Chapter from the book *Bio-Inspired Computational Algorithms and Their Applications*
Downloaded from: http://www.intechopen.com/books/bio-inspired-computational-algorithms-and-their-applications

Interested in publishing with InTechOpen?
Contact us at book.department@intechopen.com
The Roles of Crossover and Mutation in Real-Coded Genetic Algorithms

Yourim Yoon¹ and Yong-Hyuk Kim²

¹School of Computer Science and Engineering, Seoul National University, Seoul
²Department of Computer Science and Engineering, Kwangwoon University, Seoul
Republic of Korea

1. Introduction

We recognized that the roles of crossover and mutation in real encoding are quite different from those in binary encoding during performing previous work with real-coded genetic algorithms (Yoon et al., 2012). In this study, we are to argue the distinct roles of genetic operators in real encodings.

Recently many studies on evolutionary algorithms using real encoding have been done. They include ant colony optimization (Socha & Dorigo, 2008), artificial bee colony algorithm (Akay & Karaboga, 2010; Kang et al., 2011), evolution strategies (ES) (Beyer, 2001), differential evolution (Das & Suganthan, 2011; Dasgupta et al., 2009; Kukkonen & Lampinen, 2004; 2005; Mezura-Montes et al., 2010; Noman & Iba, 2005; Rönkkönen et al., 2005; Storn & Price, 1997; Zhang et al., 2008), particle swarm optimization (Chen et al., 2007; Huang et al., 2010; Juang et al., 2011; Krohling & Coelho, 2006; l. Sun et al., 2011), and so on. In particular, in the field of ES, we can find many studies based on self-adaptive techniques (Beyer & Deb, 2001; Hansen & Ostermeier, 2001; Igel et al., 2007; 2006; Jägersküpper, 2007; Kita, 2001; Kramer, 2008a;b; Kramer et al., 2007; Meyer-Nieberg & Beyer, 2007; Wei et al., 2011).

Many researchers have also concentrated on using real-valued genes in genetic algorithms (GAs), as in (Ripon et al., 2007). It is reported that, for some problems, real-coded representation and associated techniques outperform conventional binary representation (Eshelman & Schaffer, 1993; Herrera et al., 1998; Janikow & Michalewicz, 1991; Lozano et al., 2004; Ono et al., 1999; Ono & Kobayashi, 1997; Surry & Radcliffe, 1996; Wright, 1991). Several theoretical studies of real-coded GAs have also been performed (Goldberg, 1991; Higuchi et al., 2000; Kita et al., 1998; Qi & Palmieri, 1994a;b). However, the role and behavior of genetic operators in real-coded GAs are fundamentally different from those in binary encodings although motivation of the operators and the framework of GAs are similar.

In this chapter, we try to verify different properties of crossover and mutation in real encodings from those in binary encodings through various experiments. We especially concentrate on the effect of genetic operators (the bias and functions of crossover and mutation) when they are used in real-coded GAs.

*Corresponding author: Yong-Hyuk Kim
The remainder of this chapter is organized as follows. Traditional and recent genetic operators in real encoding are introduced in Section 2. Previous genetic operators are presented in Section 2.1 and ones we used in real encoding in this study are described in Section 2.2. In Section 3, we describe the concept of bias of genetic operators and analyze that in the case of crossover and mutation for GAs. In Section 4, experimental results for various combinations of crossover and mutation are provided and analyzed. Finally, we make conclusions in Section 5.

2. Genetic operators in real encoding

2.1 Previous operators

The roles of crossover and mutation may change according to the selection of the operators. We reviewed the most frequently used crossover and mutation operators for real-code representation. We are to analyze how the roles of crossover and mutation can change by studying various combinations of crossover and mutation operators.

In literature many crossover operators for real-code representation are found. Traditional crossover operators for the real-code representation are described in (Bäch et al., 2000). The two main families of traditional crossover operators (Mühlenbein & Schlierkamp-Voosen, 1993) are discrete crossovers\(^1\) (Reed et al., 1967) and blend crossovers (Michalewicz, 1996). Blend crossover operators can be distinguished into line crossovers and box crossovers. Important variations of the last two crossover operators are the extended-line crossover and the extended-box crossover (Mühlenbein, 1994).

The discrete recombination family is the straightforward extension to real vectors of the family of mask-based crossover operators for binary strings including \(n\)-point and uniform crossover.

\(^1\) It is also called dominant crossover.
The Roles of Crossover and Mutation in Real-Coded Genetic Algorithms

The Roles of Crossover and Mutation in Real-Coded Genetic Algorithms

box-crossover\((x, y)\)

\{
    \text{for} \ i \leftarrow 1 \ \text{to} \ n
    \ z_i \leftarrow \text{a random real number in} \ [\min(x_i, y_i), \max(x_i, y_i)]
    \ \text{return} \ z = (z_1, z_2, \ldots, z_n)
\}

Fig. 2. Pseudo-code of box crossover

extended-box-crossover\((x, y)\)

\{
    \text{for} \ i \leftarrow 1 \ \text{to} \ n
    \ m \leftarrow \min(x_i, y_i), \ M \leftarrow \max(x_i, y_i)
    \ em \leftarrow m - a(M - m), \ eM \leftarrow M + a(M - m)
    \ z_i \leftarrow \text{a random real number in} \ [\min(em, l_i), \max(eM, u_i)]
    \ \text{return} \ z = (z_1, z_2, \ldots, z_n)
\}

\// \ a \text{ is extension rate.}

Fig. 3. Pseudo-code of extended-box crossover

The mask is still a binary vector dictating for each position of the offspring vector from which parent the (real) value for that position is taken.

The blend recombination family does not exchange values between parents like discrete recombinations but it averages or blends them. Line recombination returns offspring on the (Euclidean) line segment connecting the two parents. Box recombination returns offspring in the box (hyper-rectangle) whose diagonally opposite corners are the parents. Extended-line recombination picks offspring on an extended segment passing through the parent vectors but extending beyond them and not only in the section between them. Analogously extended-box recombination picks offspring on an extended box whose main diagonal passes through the parents but extends beyond them.

Recently several new crossovers for the real-coded representation have been designed. Several non-traditional crossover operators for real-coded representation are found in the recent literature. They include SBX (simulated binary crossover) (Ballester & Carter, 2003; 2004b; Deb & Agrawal, 1995; Deb & Beyer, 1999; Deb & Kumar, 1995; Deb et al., 2007), UNDX (unimodal normal distribution crossover) (Kita et al., 1998; 1999; Ono et al., 1999; Ono & Kobayashi, 1997), SPX (simplex crossover) (Higuchi et al., 2000; Tsutsui & Goldberg, 2002; Tsutsui et al., 2001; 1999), PCX (parent-centric crossover) (Ballester & Carter, 2004a; Deb et al., 2002), etc (Herrera et al., 2003; 2005; Takahashi & Kita, 2001). Most of them are complex and based on the specific probability distribution of the offspring (SBX, UNDX, and PCX), self-adaptivity (SBX and UNDX), or multiple parents (UNDX and SPX). Some of them, e.g., include the function of mutation operators. In this study, we focus on traditional crossover that does not consider the specific probability distribution of the offspring but only what offspring can be generated with a probability greater than zero, given the two parents.
The most common form of mutation for real-code vectors generates an offspring vector by adding a vector \( M \) of random variables with expectation zero to the parent vector. There are two types of mutations bounded and unbounded depending on the fact that the range of the random variable is bounded or unbounded. The most frequently used bounded mutations are the creep mutation and the single-variable mutation and for the unbounded case is the Gaussian mutation. For the creep (or hyper-box) mutation \( M \sim U([-a, a]^n) \) is a vector of uniform random variables, where \( a \) is a parameter defining the limits of the offspring area. This operator yields offspring within a hyper-box centered in the parent vector. For the single-variable mutation \( M \) is a vector in which all entries are set to zero except for a random entry which is a uniform random variable \( \sim U([-a,a]) \). Bounded mutation operators may get stuck in local optima. In contrast, unbounded mutation operators guarantee asymptotic global convergence. The primary unbounded mutation is the Gaussian mutation for which \( M \) is a multivariate Gaussian distribution.

### 2.2 Adopted operators for this study

As crossover operators for our analysis, we adopted four representative crossovers: box, extended-box, line, and extended-line crossovers. Their pseudo-codes are shown in Figures 2, 3, 4, and 5, respectively and the possible range for each crossover is represented in Figure 1.
mutation($z, p$)
{
    for $i \leftarrow 1$ to $n$
        if a random number from $[0, 1]$ is less than mutation rate $p$
            $z_i \leftarrow z_i + N(0, (u_i - l_i)/10)$;
    return $z = (z_1, z_2, \ldots, z_n)$;
}

Fig. 6. Pseudo-code of mutation

fine-mutation($x, y, z, p$)
{
    for $i \leftarrow 1$ to $n$
        if a random number from $[0, 1]$ is less than mutation rate $p$
            $z_i \leftarrow z_i + N(0, |x_i - y_i|)$;
    return $z = (z_1, z_2, \ldots, z_n)$;
}

Fig. 7. Pseudo-code of fine mutation

And, as mutation operators for our analysis, we adopted two kinds of mutation: Gaussian mutation and fine mutation. Their pseudo-codes are shown in Figures 6 and 7, respectively. The Gaussian mutation is a simple static Gaussian mutation, the same as in Tsutsui & Goldberg (2001). The $i$-th parameter $z_i$ of an individual is mutated by $z_i = z_i + N(0, \sigma_i)$ with a mutation rate $p$, where $N(0, \sigma_i)$ is an independent random Gaussian number with the mean of zero and the standard deviation of $\sigma_i$. In our study, $\sigma_i$ is fixed to $(u_i - l_i)/10$ - the tenth of width of given area. The fine mutation is a simple dynamic Gaussian mutation inspired from Ballester & Carter (2004b). In different with Gaussian mutation, it depends on the distance between parents and, as population converges, the strength of the mutation approaches zero.

3. Bias of genetic operators

Pre-existing crossovers for the real-coded representation have an inherent bias toward the center of the space. Some boundary extension techniques to reduce crossover bias have been extensively studied (Someya & Yamamura, 2005; Tsutsui, 1998; Tsutsui & Goldberg, 2001). The concept of crossover bias first appeared in (Eshelman et al., 1997) and it has been extensively used in (Someya & Yamamura, 2005; Tsutsui & Goldberg, 2001), in which they tried to remove the bias of real-coded crossover heuristically (and theoretically incompletely).

Notice that the notion of bias of a crossover operator has different definitions depending upon the underlying representation considered. The bias toward the center of the space considered in real-coded crossovers conceptually differs from the crossover biases on binary strings, which focus on how many bits are passed to the offspring and from which positions, which, in turn conceptually differs from the bias considered in Genetic Programming focusing on bloat.

The notion of bias so defined can be understood as being the inherent preference of a search operator for specific areas of the search space. This is an important search property of a search
operator: an evolutionary algorithm using that operator, without selection, is attracted to the areas the search operator prefers. Arguably, also when selection is present, the operator bias acts as a background force that makes the search keener to go toward the areas preferred by the search operator. This is not necessarily bad if the bias is toward the optimum or an area with high-quality solutions. However, it may negatively affect performance if the bias is toward an area of poor-quality solutions. If we do not know the spatial distribution of the fitness of the problem, we may prefer not to have any \textit{a priori} bias of the search operator, and instead use only the bias of selection, which is informed by the fitness of sampled solutions that constitute empirical knowledge about promising areas obtained in the search, and which is better understood.

In this chapter, we investigate the bias caused by crossover itself and crossover combined with mutation in real-coded GAs. Intuitively, box and line crossover are biased toward the center on the Euclidean space. This intuition is easy to verify experimentally by picking a large number of pairs (ideally infinitely many) of random parents and generating offspring uniformly at random in the boxes (or lines) identified by the pairs of parents.
In the Hamming space, the distribution of the offspring of uniform crossover tends in the limit to be uniform on all space, whereas in the Euclidean space the distribution of the offspring tends to be unevenly distributed on the search space and concentrates toward the center of the space. One way to compensate, but not eliminate, such bias is using extended-line and extended-box crossovers. Figure 8 visualizes the crossover bias in the one-dimensional real space by plotting frequency rates of $10^7$ offspring randomly generated by each type crossover. As we can see, box and line crossover are biased toward the center of the domain. We could also observe that extended-box and extended-line crossover largely reduce the bias but they are still biased toward the center.²

For analyzing the effect of mutation in relation with the bias, we also performed the same test using crossover combined with Gaussian mutation. We picked $10^7$ pairs of random parents, generated offspring randomly using each type crossover, and then applied Gaussian mutation. The tests are performed for various mutation rates from 0.0 to 1.0. The results for box, extended-box, line, and extended-line crossover are shown in Figures 9, 10, 11, and 12, respectively. Interestingly, for all cases, we could observe that the higher mutation rate reduces the bias more largely. However, even high mutation rates cannot eliminate the bias completely.

### 4. Combination of crossover and mutation

In this section, we try to figure out the properties of crossover and mutation through experiments using their various combinations. For our experiments, four test functions are chosen from Suganthan et al. (2005). They are described in Table 1.

We mainly followed the genetic framework by Tsutsui & Goldberg (2001). Its basic evolutionary model is quite similar to that of CHC (Eshelman, 1991) and $(μ + λ)$-ES (Beyer, 2001).

² We can find consistent results with this in Someya & Yamamura (2005); Yoon et al. (2012).
Fig. 9. Bias of box crossover with mutation
Fig. 10. Bias of extended-box crossover (α = 0.5) with mutation
(a) Crossover without mutation
(b) Crossover with mutation ($p = 0.05$)
(c) Crossover with mutation ($p = 0.1$)
(d) Crossover with mutation ($p = 0.2$)
(e) Crossover with mutation ($p = 0.5$)
(f) Crossover with mutation ($p = 1.0$)

Fig. 11. Bias of line crossover with mutation
Fig. 12. Bias of extended-line crossover ($\alpha = 0.5$) with mutation
Let $N$ the population size. A collection of $N/2$ pairs is randomly composed, and crossover and mutation are applied to each pair, generating $N/2$ offspring. Parents and newly generated offspring are ranked and the best $N$ individuals among them are selected for the population in the next generation. The population size was 400 for all experiments. If the population has no change during $n \times r \times (1.0 - r)$ generations, it is reinitialized except for the best individual. Here, $r$ is a divergence rate and we set it to 0.25 as in Eshelman (1991). The used GA terminates when it finds the global optimum.

For crossover, we used four crossover operators: box crossover, extended-box crossover (extension rate $\alpha$: 0.5), line crossover, and extended-line crossover (extension rate $\alpha$: 0.5). After crossover, we either mutate the offspring or do not. We used two different mutation operators; Gaussian mutation and fine mutation. Different mutation rates were applied to each crossover type and the rates decrease as the number of generations increases.

Table 2 shows the results from 30 runs. Each value in ‘Ave’ means the average function value from 30 runs. The smaller, the better. The limit of function evaluations is 50,000, i.e., the genetic algorithm terminates after 50,000 evaluations and outputs the best solution among evaluated ones so far over generations. In the table, $k = 1 + \lfloor \text{numberOfGenerations}/100 \rfloor$ and the rate of fine mutation is $0.5/k$.

From these experiments we can obtain the following properties.

- There is no superior operator combination for all over the problem instances. For the shifted sphere, box crossover with fine mutation showed the best performance. For the shifted Schwefel, line crossover with Gaussian mutation, for the shifted Rosenbrock, extended-box crossover without mutation, and for the shifted Rastrigin, box crossover with fine mutation showed the best performances, respectively. So we can know that suitable crossover and mutation can be varied depending on the property of given problem.
- Without mutation, extended-box crossover showed the best performance. That is, when we do not know the characteristic of given problem, it is a general choice that we use extended-box crossover as a crossover operator in real-coded genetic algorithms. It is convenient since parameter tuning with mutation is not required. However, it is possible to surpass the performance of extended-box crossover using well-designed combination of crossover and mutation.
- Unusually, for extended-box crossover, the results without mutation is the best and the performance becomes worse as mutation rate increases. However, for box crossover, moderate rate of mutation has a good effect to the performance. For all cases, box crossover with mutation showed better performance than that without mutation. From this fact, we can infer that extended-box crossover contains the function of mutation in itself but box crossover does not.
- Except for extended-box crossover, the results of crossover with mutation were better than those of crossover without mutation. In particular, fine mutation was better than Gaussian mutation. Fine mutation depends on the distance between parents so, as population converges, the strength of the mutation approaches zero. That is, the amount of mutation becomes very fine as population converges. In binary encodings, the main role of mutation is perturbation effect to prevent premature convergence. However, we can know that, in real encoding, the function of fine tuning by mutation is also important from this experiment.
### Table 2. Results

<table>
<thead>
<tr>
<th>Crossover</th>
<th>Function</th>
<th>Shifted Sphere Ave ($\sigma/\sqrt{n}$)</th>
<th>Shifted Schwefel Ave ($\sigma/\sqrt{n}$)</th>
<th>Shifted Rosenbrock Ave ($\sigma/\sqrt{n}$)</th>
<th>Shifted Rastrigin Ave ($\sigma/\sqrt{n}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td>5.46e+02 (4.59e+01)</td>
<td>1.68e+04 (9.05e+02)</td>
<td>1.87e+07 (1.70e+06)</td>
<td>2.66e+02 (1.91e+00)</td>
</tr>
<tr>
<td>0.05/k</td>
<td></td>
<td>6.40e+01 (2.17e+01)</td>
<td>1.16e+04 (5.33e+02)</td>
<td>4.64e+06 (3.86e+05)</td>
<td>2.68e+02 (1.65e+00)</td>
</tr>
<tr>
<td>0.10/k</td>
<td></td>
<td>6.37e+02 (1.55e+02)</td>
<td>7.99e+03 (4.79e+02)</td>
<td>4.74e+06 (2.34e+04)</td>
<td>2.64e+02 (1.58e+00)</td>
</tr>
<tr>
<td>0.20/k</td>
<td></td>
<td>9.87e+02 (3.13e+03)</td>
<td>7.87e+03 (3.18e+02)</td>
<td>1.89e+07 (8.19e+02)</td>
<td>2.27e+02 (1.28e+02)</td>
</tr>
<tr>
<td>0.50/k</td>
<td></td>
<td>2.38e+03 (6.45e+03)</td>
<td>1.24e+04 (3.76e+02)</td>
<td>7.12e+07 (2.45e+04)</td>
<td>1.91e+02 (1.72e+02)</td>
</tr>
<tr>
<td>1.00/k</td>
<td></td>
<td>2.46e+04 (4.46e+02)</td>
<td>2.47e+04 (3.03e+02)</td>
<td>5.36e+09 (1.69e+08)</td>
<td>1.83e+02 (1.64e+00)</td>
</tr>
<tr>
<td>Box</td>
<td>Fine + 0.05/k</td>
<td>-4.50e+02 (4.63e-03)</td>
<td>8.74e+03 (1.16e+02)</td>
<td>7.45e+02 (3.21e+01)</td>
<td>2.12e+02 (2.33e+00)</td>
</tr>
<tr>
<td></td>
<td>Fine + 0.10/k</td>
<td>-4.50e+02 (2.54e-02)</td>
<td>8.43e+03 (1.36e+02)</td>
<td>1.99e+03 (8.59e+01)</td>
<td>2.77e+02 (3.16e+00)</td>
</tr>
<tr>
<td></td>
<td>Fine + 0.20/k</td>
<td>-4.48e+02 (8.07e-02)</td>
<td>8.10e+03 (1.42e+02)</td>
<td>3.35e+03 (1.59e+02)</td>
<td>2.39e+02 (3.26e+00)</td>
</tr>
<tr>
<td></td>
<td>Fine + 0.50/k</td>
<td>-3.40e+02 (3.04e+00)</td>
<td>8.10e+03 (1.17e+02)</td>
<td>2.16e+05 (1.02e+04)</td>
<td>1.94e+02 (1.90e+00)</td>
</tr>
<tr>
<td></td>
<td>Fine + 1.00/k</td>
<td>-3.40e+02 (3.04e+00)</td>
<td>8.10e+03 (1.17e+02)</td>
<td>2.16e+05 (1.02e+04)</td>
<td>1.94e+02 (1.90e+00)</td>
</tr>
<tr>
<td>Extended</td>
<td>Fine + 0.05/k</td>
<td>-4.50e+02 (2.72e-04)</td>
<td>8.78e+03 (1.90e+02)</td>
<td>7.45e+02 (3.21e+01)</td>
<td>2.12e+02 (2.33e+00)</td>
</tr>
<tr>
<td>box</td>
<td>Fine + 0.10/k</td>
<td>-4.49e+02 (3.17e+00)</td>
<td>9.63e+03 (2.20e+02)</td>
<td>3.14e+03 (1.67e+02)</td>
<td>2.00e+02 (1.97e+00)</td>
</tr>
<tr>
<td></td>
<td>Fine + 0.20/k</td>
<td>-4.40e+02 (2.52e-01)</td>
<td>1.07e+04 (2.50e+02)</td>
<td>2.24e+04 (9.14e+01)</td>
<td>1.95e+02 (1.75e+00)</td>
</tr>
<tr>
<td></td>
<td>Fine + 0.50/k</td>
<td>-3.59e+02 (2.22e+00)</td>
<td>1.25e+04 (2.45e+02)</td>
<td>3.35e+05 (1.86e+04)</td>
<td>1.81e+02 (1.66e+00)</td>
</tr>
<tr>
<td></td>
<td>Fine + 1.00/k</td>
<td>-3.54e+02 (2.03e+00)</td>
<td>1.60e+04 (3.92e+02)</td>
<td>6.93e+06 (3.35e+05)</td>
<td>1.63e+02 (1.66e+00)</td>
</tr>
<tr>
<td>Line</td>
<td>Fine + 0.05/k</td>
<td>-4.23e+02 (7.81e+01)</td>
<td>1.76e+04 (3.62e+02)</td>
<td>2.07e+05 (1.21e+04)</td>
<td>1.75e+02 (1.78e+00)</td>
</tr>
<tr>
<td>crossover</td>
<td>Fine + 0.10/k</td>
<td>-3.67e+02 (2.09e+00)</td>
<td>1.80e+04 (4.84e+02)</td>
<td>7.16e+05 (2.99e+04)</td>
<td>1.68e+02 (1.50e+00)</td>
</tr>
<tr>
<td></td>
<td>Fine + 0.20/k</td>
<td>-2.72e+02 (3.48e+00)</td>
<td>1.85e+04 (4.39e+02)</td>
<td>1.55e+06 (6.25e+04)</td>
<td>1.68e+02 (1.83e+00)</td>
</tr>
<tr>
<td></td>
<td>Fine + 0.50/k</td>
<td>-2.18e+01 (1.03e+01)</td>
<td>1.84e+04 (5.26e+02)</td>
<td>5.45e+06 (2.49e+05)</td>
<td>1.63e+02 (2.22e+00)</td>
</tr>
<tr>
<td></td>
<td>Fine + 1.00/k</td>
<td>-9.07e+02 (3.03e+00)</td>
<td>2.24e+04 (5.74e+02)</td>
<td>2.69e+07 (1.32e+06)</td>
<td>1.52e+02 (1.92e+00)</td>
</tr>
<tr>
<td></td>
<td>Fine + 0.05/k</td>
<td>2.47e+03 (3.17e+00)</td>
<td>2.10e+04 (4.98e+02)</td>
<td>4.65e+07 (1.85e+06)</td>
<td>1.50e+02 (2.26e+00)</td>
</tr>
<tr>
<td></td>
<td>Fine + 0.10/k</td>
<td>2.47e+03 (3.17e+00)</td>
<td>2.10e+04 (4.98e+02)</td>
<td>4.65e+07 (1.85e+06)</td>
<td>1.50e+02 (2.26e+00)</td>
</tr>
<tr>
<td></td>
<td>Fine + 0.20/k</td>
<td>2.47e+03 (3.17e+00)</td>
<td>2.10e+04 (4.98e+02)</td>
<td>4.65e+07 (1.85e+06)</td>
<td>1.50e+02 (2.26e+00)</td>
</tr>
<tr>
<td></td>
<td>Fine + 0.50/k</td>
<td>2.47e+03 (3.17e+00)</td>
<td>2.10e+04 (4.98e+02)</td>
<td>4.65e+07 (1.85e+06)</td>
<td>1.50e+02 (2.26e+00)</td>
</tr>
<tr>
<td></td>
<td>Fine + 1.00/k</td>
<td>2.47e+03 (3.17e+00)</td>
<td>2.10e+04 (4.98e+02)</td>
<td>4.65e+07 (1.85e+06)</td>
<td>1.50e+02 (2.26e+00)</td>
</tr>
</tbody>
</table>

The Roles of Crossover and Mutation in Real-Coded Genetic Algorithms
5. Conclusions

In this chapter, we tried to analyze distinct roles of crossover and mutation when using real encoding in genetic algorithms. We investigated the bias of crossover and mutation. From this investigation, we could know that extended crossover and mutation can reduce the inherent bias of traditional crossover in real-coded genetic algorithms.

We also studied the functions of crossover and mutation operators through experiments for various combinations of both operators. From these experiments, we could know that extended-box crossover is good in the case of using only crossover without mutation. However, it is possible to surpass the performance of extended-box crossover using well-designed combination of crossover and mutation. In the case of other crossover operators, not only the function of perturbation but also that of fine tuning by mutation is important, but extended-box crossover contains the fine tuning function in itself.

There are many other test functions defined on real domains. We conducted experiments with limited test functions. We may obtain more reliable conclusions through experiments with more other functions. So, more extended experiments on more various test functions are needed for future work. We may also find other useful properties from those empirical study.

6. Acknowledgments

The authors would like to thank Dr. Alberto Moraglio for his encouragement and valuable comments in improving this study. This work was supported by the Research and Development of Advanced Weather Technology of National Institute of Meteorological Research (NIMR) of Korea in 2011.

7. References


Bio-inspired computational algorithms are always hot research topics in artificial intelligence communities. Biology is a bewildering source of inspiration for the design of intelligent artifacts that are capable of efficient and autonomous operation in unknown and changing environments. It is difficult to resist the fascination of creating artifacts that display elements of lifelike intelligence, thus needing techniques for control, optimization, prediction, security, design, and so on. Bio-Inspired Computational Algorithms and Their Applications is a compendium that addresses this need. It integrates contrasting techniques of genetic algorithms, artificial immune systems, particle swarm optimization, and hybrid models to solve many real-world problems. The works presented in this book give insights into the creation of innovative improvements over algorithm performance, potential applications on various practical tasks, and combination of different techniques. The book provides a reference to researchers, practitioners, and students in both artificial intelligence and engineering communities, forming a foundation for the development of the field.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following: