# Improvement of Real-coded Genetic Algorithms for Optimization Problems

Hiroshi Kinjo, Hiroki Nakanishi and Tetsuhiko Yamamoto
kinjo,hiroki,yamamoto@mibai.tec.u-ryukyu.ac.jp
Faculty of Engineering, University of the Ryukyus
Nishihara, Okinawa 903-0213, Japan

Duong Sam Chau
Faculty of Electric Mechanics
Hanoi Agricultural University
Gialam, Hanoi, Vietnam

## Abstract

Genetic algorithms (GAs) are widely used in solving optimization problems. In this paper, we present a crossover method used in real-coded GAs for multivariable optimization problems. A well-known crossover operator of real-coded GAs is the blend crossover (BLX). The BLX has a range parameter that determines the offspring production range. The search performance of the GA depends on the value of the range parameter. However, determination of the range parameter is sometimes difficult. In this paper, we propose a crossover operator for real-coded GAs that is a close-to-parent offspring. The crossover is based on the idea that offspring should be close to their parents. In order to improve the evolution performance, we applied a mutation operator to the real-coded GA. Simulation shows that the use of a close-to-parent crossover with the mutation operator effectively improves search performance.

**Keywords:** Real-coded GA, Crossover method, Close-to-parent offspring, Optimization problem.

## 1 Introduction

Recently, genetic algorithms (GAs) have been applied widely and effectively in various fields [1]-[3]. The most useful property of GAs is their ability to solve search and optimization problems with very little required information about the problems. The performance of GAs depends, to a great extent, on the performance of the crossover operator used. The crossover operation is performed upon the selected chromosome. The crossover operates on two chromosomes at a time and generates offspring by combining the features of both chromosomes. A chromosome is usually a binary bit string, but not necessarily. There are several different variants of basic GAs. It is possible to use real-coded (or floating-point) genes and actually, several methods have been presented that use real-coded genes [4]. The crossover operators of the real-coded GA called the blend crossover (BLX-$\alpha$). The BLX has a range parameter that determines the production ranges of the offspring. The evolution performance of GAs is dependent on the value of the range parameter. Selection the range parameter is determined by trial and error. However, the determination of the range parameter is sometimes difficult.

The BLX is an offspring production method using a random number of uniform distributions based on the intervals of two parents and a range parameter. That is, the method is not based on the parents them selves but on the intervals between parents. We consider that offspring production should be based on the parents them selves. Based on the above-mentioned idea, in this paper, we present a new method, i.e., a close-to-parent crossover. Furthermore we applied a mutation operator to the crossover system to improve the search performance for optimization problems.

In section 2, we describe the close-to-parent offspring production method. In section 3, we describe the search performance of the crossover for three two-variable functions. In section 4, we describe the evolution performance for a neural network training problem. In section 5, we conclude this paper.

## 2 Close-to-parent offspring

Figure 1 (a) shows, a conventional crossover, the BLX. In the figure, offspring are produced using random values on a uniform distribution in the range of $[p_1 - \alpha I, p_2 + \alpha I]$, where $p_1$ and $p_2$ are the real values of the parents and $\alpha$ is the range parameter. $\alpha$ determines the search area of the offspring.

Figure 1(b) shows, the proposed crossover, that is, the close-to-parent offspring (CPO). CPO also has a range parameter $\alpha$ that determines the offspring production range based on each parent. For producing offspring using CPO, two individuals are produced using random values that have uniform distributions in

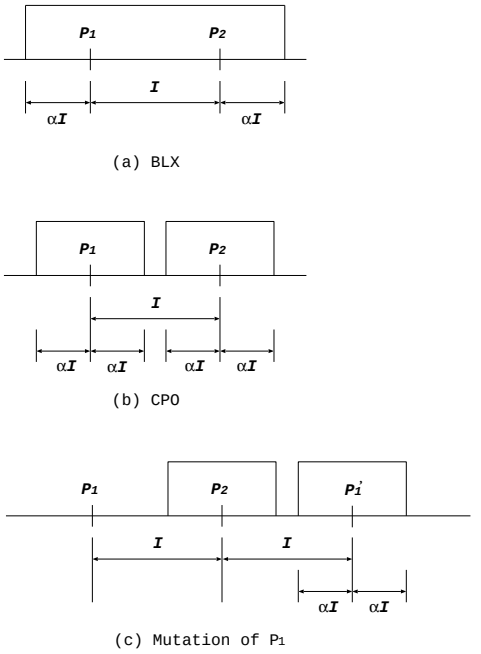



Fig. 1 Offspring production methods

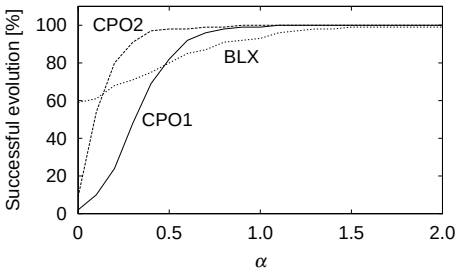



Fig. 2 Evolution performance of Sphere function

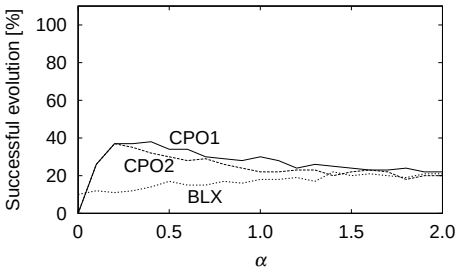



Fig. 3 Evolution performance for Rosenbrock function

the ranges of $[p_1 - \alpha I,\, p_1 + \alpha I]$ and $[p_2 - \alpha I,\, p_2 + \alpha I]$.

Figure 1 (c) shows a mutation of the CPO. In the figure, $p_1'$ denotes the center of the range of the mutation of parent $p_1$. The range of the offspring applied to the mutation is $[p_1' - \alpha I,\, p_1' + \alpha I]$.

## 3 Two-variable optimization problem

In this section we investigate the search performance of the proposed method for optimization problems for three two-variable functions: Sphere function, Rosenbrock function, and Rastrigin function [5].

The search performance is measured using successful evolution rates obtained from the minimum values of the functions in the GA.

In this test, we use the following GA parameters: population size is $N_p = 100$ and generation number is limited to 300. The produced offspring is 60% of the population $N_p$. The parents selection is the roulette wheel selection.

The sphere function, which is the simplest case in this test, is described by the following equation.

$$f(x, y) = x^2 + y^2, \quad x, y \in [-1.5, 1.5] \tag{1}$$

The minimum value of this function is 0.0 and occurs when $(x, y) = (0, 0)$.

Figure 2 shows the evolution performance of the Sphere function. In the figure, the CPO1 line shows the result of the cross-to-parent offspring and the CPO2 line shows the results of CPO with mutation, where the mutation rate is 20 %. We define a successful evolution as that having a minimum value of the function $f(x, y) < 0.001$. The lines of successful rate in the figure show the mean value of 1000 trials. We can see that the range of $\alpha$ in CPO2 which has a good performance is wider than that in BLX.

Figure 3 shows the evolution performance for the Rosenbrock function. The Rosenbrock function is described by the following equation.

$$f(x, y) = 100(x - y^2)^2 + (y - 1)^2, \quad x, y \in [-0.25, 1.25] \tag{2}$$

The minimum value of this function is 0.0 and occurs when $(x, y) = (0, 0)$. We can see that the rate of successful evolution of CPO1 is approximately two times better than that of BLX at $\alpha = 0.3$. From the figure, the result of CPO2, i.e., with the mutation operator, is poor compared with the result obtained when mutation operator is not used. In this case the mutation does not have a good effect on search performance.

Figure 4 shows the evolution performance for the Rastrigin function. The Rastrigin function, which is the most complicated function in this research, is described by the following equation.

$$f(x, y) = 20 + \{x^2 - 10\cos(2\pi x)\} + \{y^2 - 10\cos(2\pi y)\}, \quad x, y \in [-0.25, 1.25] \tag{3}$$

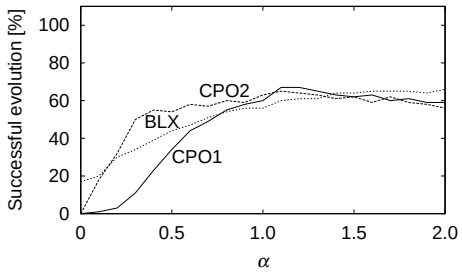



Fig. 4 Evolution performance for Rastrigin function

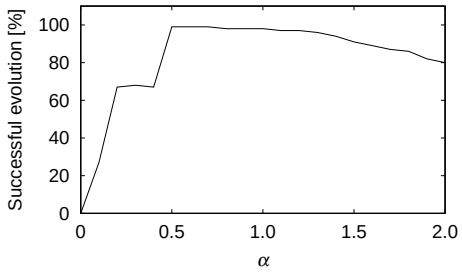



Fig. 5 Evolution performance for Rastrigin function using CPO with sign mutation

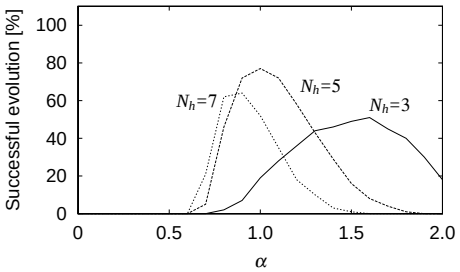
The minimum value of this function is 0.0 and occurs when $(x, y) = (0, 0)$. We can observe that the COP2 has a better search performance than BLX and CPO1. In the case of the Rastrigin function, it is difficult to search the solution because the surface of the function is in the shape of valleys and peaks [5]. We considered that a mutation more effective and severer than the range mutation described in Fig. 1 (c) is required. Figure 5 shows a result of CPO with a sign mutation. The sign mutation means a change in the sign of the real value of the offspring. This figure shows the result when the mutation rate is 20 %. We can see that CPO with a sign mutation has a better evolution performance than the previous results of the three methods shown in Fig. 4.

## 4 Neural network training problem

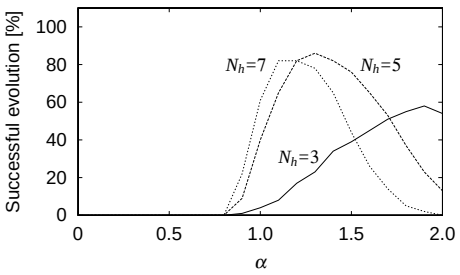
This section presents simulation results of neural network training problems. The training of neural network is known as a nonlinear multivariable optimization problem.

In order to obtain results comparable to those of the BLX, We should select the well-known exclusive-or (XOR) problem.

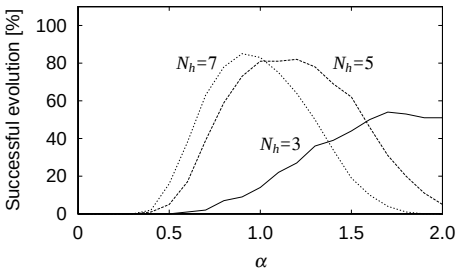
In this test, we use GAs for training a XOR network. For clearer results, we examine the XOR networks for various population sizes of GAs: $N_p = 25, N_p = 50$ and $N_p = 100$. In each case of $N_p$, we also change the number of neurons in the hidden layer of the network: $N_h = 3, N_h = 5$ and $N_h = 7$. The performance results of the XOR network training are shown in Figs. 6-8; in this training we use GAs with crossover operators: BLX, CPO1 and CPO2. We can see that, in all of the cases, CPO achieved better performances than BLX could. Moreover, in CPO, the range of $\alpha$, which has a good performance, is significantly wider than that in BLX. That is, CPO could obtain good results several times better than those of BLX. We can also observe that the addition of mutation effects the good results of the neural network training problem.




(a) BLX

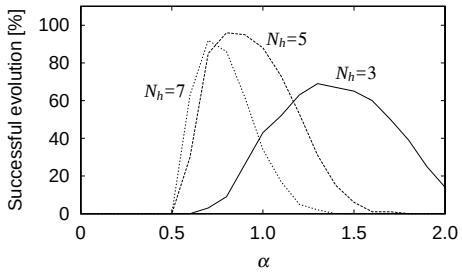



(b) CPO1

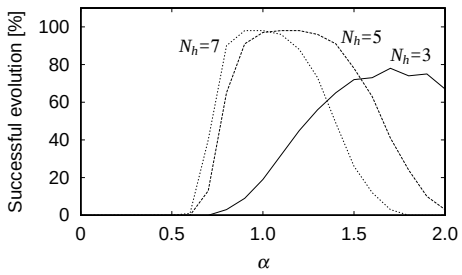



(c) CPO2

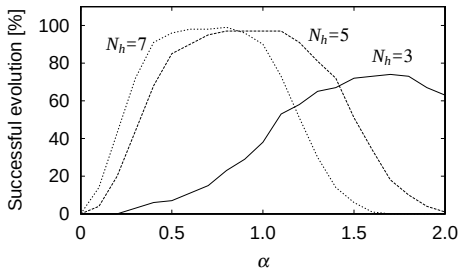
Fig. 6 Evolution performance for neural network ($N_p = 25$)




(a) BLX

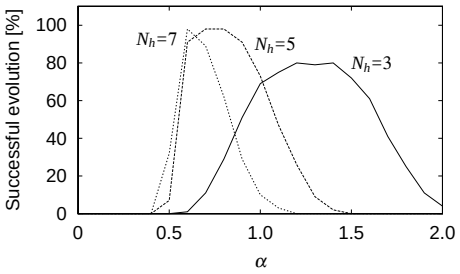



(b) CPO1

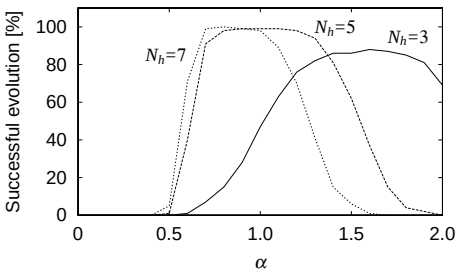



(c) CPO2

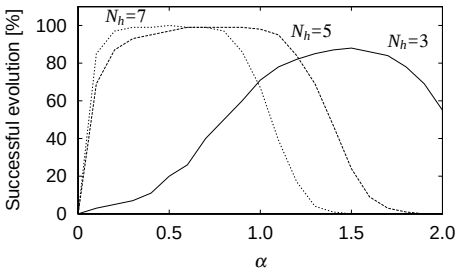
Fig. 7 Evolution performance for neural network ($N_p = 50$)




(a) BLX




(b) CPO1




(c) CPO2

Fig. 8 Evolution performance for neural network ($N_p = 100$)

## 5 Summary

Improving crossover operator in GAs is an area of active research for developing GAs. In this paper, we presented a new method of improving real-coded GAs by examining new search spaces of the crossover operator. The simulations show that GAs can achieve a good performance by changing search spaces to generate two close-to-parent offspring.

## References

[1] D. E. Goldberg, *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison-Wesley, 1989.

[2] L. Davis, *Hand Book of Genetic Algorithms*, Van Nostrand Reinhold, 1991.

[3] M. Gen and R. Cheng, *Genetic Algorithm and Engineering Design*, Wiley-Interscience, 1997.

[4] L. J. Eshelman and J. D. Schaffer, "Real-Coded Genetic Algorithms and Interval-Schemata" *Foundations of Genetic Algorithms 2*, Edited by L. Darrell Whitley, Morgan Kaufmann Publishers, 1993.

[5] H. Okamura, K. Kitasuka and T. Dohi, "Performance Evaluation of Genetic Algorithms Based on Markovian Analysis - Evaluation by Relaxation Time and Application to Real-Coded Genetic Algorithms," Transactions of the ISCIE, Vol. 16, No. 7, pp. 303-312, 2003. (in Japanese)