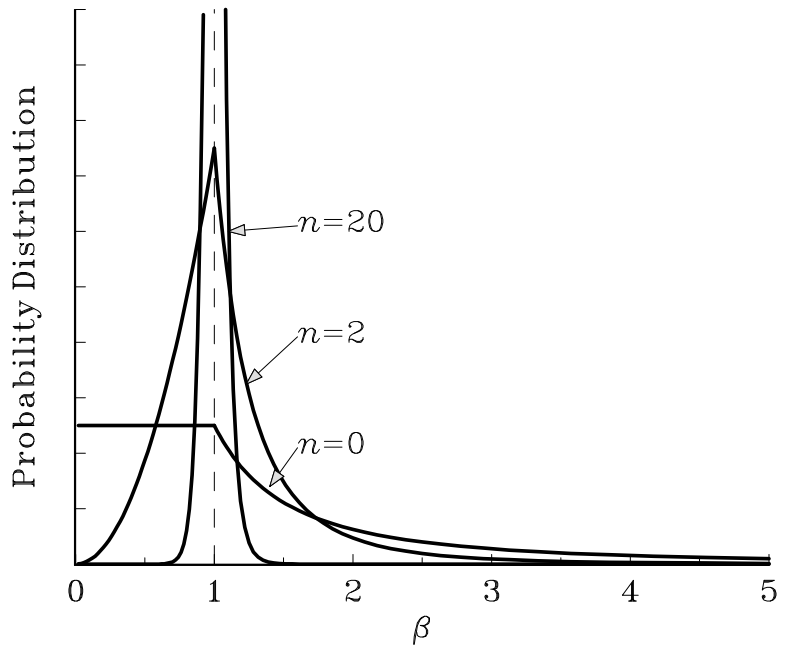

Figure 5: Probability distributions of contracting and expanding crossovers for the proposed polynomial model are shown.

solution. However, for brevity, we use a fixed compromised n for the entire simulation.

Figure 5 shows that the search of SBX crossover extends practically to any point in the real space. If lower and upper bounds for the variable is known, the above distributions can be restricted to that region, a matter we discuss in Section 6.

It is important to note that the polynomial probability distribution given in equation 18 is one of many possible probability distributions for contracting crossovers. For a different distribution, the corresponding probability distribution for expanding crossovers can be calculated by using equation 16. It is worth mentioning here that a distribution for expanding crossover can also be assumed first and then the corresponding probability distribution for contracting crossovers can be computed by replacing $\mathcal{E}(\beta)$ with $\mathcal{C}(\beta)$ in equation 16. In all such probability distributions, the probability of creating children points close to the parent points must be more than that of creating children points far away from the parent points (This simulates the effect of the single-point crossover as found in Figure 4). However, in the remainder of the paper, we use the polynomial probability distribution given in equations 18 and 19.

In the following, we discuss how the above analysis can be extended to multi-variable optimization problems.

3.1 Extension to multi-variables problems

The most common way to code multiple variables using a binary coding is to use a mapped concatenated string (Goldberg, 1989). In a single-point crossover, one cross site is chosen along the entire string and the contents on the right-side of the cross site are swapped between the parents. When binary-coded GAs are used to solve a N-variable function, and when the cross site falls on the bits representing the k-th variable from the left, the single-point crossover creates a new value of the k-th variable and swap the $(k+1)$-st to N-th variables between the parents. The same operation can be simulated by using the SBX operator on the k-th variable (k is chosen between 1 to N at random) and swapping the $(k+1)$-st to N-th variables between the parents. It has been shown elsewhere (Booker, 1993) that this feature of the single-point crossover causes a large positional bias for certain combinations of variables to get propagated from the parent to children. In this study, we choose a mechanism similar to *uniform* crossover for multiple variables (Syswerda, 1889) for choosing which variables to cross. However, we recognize here that this assumption is related to the important linkage issue of multivariable coding and our mechanism does not alleviate the linkage problem. In order to simplify the implementation of SBX operator, we choose to perform SBX crossover in each variable with a probability 0.5. But in the case of single-variable functions, since there is only one variable, the SBX is performed with a probability 1.0.

The concept of probability of crossover (p_c) used in canonical GAs still applies to this real-coded GA. This probability means that on an average $p_c N$ of the points in the mating pool are participated in the crossover.

4 Existing Real-coded Crossover Operators

The study of real-coded GAs has not received much attention among the genetic algorithm community. However, a nice description of different approaches to real-coded GAs is given in Eshelman and Schaffer (1993). The early work of Wright (1991) suggested using a linear crossover operator which compares only three points $(p_1 + p_2)$, $(1.5p_1 - 0.5p_2)$, and $(-0.5p_1 + 1.5p_2)$ and chooses the best two points. That work also suggested a crossover operator that creates a point by randomly perturbing a single variable in its range. Goldberg introduced the concept of virtual alphabets in the context of real-coded GAs (Goldberg, 1991). Although he did not suggest any particular crossover operator, he suggested blocked functions which may cause difficulty to real-coded GAs. Eshelman and Schaffer (1993) have introduced the notion of *interval* schemata for real-coded genetic algorithms and suggested a blend crossover (BLX-α) operator. For two parent points p_1 and p_2 (assuming $p_1 < p_2$), the BLX-α randomly picks a point in the range $(p_1 - \alpha(p_2 - p_1),\ p_2 + \alpha(p_2 - p_1))$. If α is zero, this crossover creates a random point in the range $(p_1,\ p_2)$. In a number of test problems, they have reported that

BLX-0.5 (with $\alpha = 0.5$) performs better than the BLX operators with any other α value. It is worthwhile to mention here that the interval schamata are similar in concept to Goldberg's virtual alphabets—both representing contiguous regions in the search space.

4.1 Search power

For the above real-coded crossovers, we investigate the search power, as defined in the previous section. The search power of Wright's linear crossover is clearly not adequate (too deterministic), because only three children points can be obtained from two parent strings. Wright's uniform crossover or Eshelman and Schaffer's BLX-0.0 operator has the flexibility of creating any point enclosed by the parent points. As discussed by Eshelman and Schaffer (1993), this crossover preserves the interval schemata representing contiguous regions in the search space. In the following, we take a closer look at the interval schema processing under BLX operator.

Consider a variable whose range is [0,8]. According to the formulations of Eshelman and Schaffer, there are 45 interval schemata for the above range of the variable (assuming that the points represented by an interval schema are bounded by integers only). Let us also consider two parent points $p_1 = 2$ and $p_2 = 5$, respectively. The first point represents a total of $(2 + 1)(9 - 2)$ or 21 interval schemata and the second point represents $(5 + 1)(9 - 5)$ or 24 interval schemata. These schemata are shown in the following table:

Number of interval schemata

Parent point at 2			Parent point at 5			
(2,2)			(5,5)			
(1,2)	(2,3)		(4,5)	(5,6)		
(0,2)	(1,3)	(2,4)	(3,5)	(4,6)	(5,7)	
(0,3)	(1,4)	(2,5)	(2,5)	(3,6)	(4,7)	(5,8)
(0,4)	(1,5)	(2,6)	(1,5)	(2,6)	(3,7)	(4,8)
(0,5)	(1,6)	(2,7)	(0,5)	(1,6)	(2,7)	(3,8)
(0,6)	(1,7)	(2,8)	(0,6)	(1,7)	(2,8)	
(0,7)	(1,8)		(0,7)	(1,8)		
(0,8)			(0,8)			

The common schemata between the two parent points are shown inside boxes and there are 12 such schemata. One underlying feature of a successful coding-crossover interaction is that a crossover must propagate the common schemata (appropriate to the coding) between the two participating parents to their children. The single-point crossover in the binary-coded GAs has this property for propagating Holland's schemata. In the real-coded GAs, if we assume that the interval schemata are what get processed, then the probability of creating a child point can be assumed to vary proportionately with the number of common schemata between the child and parent points. This argument in a sense signifies that a point *closer* (in the context of the interval schemata) to a parent point is more likely to get selected as a child point. With this argument, we can now find the probability distribution of a child point for the two chosen parents $p_1 = 2$ and $p_2 = 5$.

Any point in the range (2,5) contains all 12 common schemata of the two parents. Thus any child point in the range (2,5) is equally probable. Therefore, we may argue that the BLX-0.0 or Wright's uniform crossover (contracting crossover) propagates all common interval schemata from the parent to the children. But when BLX-α (α is positive) is used, all common interval schemata in both parents are not propagated to the children point outside the range enclosed by the parent points. For these crossovers (expanding crossovers), the probability can be assigned to depend linearly with the number of common interval schemata between children and parent points. In the above example, the point $c_1 = 1$ will represent only eight of the above common schemata, the point $c_1 = 0$ will represent four of the above common schemata, and so on. In general, the number of common interval schemata among any three points is $(x_l + 1)(N - x_h)$, where x_l and x_h are the lowest and the highest values of the three points and N is the entire range of the search space. In the above problem, since the range ([0,8]) is considered, $N = 9$. For children points $x < 2$, the number of common schemata is $(x + 1)(9 - 5)$ or $4(x + 1)$. For children points $2 \leq x \leq 5$, the number of common schemata is $(2 + 1)(9 - 5)$ or 12 (a constant) and for children points $x > 5$, the number of common schemata is $(2 + 1)(9 - x)$ or $3(9 - x)$. These numbers are plotted in Figure 6. If the probability of creating a child point is assigned according to these numbers, then the distribution is uniform for points enclosed by the parents and linearly varying for outside points. The BLX-0.5 crossover suggested by Eshelman and Schaffer (1993) deviates from this observation in the case of expanding crossovers (BLX-0.5 operator has a uniform distribution, as shown in Figure 7).

It is interesting to note that when these probability distributions are plotted as functions of the spread factor β (Figure 7), similar probability distributions emerge. The contracting crossovers have uniform probability distribution for any child point inside the region enclosed by the parent points and the expanding crossovers have a diminishing probability for children points away from the parent points. It is also interesting that the above probability distribution is a special case of the SBX operator developed in the previous section. For $n = 0$, all contracting crossovers have a uniform probability and the probability in the expanding crossovers diminishes with the increase in the spread factor. The only discrepancy between these two distributions is that in the SBX operator with $n = 0$, the probability reduces in a nonlinear manner and that in the ideal interval-schema processing, the probability reduces linearly. Nevertheless, we can argue that the SBX operator with $n = 0$ respects interval-schema processing better than the BLX-0.5 operator. However, other SBX operators with nonzero n do not respect interval schema. The above analysis suggests that the other SBX operators may respect some other schema which represents contiguous points but with varying importance. Since such schemata would be a variation of the interval schemata or locality formae, they would also satisfy Radcliffe's design criteria (Radcliffe, 1991). However, the formalization of these schemata is merely a mathematical exercise, which would be of little importance to the current study. Thus, we avoid that exercise in this paper and present simulation results of SBX operator on a number of test problems.

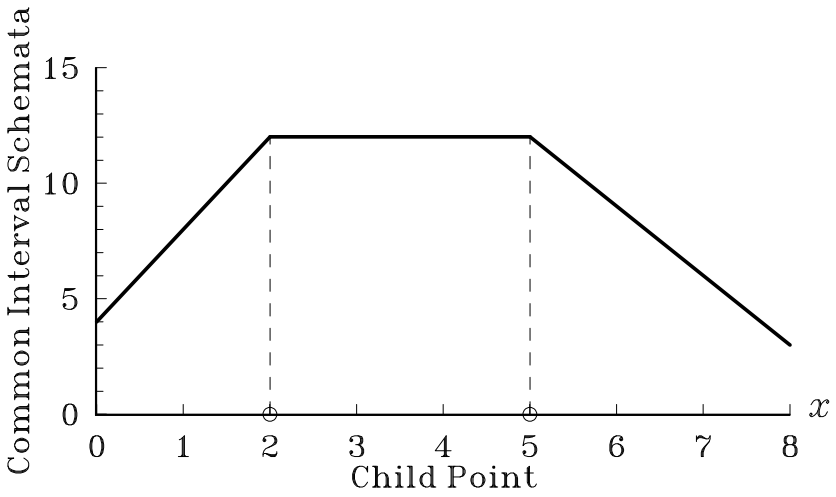

Figure 6: Number of common interval schemata between a children points and the parent points are shown. The parent points are at 2 and 5.

5 Simulation Results

To investigate the efficacy of the SBX operator, we have chosen a test bed of 12 functions, of which nine are taken from the existing literature. The test functions range from single-variable functions to multi-variable functions, from unimodal functions to multimodal functions, and from deterministic functions to stochastic functions. Finally, a two-variable blocked function is included to compare the performance of real-coded GAs with SBX and binary-coded GAs with single-point crossover on a difficult function.

A real-coded genetic algorithm is implemented using C programming language. The initial population of variable vectors is created in an initial range of variables supplied by the user. However, in subsequent generations, the population is not guaranteed to be confined to the given range, because the SBX operator has a nonzero (albeit small) probability of creating children points far away from the parent points. This is advantageous in problems where the location of the true optimum is not known beforehand. We demonstrate this flexibility of

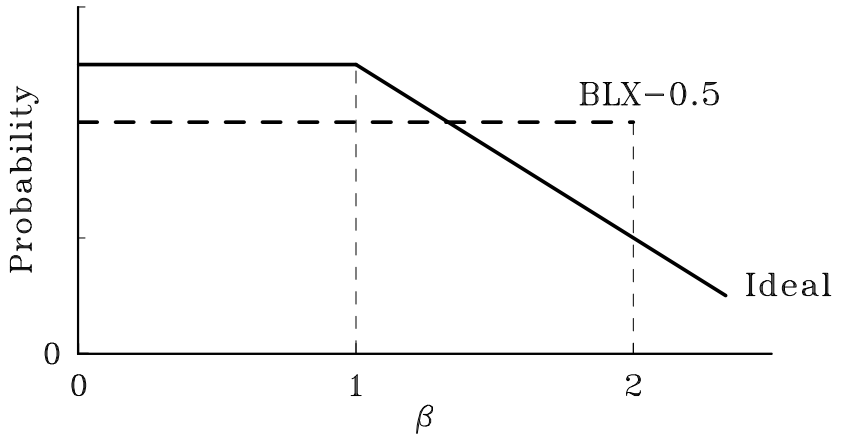

Figure 7: Probability distributions of BLX-0.5 and the ideal distribution are shown with spread factor β.

real-coded GAs with the SBX operator in test problems by initializing the population far away from the true optimum. However, this feature of the SBX operator is redundant for problems where the lower and upper limits of the variables are absolutely known. We discuss a simple modification to this SBX operator to confine the search in the given range in Section 6. In all our simulations, we allow the SBX to create any point in the real space with probability distributions given in equations 18 and 19. In order to investigate the effect of the probability distribution of contracting and expanding crossovers in the SBX operator, we present simulation runs with a number of different exponent values (n). It may be recalled here that for small values of the exponent n, the children points far away from the parent points can be obtained, but for large n, only points closer to the parent points are likely to be obtained. Mutation is not used to primarily investigate the search power of SBX operator alone. Four different criteria are used to terminate the search process. A simulation run is terminated if any one (or more) of them is satisfied. The criteria are as follows:

1. The best individual in the population is close to the true optimal point with a small

termination margin ϵ in all problem variables. In this case, we assume that the GA has found the true optimal point.

2. The function value of the best individual in the population is smaller than or equal to a target function value f_T. In this case, we assume that the GA has converged near the true optimal solution.

3. All individuals in the population have converged to a small neighborhood (within a small margin ϵ in all problem variables) to a point other than the global optimum point. In this case, a premature convergence occurs.

4. The number of generation has exceeded a specified large number $T_{\max}$. In this case, the GA has not converged to any solution at all.

Unless specified, all simulations use a target function value f_T equal to the function value at the global optimum point and a $T_{\max}$ value equal to 200. The performance of real-coded GAs with the SBX operator is compared with BLX-0.5 and with single-point crossover. In all simulations, a binary tournament selection without replacement and a crossover probability of 1.0 are used. Average number of function evaluations required to successfully solve a problem (when either criterion 1 or 2 above is satisfied) is tabulated for each problem. Other GA parameters are mentioned in the description of the simulation results.

5.1 V function

This function is a single-variable function defined in the range $0 \leq x \leq 1$ (Eshelman and Schaffer, 1993). The function has only one minimum point at $x = 0.5$:

$$f_1(x) = |x - 0.5|. \tag{20}$$

First, the binary-coded GAs are used. With a string length of 30 and a population size of 50, binary-coded GAs with the single-point crossover operator find a point with six decimal places of accuracy ($\epsilon = 10^{-6}$) only 23 out of 100 times. In rest of the runs, GAs have converged prematurely to a point close to the true optimum, but not as close as within ϵ from the optimum. The average function value of the best individual in all 100 runs is found to be 0.0015, which is very close to the function value of the true optimum ($f_1^* = 0.0$). It is important to note that the minimum point of the above V-function corresponds to a Hamming cliff in the binary coding used, but with $\epsilon = 10^{-6}$ there are about 2,147 strings representing the region ($0.5 - \epsilon$, $0.5 + \epsilon$). However, the presence of Hamming cliffs is one of the problems of the binary coding, as encountered by other researchers (Davis, 1991).

Real-coded GAs with the SBX operator are applied next. A summary of the simulation results is tabulated in Table 1. A termination margin of $\epsilon = 10^{-6}$ is used. The advantage of real-coded GAs over binary-coded GAs in this problem is largely due to the Hamming cliff problem at the optimum point. The results also show that the real-coded GAs with the SBX operator find the optimum point in all simulations with the chosen tolerance. When even

a smaller termination margin (up to the machine precision) is used, a similar performance of SBX is observed. This demonstrates the ability of the real-coded GAs to overcome the precision problem of binary-coded GAs. Notice that the performance of the SBX operator remains the same for different values of the exponent n. But for a large n, more number of function evaluations are required to find the optimum point.

The success rate of BLX-0.5 operator on this problem is equally well to SBX. Since the SBX operator performs expanding crossovers about the half the time and searches more region than BLX-0.5, the function evaluations required to solve the above problem is less in the case of BLX operator. But, when the same problem is solved with a population initialized in the range (0.9, 1.0), the performance of SBX is much better than that of the BLX-0.5 operator. The success rate of SBX is the same as before, but the success rate of BLX-0.5 reduces to about 25%. Recall that the minimum point does not lie in the range where the initial population is created. In order to find the optimum point, the search operator has to search outside this range. Obviously, the binary-coded GAs will fail to find the true optimum in this case, because of the fixed-mapped coding of the variable. The search power of SBX is further tested by starting with populations confined in the region (0.9999, 1.0000). The termination margin used in this case is also 10^{-6}. With SBX, the expanding crossovers find better points outside the initial range and the whole population migrates towards the true optimum. The success rate of SBX is again 100% in this case, whereas BLX-0.5 performs poorly.

Table 1: Simulation results of GAs with different crossover operators on the V-function are tabulated. The number in brackets in column six indicates the number of times GAs did not converge to any point at all.

Operator	Range	Popsize	n or ℓ	Performance of 100 runs		Avg. Trials
				Success	Prem. conv.	
1-pt	(0,1)	50	30	23	77	834.8
SBX	(0,1)	50	0	100	0	929.5
			2	100	0	748.5
			5	100	0	818.5
		100	0	100	0	1739.0
			2	100	0	1396.0
			5	100	0	1321.0
BLX-0.5	(0,1)	50		100	0	746.0
		100		100	0	1368.0
SBX	(0.9,1)	50	0	100	0	1279.5
			2	100	0	2018.5
BLX-0.5	(0.9,1)	50		24	76	2604.2
SBX	(0.9999,1)	50	0	100	0	1790.0
			2	100	0	4318.0
BLX-0.5	(0.9999,1)	50		0	61(39)	

5.2 V-cliff function

This function is similar to the V function, except that the function has a discontinuity at the minimum point ($x^* = 0.5$) (Eshelman and Schaffer, 1991):

$$f_2(x) = \begin{cases} 0.6 - x, & \text{if } x < 0.5; \\ x - 0.5, & \text{otherwise.} \end{cases} \tag{21}$$

The tolerance parameter $\epsilon = 10^{-6}$ is used again. Table 2 shows the performance of binary-coded GAs with the single-point crossover, real-coded GAs with SBX and BLX-0.5 on this function. The binary-coded GAs with single-point crossover is not able to find a solution

Table 2: Simulation results of GAs with different crossover operators on the V-Cliff function are tabulated. The number in brackets in column six indicates the number of times GAs did not converge to any point at all.

				Performance of 100 runs		
Operator	Range	Popsize	n or ℓ	Success	Prem. conv.	Avg. Trials
1-pt	(0,1)	50	30	16	84	840.6
SBX	(0,1)	50	0	88	0(12)	2,015.9
			2	100	0	1,382.0
			5	58	42	1,191.4
BLX-0.5	(0,1)	50		100	0	1,726.0
SBX	(0.9999,1)	50	0	88	0(12)	2,795.5
			1	100	0	3,253.0
			2	74	2(24)	7,147.3
BLX-0.5	(0.9999,1)	50		0	62(38)	

close to the true optimum in 100% simulations. The performance of binary-coded GAs is worse than the simple V-function, because of the Hamming cliff at the optimum point and chosen termination criterion (the convergence can practically occur from only one side of the optimum). The real-coded GAs with the SBX operator, on the other hand, performs well for small values of exponent n. The deterioration in the performance of SBX with either $n = 0$ and large n is due to the discontinuity at the optimum point. However, the performance of BLX-0.5 is 100% with slightly more function evaluations than that of SBX with $n = 2$. The table clearly shows that SBX is superior to BLX-0.5 when the population is initialized far away from the minimum point.

5.3 Bimodal, equal spread function

The bimodal function used here has two minimum points—one is better than the other. The basin of attraction of each minimum point is identical to the other:

$$f_3(x) = \begin{cases} -\exp(-(x-0.25)^2/0.01), & \text{if } x \le 0.5; \\ -0.5\exp(-(x-0.75)^2/0.01), & \text{otherwise.} \end{cases} \tag{22}$$

This function has a local minimum at $x = 0.75$ and a global minimum at $x = 0.25$. The function values at the local and global minima are -0.5 and -1.0 respectively. This function is chosen to test whether real-coded GAs with SBX can find the global minimum point or not. Table 3 compares the performance of both GAs for a termination margin of $\epsilon = 10^{-4}$. The performance of binary-coded GAs is inferior to that of real-coded GAs with SBX or BLX-0.5, because of problems related to the Hamming cliff at the optimum point and fixed-length coding. In binary-coded GAs, a larger population is required to achieve a desired solution accuracy for a problem coded in a bigger string (Goldberg, Deb, and Clark, 1992). Once again, real-coded GAs with both SBX and BLX-0.5 perform 100% of the time. Despite the presence of the local optimum, real-coded GAs find the true optimum point with the desired accuracy.

Table 3: Simulation results of GAs with different crossover operators on the function f_3 are tabulated.

				Performance of 50 runs		
Operator	Range	Popsize	n or ℓ	Success	Prem. conv.	Avg. Trials
1-pt	(0.0, 1.0)	100	20	49	1	765.3
			50	30	20	1023.3
SBX	(0.0,1.0)	100	0	50	0	870.0
			2	50	0	660.0
			5	50	0	642.0
BLX-0.5	(0.0,1.0)	100		50	0	670.0
SBX	(0.5,1.0)	100	0	4	46	4500.0
			2	9	41	1400.0
			5	1	49	1800.0
BLX-0.5	(0.5,1.0)	100		0	50	

The performance of neither of the two real-coded GAs is good for runs with adverse initial populations (with a population initialized in the local basin). Since, the population is initialized in the range $(0.5, 1.0)$, the global minimum point ($x^* = 0.25$) is not included in the range. Thus, the binary-coded GAs will not be able to find this minimum point with the above initial populations (problem of fixed, mapped coding). The performance of the real-coded GAs with SBX is marginally better than with BLX-0.5. Since about 50% of the points are created outside the range enclosed by the parents, a few points in the global basin are expected to be created by SBX. Since the binary tournament selection scheme and a $p_c = 1$ are used in the above simulations, the selection pressure is not enough to maintain those points in the global basin. When the crossover probability of SBX is reduced to $p_c = 0.5$, the disruption of good points reduces and the success rate of SBX increases to 34 out of 50 runs with $n = 0$ (with 2,006 function evaluations).

5.4 Bimodal, unequal spread function

This function is a modification of the previous function. Here, the basin of attraction of the global minimum is smaller than that of the local minimum:

$$f_4(x) = \begin{cases} -\exp(-(x-0.2)/0.04)^2), & \text{if } x \leq 0.3; \\ -0.5\exp(-(x-0.6)/0.4)^2), & \text{otherwise.} \end{cases} \quad (23)$$

Table 4 compares the performance of single-point crossover, SBX, and BLX-0.5 on this function. Compared to the previous function, the performance of the binary-coded GAs with single-point crossover in this function is marginally worse because the global basin is comparatively smaller. The performance of the real-coded GAs with SBX operator remains unchanged except that the SBX with a small exponent n does not work that well. The reason for its failure is the same as that in the previous function with adverse initial population. Although some points in the global basin are created, the selection pressure is not sufficient to maintain them in subsequent generations.

Table 4: Simulation results of GAs with different crossover operators on the function f_4 are tabulated. The number inside brackets in column six indicates the number of times the GA did not converge to any point at all.

				Performance of 50 runs		
Operator	Range	Popsize	n or ℓ	Success	Prem. conv.	Avg. Trials
1-pt	(0,1)	100	20	45	5	817.8
SBX	(0,1)	100	0	8	0(42)	450.0
			2	40	6(4)	1117.5
			5	50	0	730.0
			10	50	0	688.0
BLX-0.5	(0,1)	100		4	45(1)	1575.0

The difference in performance of SBX and BLX-0.5 is clear in this function. Since the local basin occupies most of the search space, a large portion of the initial random population is created in the local basin. About 45 out of 50 GA simulations with BLX-0.5 get confined to the local basin and finally converge to the local optimal solution. Even though some points are created in the global basin, they are difficult to maintain in the population, because of the large disruption rate associated with the BLX-0.5 operator. On the other hand, with SBX having a moderate n, the spread of the children points created using SBX is not substantial. The population gradually shifts towards the global basin and finally converges to the global optimum point.

5.5 Pole problem

The pole problem is a two variable function having four minimum points, of which one is the global minimum. The function is hypothetically constituted to have a light pole at each

minimum point with certain illumination capability. The illumination is maximum at the pole and diminishes away from the pole. When multiple poles are present in the search space, the mixed illumination results. For a given distribution of illumination levels in the search space, the objective of the problem is to find the point with maximum illumination. The problem can be written mathematically as follows:

$$\text{Maximize} \quad f_5(x,y) = \sum_{i=1}^{4} \frac{c_i h_i}{[h_i^2 + (x-x_i)^2 + (y-y_i)^2]^{3/2}}, \tag{24}$$

where (x_i, y_i) represents the coordinates of the poles, c_i represents the intensities of light at the i-th pole, and h_i represents the height of the i-th pole. In the parlance of optimization, the parameter, (x_i, y_i) represents the i-th optimum point, c_i/h_i^2 signifies the function value at the i-th peak, and h_i signifies basin of attraction of the i-th peak. In the above problem, there are four peaks with the following parameter values:

	i			
	1	2	3	4
x_i	0.4	0.3	0.7	0.8
y_i	0.3	0.7	0.2	0.8
c_i	1.0	1.0	1.0	1.125
h_i	0.1	0.1	0.1	0.075.

The optimization problem has a global maximum point at $x^* = (0.8, 0.8)$. Since this is a two-variable optimization problem, the technique described in Section 3.2 is used here. The SBX operator is performed for each variable with a probability 0.5. This causes about half of the variables (on an average) to undergo crossover operation. We follow this strategy for all multi-variable functions and for the BLX-0.5 operator. Table 5 shows the performance of single-point, SBX, and BLX on this problem for a termination margin $\epsilon = 10^{-3}$.

Table 5: Simulation results of GAs with different crossover operators on the function f_5 are tabulated. The number inside brackets in column five indicates the number of times the GA did not converge to any point at all.

			Performance of 10 runs		
Operator	Popsize	n or ℓ	Success	Prem. conv.	Avg. Trials
1-pt	200	20	6	4	3666.7
		30	9	1	3977.8
		40	8	2	2975.0
SBX	200	0	0	9(1)	
		2	8	2	3,375.0
		5	9	1	3,200.0
BLX-0.5	200		3	7	2,933.3

We observe that real-coded GAs with the SBX operator (for n between 2 to 5) perform as good as the binary-coded GAs with the single-point crossover. However, real-coded GAs with BLX-0.5 do not perform as well as other two GAs.

Table 6: Simulation results of GAs with different crossover operators on De Jong's five functions are tabulated. The number inside brackets in column seven indicates the number of times the GA did not converge to any point at all.

Function	Tolerance	Operator	Popsize	n or ℓ	Perform. of 50 runs		Avg. Trials
					Success	Premature conv.	
Function-I	10^{-2}	1-pt	100	30	25	25	2532.0
		SBX	100	0	50	0	2556.0
				4	50	0	2190.0
				10	24	26	2420.8
Function-II	10^{-3}	1-pt	100	24-40	0	50	
		SBX		0-20	0	50	
Function-III	1.0	1-pt	100	20	50	0	560.0
				50	50	0	720.0
		SBX		0	50	0	1256.0
				5	50	0	722.0
Function-IV	0.16	1-pt	100	240	4	42(4)	15200.0
		SBX		0	0	0(50)	
				2	50	0	17032.0
				5	50	0	9372.0
Function-V	0.1	1-pt	100	20	48	2	756.2
				40	49	1	836.7
		SBX		0	7	2(41)	5014.3
				5	36	14	1097.2
				10	37	13	891.9

From the above simulations, we observe that the BLX-0.5 operator performs well only on well-behaved unimodal functions having a continuous search space, but does not perform as well in problems with multiple optimum points. Moreover, the focus in this paper is to compare the SBX operator used in real-coded GAs with the single-point crossover operator used in binary-coded GAs. Thus, in subsequent simulations, we compare single-point and SBX operators only.

5.6 De Jong's functions

All five De Jong's test functions (De Jong, 1975) are tried next. The simulation results for the single-point and the SBX operators are tabulated in Table 6.

Function I is a three-variable unimodal function having a minimum point at $(0, 0, 0)^T$. In De Jong's study, the function is defined in the range $-5.12 \leq x_i \leq 5.12$ for $i = 1, 2$, and 3. In real-coded GAs with the SBX operator, points outside this range may also be created. Thus, we assume that the function is also defined outside the above range, but the initial

population is created in the above range. The simulation results (tabulated in Table 6) show that GAs with SBX are able to find the minimum point with two decimal places of accuracy in all simulations for small values of n. The binary-coded GAs with the single-point crossover find the minimum point in only 50% of the simulations. The results for SBX is expected because the function is unimodal and continuous in the real search space. In binary-coded GAs, however, the minimum point falls in a Hamming cliff, which causes difficulty to solve the problem in 100% simulations.

Function II is a two-dimensional minimization problem with the global minimum at $(1,1)^T$. The function was originally defined in the range $-2.048 \leq x_1, x_2 \leq 2.048$ for binary-coded GAs (De Jong, 1975). For real-coded GAs, the initial population is created in the above region. The results in Table 6 shows that both GAs can not find the optimum point $(1,1)^T$. In this function, the slope towards the optimum point is very small and the population gets converged to a point along the ridge. When the SBX operator is performed along a line joining the two parents (similar to Wright's line crossover (Wright, 1991) but a point is chosen with the polynomial probability distributions), near 100% convergence is observed for $n = 1$ to 3.

Function-III is a five-dimensional stair-case like function where the function value is monotonically nondecreasing as the variables are increased. The original function was designed for $-5.12 \leq x_i \leq 5.12$ for $i = 1$ to 5. Since the SBX operator may create children points outside this range, we have modified the original function by taking the absolute value of the function creating the initial population in the range $0 \leq x_i \leq 10.24$. This results in a minimum point at $x_i = 0$. In order to make the comparison fair, binary GAs are also run on this modified function in the range $0 \leq x_i \leq 10.24$. Both GAs perform equally well on this function. It is interesting to note that despite the function is discontinuous at many points in the search space, the real-coded GAs with SBX are able to find the true optimum. This is because the function is monotonically nondecreasing towards the optimum point and because the population-approach of GAs does not allow them to get stuck at any of the plateau.

Function-IV is a 30-dimensional unimodal function with a zero-mean Gaussian noise. The optimum point of the deterministic function is $x_i = 0$ ($i = 1$ to 30) with a function value equal to zero. But due to the addition of the Gaussian noise, the optimum point and hence the optimum function value changes. To decide the convergence to the optimum, we have assumed a target function value of $f_T = -3.0$. If any point has a function value smaller than f_T, the simulation is terminated. A termination margin of $\epsilon = 0.1$ is used. Real-coded GAs with the SBX operator perform better than the binary-coded GAs for $n = 2$ to 5, probably because the population size used in the simulation is inadequate in the case of binary GAs.

Function-V is a 25-dimensional, multimodal function having 25 minimum points, of which only one is the global minimum. The basin of attraction of the global minimum point is very narrow; thus, the classical optimization methods may have difficulty to solve the problem to global optimality. Simulation results show the superiority of binary-coded GAs with the single-point crossover on this function. The part of the success of binary-coded GAs in this problem is due to the biased positioning of the optimum points in equal intervals. When the

locations of 25 optima are randomly placed, the difference in the performance of binary-coded GAs and real-coded GAs is small—binary-coded GAs succeed only 64% of the simulations, whereas real-coded GAs with SBX succeed in 54% of the simulations.

From these simulations we conclude that real-coded GAs with SBX have performed better than binary-coded GAs on De Jong's functions having continuous search space and have performed similar to binary-coded GAs on discrete De Jong's test functions. In order to bolster our conclusions further, we apply SBX operator to two more functions.

5.7 Rastrigin's function

Rastrigin's function was used by a number of GA researchers in the recent past (Muhlenbein, Schomisch, and Born, 1991; Gordon and Whitley, 1993). The function has 20 variables and contains 10^{20} local minimum points, of which only one is the global minimum point:

$$f_{11}(x) = 200 + \sum_{i=1}^{20} x_i^2 - 10\cos(2\pi x_i), \quad -5.12 \leq x_i \leq 5.12. \tag{25}$$

The global minimum point is at $x_i = 0$ for all $i = 1, 2, \ldots, 20$. At this point, the function value is zero. The other local minima occur when the cosine term is one. This function is a difficult optimization problem, because there are 2^{20} different local optimal points (surrounding the global optimum) with function values very close to that at the global optimal point. We set $\epsilon = 0.1$ and the target function value $f_T = 1.0$. This allows a termination of the simulation when at least 19 out of 20 variables have converged to their global minimum values.

Binary-coded GAs with equal substring length for all variables are applied first. In all ten simulations, binary-coded GAs prematurely converge to a wrong solution (Table 7). The aver-

Table 7: Simulation results of GAs with different crossover operators on the function f_{11} are tabulated. The number inside brackets in column five indicates the number of times the GA did not converge to any point at all.

			Performance of 10 runs		
Operator	Popsize	n or ℓ	Success	Prem. conv.	Avg. Trials
Binary	400	200	0	10	
		300	0	10	
SBX	400	0	0	0(10)	
		2	10	0	49,680.0
		5	6	4	39,866.7
		10	10	0	28,000.0
		20	10	0	23,360.0

age function value of the best solution in all ten simulations is found to be 11.32. Real-coded GAs with the SBX operator perform better than binary-coded GAs. The average function value of the best solution in all simulations is 6.81 for $n = 2$ and 3.77 for $n = 10$, which are also better compared to that obtained for binary-coded GAs. The success of real-coded GAs in this

problem is due to the controlled allocation of children points under the SBX operator. With moderate n, the children points are not far away from the parent points. So the search slowly progresses towards the optimum point. It is worth mentioning here that with a tolerance of 0.1 in each of 20 variables, a enumerative search method requires about $[(5.12-(-5.12))/0.1]^{20}$ or $1.48(10^{40})$ function evaluations. Real-coded GAs took only a fraction of this search space to locate a near global minimum point. These results show the efficacy of real-coded GAs with the SBX operator in solving multi-variable, multi-modal functions.

5.8 A two-variable blocked function

A two-variable blocked function is designed according to Goldberg's (1991) suggestion. The mean (over x_1) and global slice (at x_1^*) of the function are plotted in Figure 8. The global

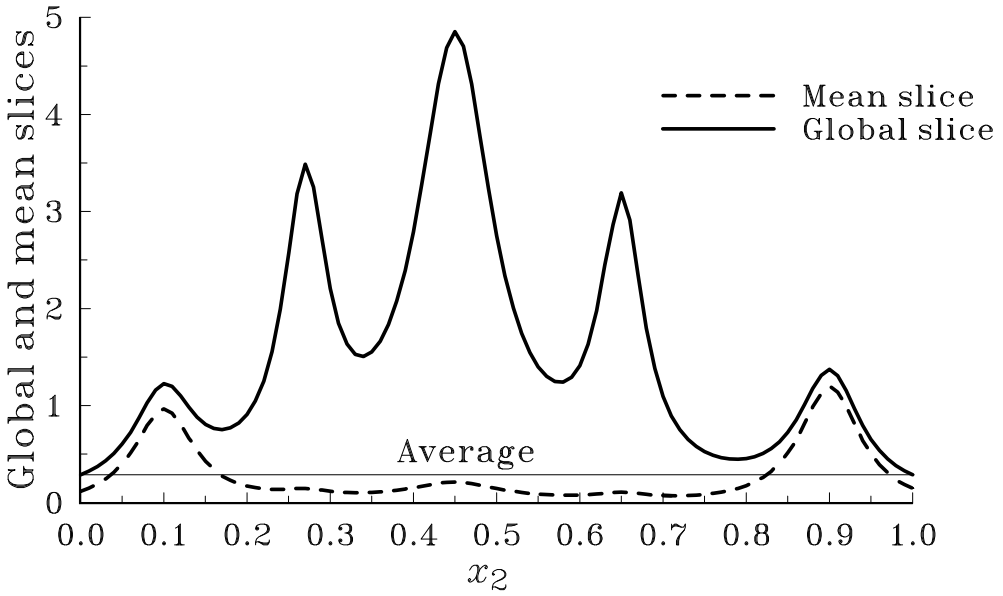

Figure 8: The mean slice and global slice (at x^*) of the function f_{12} are shown. The average function value over the search space is 0.287.

maximum point lies at the point $(0.4, 0.45)^T$. Two local *hills* are placed on either side of the global optimum. In the global slice (taken at $x_1 = 0.4$), the global maximum point is surrounded by two local maximum points, but the average slice (averaged on x_1 values) is

Table 8: Simulation results of GAs with different crossover operators on the function f_{12} are tabulated. The quantity in bracket indicates in column five the number of times GAs have converged to the global basin, but not to the global optimum point.

			Performance of 50 runs		
Operator	Popsize	n or ℓ	Success	Prem. conv.	Avg. Trials
Binary	100	20	5	45(27)	1,480.0
		30	15	35(27)	1,753.3
		50	6	44(32)	1,616.7
SBX	100	0	42	8(0)	2,323.8
		2	47	3(0)	1,872.3
		5	40	10(1)	1,677.5

predominant by the local peaks. It is interesting to note that the search along the variable x_1 is unimodal. Thus, the population is expected to converge at $x_1 = 0.4$. Once most of the population converges on or near $x_1 = 0.4$, the search progresses along the global slice. Since the population is converged to the extreme local peaks, it becomes difficult for classical algorithms to find the global point. The function is given below:

$$f_{12}(x_1, x_2) = -(x_1 - 0.4)^2 + \sum_{i=1}^{5} \frac{a_i}{b_i + r_i(x_1 - 0.4)^2 + (x_2 - c_i)^2}. \tag{26}$$

Here, the following parameters are used:

	i				
	1	2	3	4	5
a_i	0.002	0.0025	0.014	0.003	0.0028
b_i	0.002	0.0020	0.003	0.001	0.0010
c_i	0.1	0.9	0.45	0.27	0.65
r_i	0	0	10	10	10

Simulation results with binary-coded and real-coded GAs are tabulated in Table 8. Although the binary-coded GAs with single-point crossover converge to the global optimum point (with three decimal places of accuracy) only 5 to 15 times, they converge to the global basin in about 80% of the simulations. Similar results are observed with the real-coded GAs with SBX; in at least 80% of the runs, they have converged to the global maximum point. In all successful simulations of real-coded GAs, it is observed that the variable x_1 converges to the true optimal value first, as predicted by Goldberg (1991). Subsequently, the search along the global slice continues and the population located at two extreme optima improves gradually and can overcome the barrier of the local peaks to finally converge to the global optimum. Whenever real-coded GAs have failed to find the global optimum point, the convergence to the variable x_2 has occurred first. When this happens, the population shifts in any one or both the peaks shown on the mean slice plot (dashed line) in Figure 8. Since the function on x_1 is unimodal with maximum at $x_1 = 0.4$, the solution converges to a wrong solution.

Further experiments are necessary to conclude how the proposed algorithm will work on higher-dimensional blocked functions, but the simulation results on the above two-variable blocked function show that although the SBX operator works on real parameter values, its search power in solving the function is similar to that in the single-point crossover.

6 Extensions

The above study opens a number of avenues for further research. Some of them are discussed below:

1. Bounded search space,
2. Selection of other form of probability distributions for contracting and expanding crossovers,
3. Real-coded mutation operator,
4. Application of SBX to real-world optimization problems,
5. Simulating other binary crossovers for real-coded GAs,
6. Population sizing and convergence proof,
7. Multimodal and multiobjective optimization using real-coded GAs.

In the present study, we have allowed the children points to lie anywhere between negative infinity to positive infinity. In certain problems, the search space could be bounded. For example, if the length of a member in an engineering component design is a variable, negative values of the variable do not make sense. The SBX operator used in this study does not always guarantee a child point with a positive value, even though both parents may have positive values. The contracting and expanding probability distributions can be modified so that the probability of creating points outside a given bound is zero, while maintaining the same nature of the probability distribution. One way to achieve this is that for any two given parent points, the cumulative probability of the children points within the given bound is first calculated. If this probability is a (say), the probability distribution function is multiplied by a factor $1/a$, so that the overall probability for points within the bounds is equal to one.

The probability distribution for contracting crossover used in this paper is a polynomial function to the spread factor. This distribution is used to reduce the computations required in each crossover operation. Other functional forms of the probability distributions can also be used, but care must be taken to choose a distribution which is similar in form to that given in Figure 4. In this respect, a Gaussian-like distribution with mean at $\beta = 1$ and variance (σ^2) chosen to have at most $\pm 3\sigma$ points within a specified bound can also be used.

In this paper, we investigated the effect of the crossover alone, keeping the mutation probability zero. In order to maintain diversity in the population, a mutation operator can be used. In real-coded GAs, the mutation operation can be achieved by choosing a neighboring point

in the vicinity of the current point. This can be implemented by fixing an explicit probability distribution or by adopting the methods used in other population-based real-coded studies (Back, Hoffmeister, and Schwefel, 1991).

The present study shows that real-coded GAs is preferred to binary-coded GAs for problems with continuous search space and for problems where arbitrary precision is desired. Many engineering design optimization problems contain mixed variables—both discrete and continuous. The classical optimization methods do not perform very well in solving these kinds of problems (known as mixed integer programming problems). GAs with a combination of both binary coding (for discrete variables) and real coding (for continuous variables) can be used to solve these problems efficiently. The crossover operation can be performed as follows. A variable is first chosen for crossover. If a discrete variable needs to be crossed, the single-point crossover or another binary operator can be used to the bits representing that variable only. But if a continuous variable needs to be crossed, the SBX operator can be used. This way the resulting string also becomes a valid string. We are currently applying this technique to truss-structure optimization problems, where optimization of truss topology (represented by binary variables) and cross-section of members (continuous variables) are achieved simultaneously, an approach which is ideal but can not be achieved using classical optimization methods.

The SBX operator developed here is based on the probability distribution of children strings in a single-point crossover. Similar probability distributions (a function of the spread factor β) can also be obtained for multi-point and uniform crossovers and other similar real-coded crossovers can be designed. Probability distributions similar to the one developed here are expected, it may be interesting to investigate whether they turn out otherwise. Depending on the outcome of that analysis, the issue of superiority of one crossover over other can be resolved, particularly in the context of GA's application to continuous variables.

Throughout in this study, reasonable values of the population size are used. As demonstrated elsewhere (Goldberg, Deb, and Clark, 1992), the proper population sizing is important for successful working of GAs. That study also suggested a population sizing based on the ratio of the difference in detectable function values to the variance of the function. An attempt can be made to derive a population sizing for real-coded GAs from a similar viewpoint. In binary-coded GAs, the major difficulty towards achieving a convergence proof lies in the mechanics of the crossover operator. Although the mechanics of the single-point crossover (or for that matter any other binary crossover operator) is simple, their mathematical modeling becomes difficult. Any convergence proof of a stochastic process involves calculation of transition probabilities. Since in the SBX operator, a direct probability distribution of children points for any given pair of parent points is used, the proof of convergence of genetic algorithms may become easier.

The success of binary GAs in many single-objective, unimodal problems has led researchers to extend GA's applicability to other types of optimization problems—multimodal and multiobjective function optimization problems. In order to find multiple optimal solutions simultaneously, *sharing* functions are used to create artificial fitness values, which are used in the

reproduction operator (Deb, 1989; Goldberg and Richardson, 1987). In order to find multiple Pareto optimal points for multiobjective function optimization, the concept of *nondomination* is used to create artificial fitness values, which are again used in the reproduction operator (Srinivas and Deb, in press). Since only reproduction operator needs modification to solve the above problems, similar reproduction techniques can be used along with the SBX operator used in this study to investigate the efficacy of real-coded GAs in multimodal and multiobjective problems defined on a continuous search space.

7 Conclusions

In this paper, a real-coded crossover operator has been developed based on the search characteristics of a single-point crossover used in binary-coded GAs. In order to define the search power of a crossover operator, a spread factor has been introduced as the ratio of the absolute differences of the children points to that of the parent points. Thereafter, the probability of creating a child point for two given parent points has been derived for the single-point crossover. Motivated by the success of binary-coded GAs in problems with discrete search space, simulated binary crossover (SBX) operator has been developed to solve problems having continuous search space. The SBX operator has similar search power as that of the single-point crossover.

On a number of test functions including De Jong's five test functions, it has been found that real-coded GAs with the SBX operator can overcome a number of difficulties inherent with binary-coded GAs in solving continuous search space problems—Hamming cliff problem, arbitrary precision problem, and fixed mapped coding problem. In the comparison of real-coded GAs with SBX operator and binary-coded GAs with single-point crossover operator, it has been observed that the performance of the former is better than the latter on continuous functions and the performance of the former is similar to the latter in solving discrete and difficult functions. In comparison with another real-coded crossover operator (BLX-0.5) suggested elsewhere, SBX performs better in difficult test functions. It has also been observed that SBX is particularly useful in problems where the bounds of the optimum point is not known a priori and where there are multiple optima, of which one is global.

Real-coded GAs with the SBX operator have been also attempted to solve a two-variable blocked function (the concept of blocked functions were introduced by Goldberg (1991)). Blocked functions are difficult for real-coded GAs, because local optimal points block the progress of search to continue towards the global optimal point. The simulation results on the two-variable blocked function have shown that in most occasions, the search proceeds the way as predicted by Goldberg. Most importantly, it has been observed that the real-coded GAs with SBX works similar to that of the binary-coded GAs with single-point crossover in overcoming the barrier of the local peaks and converge to the global basin. However, it is premature to conclude whether real-coded GAs with SBX operator can overcome the local barriers in higher-dimensional blocked functions.

These results are encouraging and suggest avenues for further research. Because the SBX operator uses a probability distribution for choosing a child point, the real-coded GAs with

SBX are one step ahead of the binary-coded GAs in terms of achieving a convergence proof for GAs. With a direct probabilistic relationship between children and parent points used in this paper, cues from the classical stochastic optimization methods can be borrowed to achieve a convergence proof of GAs, or a much closer tie between the classical optimization methods and genetic algorithms is on the horizon.

Acknowledgment

This work is funded by the Department of Science and Technology, New Delhi under grant SR/SY/E-06/93.

References

Aarts, E. and Korst, J. (1989). *Simulated annealing and Boltzmann machines: A stochastic approach to combinatorial optimization and neural computing.* Chichester: Wiley.

Back, T., Hoffmeister, F., and Schwefel, H-P. (1991). A survey of evolution strategies. In R. K. Belew and L. B. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp. 2–9).

Booker, L. B. (1993). Recombination distribution for GAs. In D. Whitley (Ed.), *Foundations of Genetic Algorithms* (pp. 29–44).

Davis, L. (1991). Bit-climbing, representational bias, and test suite design. In R. K. Belew, & L. B. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp. 18–23).

De Jong, K. A. (1975). An analysis of the behavior of a class of genetic adaptive systems (Doctoral dissertation, University of Michigan, Ann Arbor). *Dissertation Abstracts International, 36*(10), 5140B. (University Microfilms No. 76-9381)

Deb, K. (1989). Genetic algorithms in multimodal function optimization, *Master's Thesis*, (TCGA Report No. 89002). Tuscaloosa: University of Alabama.

Eshelman, L. J. and Schaffer, J. D. (1993). Real-coded genetic algorithms and interval schemata. In D. Whitley (Ed.), *Foundations of Genetic Algorithms, II* (pp. 187–202).

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning.* Reading: Addison-Wesley.

Goldberg, D. E. (1991). Real-coded genetic algorithms, virtual alphabets, and blocking. *Complex Systems, 5*(2), 139–168. (Also IlliGAL Report 90001)

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., Deb. K., Kargupta, H, and Harik, G. (1993). Rapid, accurate optimization of difficult problems using messy genetic algorithms. In S. Forrest (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms* (pp. 56–64).

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

Goldberg, D. E., and Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In J. J. Grefenstette (Ed.), *Proceedings of the Second International Conference on Genetic Algorithms* (pp. 41–49).

Gordon, V. S. and Whitley, D. (1993). Serial and parallel genetic algorithms as function optimizers. In S. Forrest (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms* (pp. 177–183).

Kargupta, H., Deb, K., and Goldberg, D. E. (1992). Ordering genetic algorithms and deception. In R. Manner and B. Manderick (Eds.), *Parallel Problem Solving from Nature II* (pp 47–56).

Muhlenbein, H., Schomisch, M., and Born, J. (1991). The parallel genetic algorithms as function optimizer. In R. K. Belew and L. B. Booker (Eds.) *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp. 271–278).

Radcliffe, N. J. (1991). Forma analysis and random respectful recombination. In R. K. Belew and L. B. Booker (Eds.) *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp. 222–229).

Radcliffe, N. J. (1993). Genetic set recombination. In D. Whitley (Ed.) *Foundations of Genetic Algorithms II* (pp. 203–219).

Rechenberg, I. (1973). *Evolutionstrategie: Optmierung technisher Systeme nach Prinzipien der biologischen Evolution.* Stuttgurt: Frommann-Holzboog Verlag.

Srinivas, N. and Deb, K. (in press). Multiobjective function optimization using nondominated sorting genetic algorithms, *Evolutionary Computation*, MIT Press.

Syswerda, G. (1989). Uniform crossover in genetic algorithms. In J. D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 2–9).

Wright, A. (1991). Genetic algorithms for real parameter optimization. In G. J. E. Rawlins (Ed.), *Foundations of Genetic Algorithms.* (pp.205–220).