Evolving Search Spaces to Emphasize the Performance Difference of Real-Coded Crossovers Using Genetic Programming

Shinichi Shirakawa, *Member, IEEE*, Noriko Yata, and Tomoharu Nagao, *Member, IEEE*

***Abstract*— When we evaluate the search performance of an evolutionary computation (EC) technique, we usually apply it to typical benchmark functions and evaluate its performance in comparison to other techniques. In experiments on limited benchmark functions, it can be difficult to understand the features of each technique. In this paper, the search spaces that emphasize the performance difference of EC techniques are evolved by Cartesian genetic programming. We focus on a real-coded genetic algorithm, which is a type of genetic algorithm that has a real-valued vector as a chromosome. In particular, we generate search spaces using the performance difference of real-coded crossovers. In the experiments, we evolve the search spaces using the combination of three types of real-coded crossovers. As a result of our experiments, the search spaces that exhibit the largest performance difference of two crossovers are generated for all the combinations.**

I. Introduction

Numerous evolutionary computation (EC) techniques and related improvements, showing their effectiveness in various problem domains, have been proposed. Especially, various approaches have attempted to solve function-optimization problems that search minimum or maximum values of real-valued function. Typical techniques to solve EC function-optimization problems include real-coded genetic algorithms (RCGAs) [1], [2], [3], [4], [5], a covariance matrix adaptation evolution strategy (CMA-ES) [6], particle swarm optimization (PSO) [7], and differential evolution (DE) [8]. When we evaluate the search performance of an EC technique, we usually apply the techniques to the typical benchmark functions and evaluate the performance of the techniques in comparison to other techniques. When there are only slight differences in the performance of the techniques in regard to benchmark functions, it is difficult to understand the unique features of each technique. In addition, in this case, the performance relative to other benchmark functions and real-world problems is not clear. Research exists that theoretically analyzes the behavior of EC techniques. However, it is difficult to analyze and understand the theory of complex EC techniques.

In this situation, there are several research studies that generate a search space that increases the performance difference of two EC techniques. Oltean succeeded in evolving one-dimensional real-valued search problems and binary-valued search problems with a large performance difference of two simple (1 + 1) ES in which settings are different [9]. In [10], [11], Langdon and Poli evolved new problems that maximize the difference in performance between two optimizers (e.g., PSO, DE, CMA-ES, and hill climbing) using genetic programming (GP) [12], [13]. In this research, the performance difference between the two optimizers is assumed to be a fitness function of GP, and search spaces represented by combining the operators $\{+, -, \times, \div\}$ are evolved. As a result, they have generated the search problems to which a certain optimizer is superior (or inferior) to other optimizers. This approach to generate search problems is useful for the analysis and understanding of search techniques.

From the standpoint of evolving problems, Hemert evolved combinatorial problem instances that are difficult to solve [14]. The technique is applied to three combinatorial optimizations: binary constraint satisfaction, Boolean satisfiability, and the travelling salesman problem. This technique successfully evolved difficult problems.

In this paper, we focus on a real-coded genetic algorithm (RCGA), which is a type of GA that has a real-valued vector as a chromosome. In particular, we generate search spaces using the performance differences of real-coded crossovers. Various real-coded crossovers have been proposed that have unique characteristics. In our experiments, we use Cartesian genetic programming (CGP) [15], [16], which is a graph-based GP technique. In addition, we extend the objective functions to two of the performance differences and a number of active nodes in CGP and attempt to generate multiple search spaces with an evolution using a multiobjective evolutionary algorithm (MOEA). Finally, we generate the search spaces to which the random search performs better than RCGA. We expect that our method will help to analyze and understand the influence on a search of the different search algorithms (in this paper, the differences between crossover methods) on the search, and we expect to provide useful feedback for the development of EC methods.

An overview of our method is presented in the next section. In Sections III to V, we show the results of three types of experiments. Finally, in Section VI, we present our conclusions and possible future work.

II. Generation of Search Spaces to Emphasize the Performance Differences of Real-Coded Crossovers

In this paper, we evolve search spaces that emphasize the performance difference between real-coded crossovers based on references [10], [11]. Fig. 1 illustrates the outline of our method. Each individual GP program corresponds to a function (expression) that represents a search space. The

The authors are with the Graduate School of Environment and Information Sciences, Yokohama National University, 79-7, Tokiwadai, Hodogaya-ku, Yokohama, Kanagawa 240-8501, Japan (phone: +81 45 3394136; email: shinichi.shirakawa@gmail.com, yata@nlab.sogo1.ynu.ac.jp, nagao@ynu.ac.jp).

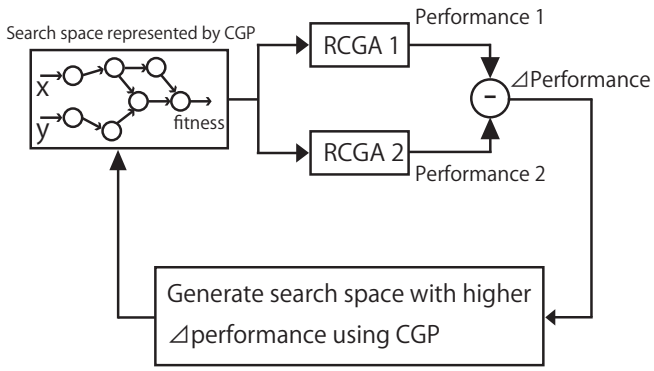


Fig. 1. Outline of our method that generates the search spaces to emphasize the performance difference of real-coded crossovers.

TABLE I
ARITHMETIC OPERATION NODES OF CGP USED IN THE EXPERIMENTS OF GENERATING THE SEARCH SPACES.

Mark	# Args	Definition
$+$	2	$x_1 + x_2$
$-$	2	$x_1 - x_2$
$\times$	2	$x_1 \times x_2$
$\div$	2	$x_1 \div x_2$ (if $x_2 = 0.0$ return x_1)
$\sqrt{\lvert\ \rvert}$	1	$\sqrt{\lvert x_1 \rvert}$
sin	1	$\sin(x_1)$
cos	1	$\cos(x_1)$
0.0	0	Constant value 0.0
1.0	0	Constant value 1.0
10.0	0	Constant value 10.0
-1.0	0	Constant value -1.0
π	0	Constant value π

performance difference of two RCGAs that differ in terms of crossover method is assumed to be a fitness function of GP in the experiment, and the search space that increases the performance difference is evolved. We expect to analyze the features that each real-coded crossover has by generating the search spaces with GP. In the experiment, two-dimensional search spaces are generated, because the grasp of the landscape of the search space is easy. Hence, each individual GP program corresponds to a function that returns a certain fitness value f from two input values (x, y).

In this paper, we use CGP [15], [16], in particular, a program that represents the network structure as a GP model. CGP constructs feed-forward network-structured programs. The nodes of CGP are categorized into three types: input nodes, arithmetic-operation nodes, and output nodes. Input nodes correspond to input values, arithmetic-operation nodes execute arithmetic operations, and the output values of the program are obtained from the output nodes. One of the benefits of this type of representation is that it allows the implicit reuse of nodes in its network. To adopt an evolutionary method, CGP uses genotype-phenotype mapping. The feed-forward network structure is encoded in the form of a linear string. Numbers are allocated to each node in advance. Increasingly large numbers are allocated to input nodes, arithmetic operation nodes, and output nodes. The nodes take their input from the output of the previous nodes in a feed-forward manner. The genotype in CGP has a fixed-length representation and consists of a string that encodes the node function ID and connections of each node in the network. However, the number of nodes in the phenotype can vary, as not all the nodes encoded in the genotype must be connected. This allows the existence of inactive nodes. Only the mutation is usually used as a genetic operator of CGP. The mutation operator affects an individual as follows:

- Randomly select several genes according to the mutation rate P_{m} for each gene.
- Randomly change the selected genes under the constraints of the structure.

The (1 + 4) evolution strategy ((1 + 4) ES) is adopted as the generation-alternation model. Because CGP has inactive nodes, a neutral effect on fitness is caused by the genetic operation (called neutrality [17]). In the (1 + 4) ES, the offspring is selected if it exhibits the same best fitness as the parent, then the searching point moves even if the fitness is not improved. Therefore, we believe that efficient search is achieved through a simple generation-alternation model. Many genetic operations in which there is no change in active nodes occur because they use a small mutation probability. In addition, the fitness calculation can be omitted when the genetic operations occur only on inactive nodes.

Table I shows the arithmetic operation nodes of CGP used in the experiments. CGP is expected to represent a complex search space using a combination of these nodes.

The fitness assignment of each individual is based on the results of the searches by two RCGAs for the search space represented by CGP. In this manner, a search space is obtained in which the performance difference of two RCGAs increases. The fitness function of each individual in CGP is defined in the following equation:

$$F_{\mathrm{cgp}} = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T}\log\left(\frac{g_1(t,n)}{g_2(t,n)}\right),$$
$$g_i(t,n) = \begin{cases} f_{\mathrm{min}} & \text{if } f_i(t,n) < f_{\mathrm{min}} \\ f_{\mathrm{max}} & \text{if } f_i(t,n) > f_{\mathrm{max}} \\ f_i(t,n) & \text{otherwize} \end{cases} \quad (1)$$

where N is the number of trials of each RCGA, T is the maximum number of generations in RCGA, and $f_i(t,n)$ is the best evaluation value on the tth generations of the nth trials. The value of $g_i(t,n)$ is limited within the range of $[f_{\mathrm{min}}, f_{\mathrm{max}}]$, and g_1 and g_2 are the values obtained from the searches of RCGA 1 and 2, which are the different settings, respectively. In the experiments, we set $N = 10$, $T = 500$, $f_{\mathrm{min}} = 10^{-8}$, $f_{\mathrm{max}} = 10.0$, respectively. The fitness function F_{cgp} is an accumulation of the differences of the log values of the best evaluation values of two RCGAs obtained in each generation. Therefore, the process of the search is also included in the performance assessment (obtaining a good evaluation value earlier in the search shows that the

TABLE II
CGP PARAMETERS USED IN THE EXPERIMENTS.

Parameter	Value
Number of generations	15000
Mutation rate P_{m}	0.02
Number of arithmetic operation nodes	200
Number of input nodes	2
Number of output nodes	1

performance is high). In the experiments for this paper, RCGAs search for the minimum value in the search spaces represented by CGP. Thus, smaller the value of $g_i(t,n)$, higher is the performance of the RCGA. Larger the value of F_{cgp}, greater is the performance difference of the RCGAs in that search space. It is shown that the performance of RCGA 2 is higher than that of RCGA 1 in the search space with a large value for F_{cgp}. A search space with a large difference between the performance of two RCGAs is evolved with CGP using such a fitness function.

Here we discuss the difference between the present study and the studies of [10], [11]. The search spaces are evolved using the performance difference between PSO, DE, and CMA-ES by tree-based GP in [10], [11]. In our study, we focus on real-coded crossovers in RCGA and evolve the search spaces using the performance difference between them using CGP. While the node functions of GP are only $\{+,-,\times,\div\}$ in [10], [11], we have detailed more node functions, as shown in Table I. Therefore, we expect to generate more complex search spaces. Our approach also differs in such a way that our fitness function considers the process of the searches. In section IV, the number of nodes in CGP is added to the objective function, and we attempt to obtain not only one search space but also a variety of search spaces simultaneously by multiobjective optimization (this is also a difference from previous studies). We can then observe which elements increase the performance difference.

III. EXPERIMENTS OF GENERATING THE SEARCH SPACES

A. Experimental Settings

The CGP parameters used in the experiments are listed in Table II. The search space is generated using the performance difference of two RCGAs with different real-coded crossovers. We use three real-coded crossovers: BLX-α [1], unimodal normal-distribution crossover (UNDX) [2], [3], and simplex crossover (SPX) [4]. The search space is generated on the basis of combination of these crossover methods in the experiments.

BLX-α uniformly and independently picks new individuals with values that lie in $[d_i - \alpha d_i, d_i + \alpha d_i]$ for each parameter, where di is the distance between the parents. The equation of BLX-α is as follows:

$$c_i = u(min(p_{1i}, p_{2i}) - \alpha I, max(p_{1i}, p_{2i}) + \alpha I) \quad (2)$$

$$I = |p_{1\mathrm{i}} - p_{2\mathrm{i}}| \quad (3)$$

where c_i is the ith parameter of offspring, p_{1i} and p_{2i} are the ith parameter of parents 1 and 2, respectively, and $u(m,n)$ is a uniformly distributed random number among $[m,n]$. We set the parameter $\alpha = 0.366$ in the experiments.

UNDX deals with variable dependence (non-separability), although the performance of BLX-α deteriorates in the search space with variable dependence. UNDX generates offspring using normally distributed random numbers defined by three parents. Offspring are generated around the line segment connecting the two parents: parent 1 and parent 2. The third parent, parent 3, is used to decide the standard deviation of the distance to the axis connecting parents 1 and 2. The generation of offspring $\vec{C}$ is as follows:

$$\vec{C} = \vec{m} + \xi\vec{d} + D\sum_{i=1}^{n-1}\eta_i\vec{e}_i \quad (4)$$

where,

$$\vec{m} = (\vec{P}_1 + \vec{P}_2)/2, \quad \vec{d} = \vec{P}_2 - \vec{P}_1, \quad (5)$$

$$D = \sqrt{|\vec{P}_3 - \vec{P}_1|^2 - \frac{[\vec{d}\cdot(\vec{P}_3 - \vec{P}_1)]^2}{|\vec{d}|^2}}, \quad (6)$$

$$\xi \sim N(0, \sigma_\xi^2), \quad \eta_i \sim N(0, \sigma_\eta^2), \quad (7)$$

$\vec{P}_1$, $\vec{P}_2$ and $\vec{P}_3$ are parent vectors of parents 1, 2, and 3, respectively, $\vec{e}_i$ is the orthogonal basis vector spanning the subspace perpendicular to the line connecting parents 1 and 2, D is the distance of the third parent from the line connecting parents 1 and 2, n is the number of dimensions in the search space, and $N(0,\sigma^2)$ is a normally distributed random number. We set the parameters $\sigma_\xi = 0.5$, $\sigma_\eta = 0.35/\sqrt{n}$, which are the recommended values [2], [3].

SPX is a multiparent recombination operator. It generates offspring vector values by uniformly sampling values from the simplex formed by $n+1$ (n is the number of dimensions in the search space) parent vectors. SPX is independent from coordinate systems in generating offspring. The SPX procedure is as follows:

1) Calculate the center of mass $\vec{G}$ from the parents $\vec{P}_1, ..., \vec{P}_{n+1}$.

$$\vec{G} = \frac{1}{n+1}\sum_{i=1}^{n+1}\vec{P}_i \quad (8)$$

2) Calculate $\vec{x}_k$ and $\vec{C}_k$ for $k = 1, ..., n+1$.

$$\vec{x}_k = \vec{G} + \varepsilon(\vec{P}_k - \vec{G}) \ \ (k = 1, ..., n+1) \quad (9)$$

$$\vec{C}_k = \begin{cases} \vec{0} \ \ (k=1) \\ r_k(\vec{x}_{k-1} - \vec{x}_k + \vec{C}_{k-1}) \\ (k = 2, ..., n+1) \end{cases} \quad (10)$$

where ε is the positive-valued parameter called expansion rate. r_k is the random number calculated by the following equation:

$$r_k = (u(0,1))^{\frac{1}{k}} \ \ (k = 2, ..., n+1) \quad (11)$$

where $u(0,1)$ is a uniformly distributed random number among $[0,1]$.

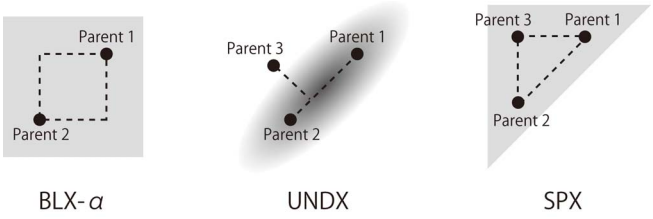


Fig. 2. Conceptual illustration of the distribution of the offspring generated for each real-coded crossover (two dimensions). BLX-α and SPX generate the offspring using uniformly distributed random numbers, while UNDX generates the offspring using normally distributed random numbers.

3) Generate the offspring $\vec{C}$.

$$\vec{C} = \vec{x}_{n+1} + \vec{C}_{n+1} \tag{12}$$

We set the parameter $\varepsilon = \sqrt{n+2}(= 2.0)$ that is the recomended value [4].

Fig. 2 illustrates the distribution of the offspring generated for each real-coded crossover in the case of a two-dimensional search space.

We use the minimal generation gap (MGG) model [18] as a generation-alternation model. It is a steady-state model proposed by Satoh et al., which exhibits the desirable convergence property of maintaining the diversity of the population, and is widely used in RCGAs. The MGG model is summarized as follows:

1) Set the generation counter $t = 0$. Randomly generate the individuals as the initial population $P(t)$.
2) Select a set of μ parents for the crossover by random sample from the population, where μ denotes the number of parents used for the crossover.
3) Generate a set of n_c children by applying the crossover to the parents.
4) Select two individuals from the set of two parents (the main two parents in UNDX and the randomly selected two parents in SPX) and n_c children. One is the elitist individual and the other is the individual from a rank-based roulette-wheel selection. Then replace the two parents with the new two individuals in population $P(t)$ to get population $P(t+1)$.
5) Stop if a certain specified condition is satisfied. Otherwise, set $t = t + 1$, and go to step 2.

The MGG model localizes its selection pressure, not to the whole population (as simple GA or steady state does), rather only to the family (offspring and parents).

Common RCGA parameters are used in each real-coded crossover. The population size 30, children size $n_c = 20$, and the maximum number of generations $T = 500$ are used in the experiments. We confirm from the exploratory experiment that we can perform adequate searches using these parameters, because the search space represented by CGP is two-dimensional. The range of the search spaces is assumed to be $[-10, 10]$, and initial individuals are generated with uniformly distributed random numbers within this range. When an offspring is generated outside the range by crossover, its parameters are corrected on the boundary. It breaks off the search for the RCGA when an evaluation value that is smaller than $f_{\min}$ is obtained during the search. When searching by each RCGA, the upper bound is not used in the fitness assignment in RCGA, although $f_{\max}$ is set to calculate the fitness value of CGP with Equation ((1)). In addition, when searching by each RCGA, the random numbers for each search of RCGAs are initialized using the same seeds. Therefore, the same fitness evaluation can be given each time for the same search space represented by CGP. We experiment ten different runs with the same parameter set at each combination of each real-coded crossover.

TABLE III

AVERAGE OF THE FITNESS VALUES OF THE SEARCH SPACES OBTAINED BY EACH SETTING (AVERAGE OVER 10 TRIALS WITH THE SAME PARAMETER SET). THE VALUES SHOWN IN PARENTHESES ARE CALCULATED USING STANDARD DEVIATION.

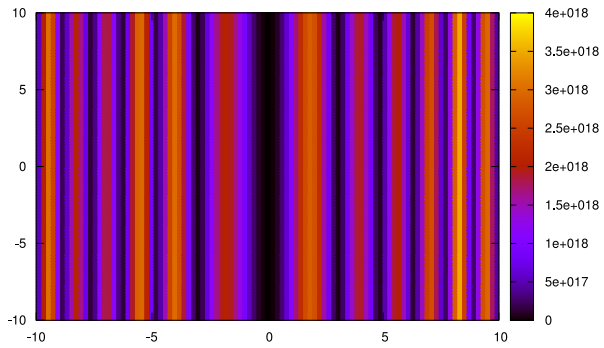
		Lose		
		BLX-α	UNDX	SPX
Win	BLX-α	–	4778	5463
		–	(1533)	(1473)
	UNDX	7521	–	5911
		(1918)	–	(1284)
	SPX	7411	2854	–
		(2024)	(1244)	–

B. Experimental Results

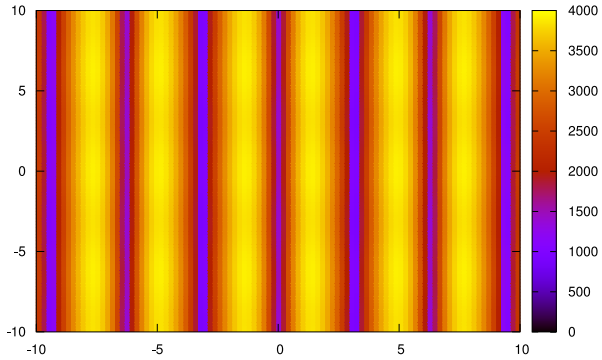
We evolve the search spaces 10 times by each setting. The average of the fitness values of the search spaces obtained by each setting is shown in Table III. The numerical values in the table show the fitness values when the search space is evolved for which the row's crossover method performs better than the column's crossover method. The values shown in parentheses are calculated using standard deviation. From this table, the search spaces that have the largest performance difference between two crossovers can be generated for all combinations.

Fig. 3 shows the landscapes with the highest fitness value among the search spaces generated in the combination of each real-coded crossover over the course of ten trials. The horizontal axis of the search space corresponds to the parameter x and the vertical axis corresponds to the parameter y. The colors used in the figures indicate the evaluation value of each coordinate. Yellow indicates high values, and blue indicates low values. We apply each real-coded crossover to each search space in Fig. 3. Table IV shows the number of successful runs for each search space over the course of 30 runs. We consider the run to be a success when an evaluation value that is smaller than $f_{\min}$ is obtained during the search.

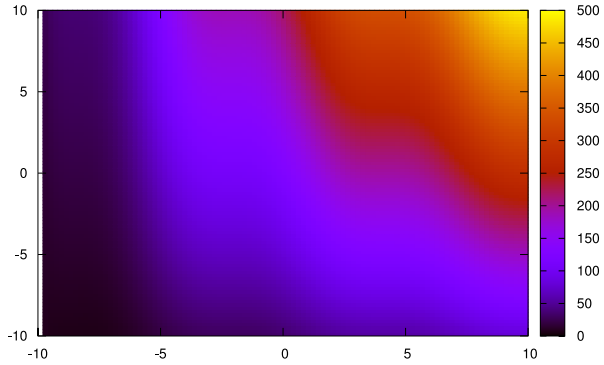
Figs. 3 (a) and (b) illustrate the search spaces to which BLX-α performs better than UNDX and SPX, respectively. Both search spaces are multimodal, and the evaluation values are decided almost exclusively on the parameter of x. In a word, there is no variable dependence. The search space in Fig. 3 (a) has a minimum value around $x = 0$, and (b) has a



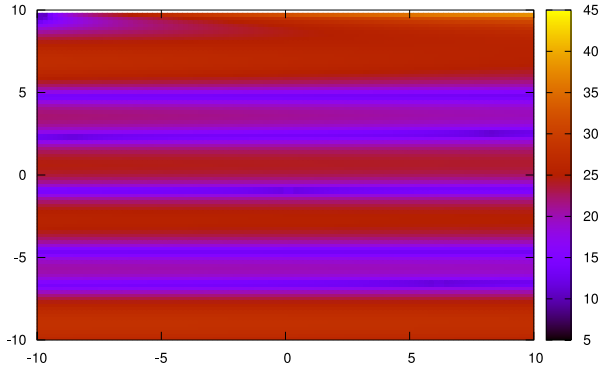
(a) BLX-α performs better than UNDX (Fitness: 7116)



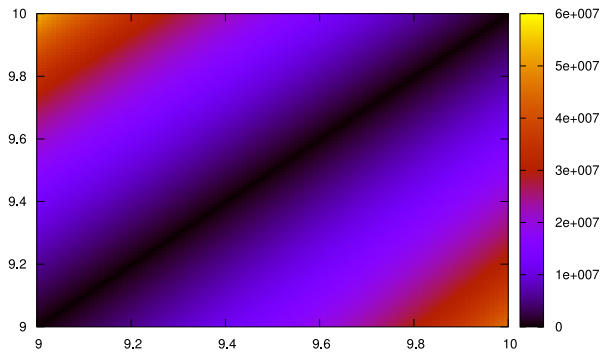
(b) BLX-α performs better than SPX (Fitness: 7669)



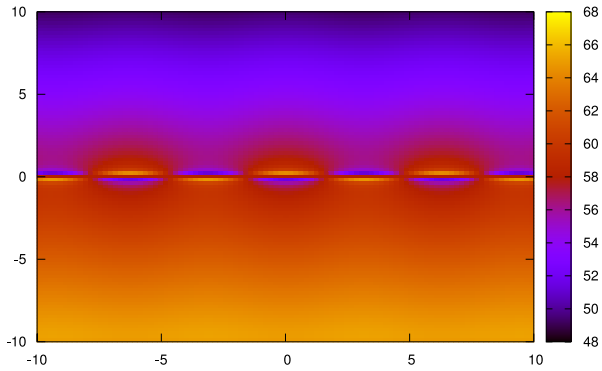
(c) UNDX performs better than BLX-α (Fitness: 9540)



(d) UNDX performs better than SPX (Fitness: 7854)



(e) SPX performs better than BLX-α (Fitness: 10132)



(f) SPX performs better than UNDX (Fitness: 5389)

Fig. 3. Examples of the generated search spaces that maximize the performance difference of real-coded crossovers. The horizontal axis of the search space corresponds to the parameter x, and the vertical axis corresponds to the parameter y. The colors in the figures indicate the evaluation value of the coordinate. Yellow indicates high values, and blue indicates low values.

minimum value near the coordinates, where x is a multiple of π. The search spaces without variable dependence (such as Figs. 3 (a) and (b)) are generated, because the performance of BLX-α deteriorates in the search space with variable dependence. From Table IV, BLX-α discovers the minimum value by all the trials, whereas the number of successful runs is a little when UNDX and SPX are applied to the search spaces.

Fig. 3 (c) illustrates the search space to which UNDX performs better than BLX-α. It is necessary to change both the x and y variables to considerably improve the evaluation value in this search space. In a word, there is variable dependence. Although this search space has minimum values near $x = -10$, it has a large value with $x = -10$. Therefore, even if the individual is generated outside the boundary and the value of $x = -10$ is obtained, a minimum value cannot be obtained. Such a situation often appears in real-world problems. It is necessary to generate the individual near $x = -10$ to obtain a minimum value in this search space. This search space is advantageous for UNDX, because UNDX generates individuals with normally distributed random numbers and can deal with variable dependence. We analyzed the process

TABLE IV
NUMBER OF SUCCESSFUL RUNS OVER THE COURSE OF 30 RUNS WHEN EACH REAL-CODED CROSSOVER IS APPLIED TO EACH SEARCH SPACE IN FIG. 3.

	BLX-α	UNDX	SPX
Fig.3 (a)	30	11	9
Fig.3 (b)	30	17	20
Fig.3 (c)	16	27	15
Fig.3 (d)	21	30	26
Fig.3 (e)	14	28	24
Fig.3 (f)	25	19	24

of the search by BLX-α, and confirmed that many individuals are generated on the boundary of $x = -10$. Thus, BLX-α fails in the search and cannot obtain a minimum value. From Table IV, the number of successful runs is the highest when UNDX are applied to this search space.

Fig. 3 (d) illustrates the search space to which UNDX performs better than SPX. Although this search space is multimodal and has a minimum value near $(x, y) = (-10, 10)$, it has a large value with $y = 10$. Therefore, it is necessary to generate the individual near $(x, y) = (-10, 10)$ to obtain a minimum value in this search space. This search space is also advantageous for UNDX as well as Fig. 3 (c), because UNDX generates individuals with normally distributed random numbers.

Fig. 3 (e) illustrates the search space to which SPX performs better than BLX-α. The range of each parameter in Fig. 3 (e) is $[9, 10]$, because the evaluation values increase considerably when parting from the straight line $x = y$. This search space has variable dependence and has the minimum values in $x = y$. From Table IV, the number of successful runs of SPX is more than that of BLX-α. However, the number of successful runs of UNDX is the highest because the performance of UNDX is not considered when the search space is evolved.

Fig. 3 (f) illustrates the search space to which SPX performs better than UNDX. This search space has minimum values in the area near $y = 0$. On the other hand, the area of large values exists in its neighborhood. The evaluation values become small increasing of the parameter of y as the entire landscape of the search space. Although the number of successful runs of SPX is more than that of UNDX in Table IV, the performance difference is not great. In addition, the number of successful runs of BLX-α is the highest because the performance of BLX-α is not considered when the search space is evolved.

Next, we discuss the functions of the generated search spaces. The function of the search space in Fig. 3 (b) is as follows:

$$\begin{aligned} f(x,y) = &\left(\sin\sqrt{|\sin(x)|} - 10\right) \\ &(9 + \sin(\sin(10)))\sin\left(\sin(\sin(10))\right)\cos(y) \\ &+10\sqrt{|\sin(x)|}\,(10 + \cos(x)) + 10 \\ &-\cos\left(10(10 + \cos(x))\right) \end{aligned} \tag{13}$$

The function of the search space in Fig. 3 (c) is as follows:

$$f(x,y) = 20\pi - 8 + \frac{\cos(x)}{y} + \sqrt{|10 + y|} - y \tag{14}$$

The functions generated by CGP include a complex one with many elements and a simple one. Because the functions that maximize Equation (1) are generated in this experiment, functions that are difficult to understand are occasionally generated. In the next section, the number of nodes in CGP is added to the objective function, and we attempt to obtain a variety of search spaces simultaneously using multiobjective optimization.

IV. GENERATION OF SEARCH SPACES USING A MULTIOBJECTIVE EVOLUTIONARY ALGORITHM

Only Equation (1) is assumed to be an objective function for generating the search spaces in the previous experiments. Therefore, complex functions (search spaces) tend to be generated. Understanding and analyzing the features of the search spaces becomes difficult, because the search spaces to which the performance difference increases tend to be complex. In this section, we extend the problem to a binary objective problem by adding the number of active nodes to the fitness function of CGP. We also attempt to obtain not only one search space but also a variety of search spaces simultaneously by multiobjective optimization. We can then observe which types of elements expand the performance difference. Equation (1) is used for the first objective function, and the number of active nodes in CGP is used for the second objective function. Equation (1) is expected to be maximized, and the number of active nodes are expected to be minimized. We use strength Pareto evolutionary algorithm 2 (SPEA2) [19] as the MOEA technique. SPEA2 assigns fitness values to individuals according to dominance and density criteria. The selection of individuals is also based on these criteria. It maintains two populations: a current population and an archive population. In each generation, the nondominated solutions in the current population and the archive are copied into the archive of the next generation. Selection occurs from both the population and the archive. The parameters of CGP are shown in Table II and the parameters of RCGAs that are explained in section III-A are used. We use population size 5 and archive size 20 as the parameters of SPEA2. We use BLX-α and UNDX as the real-coded crossovers in this experiment, and the search spaces to which UNDX performs better than BLX-α, and to which BLX-α performs better than UNDX, are evolved.

The functions of the nondominated solutions obtained by the MOEA are shown in Tables V and VI. We confirm that the performance difference increases as the number of active nodes increases, and there are common components in each function. Multimodality increases the performance difference of BLX-α and UNDX in Table V. From VI, the variable dependence and the sharp landscape (square root) increase the performance difference of UNDX and BLX-α. Fig. 4 shows the evolved landscapes to which BLX-α performs better than UNDX when the number of active nodes is 7, 9,

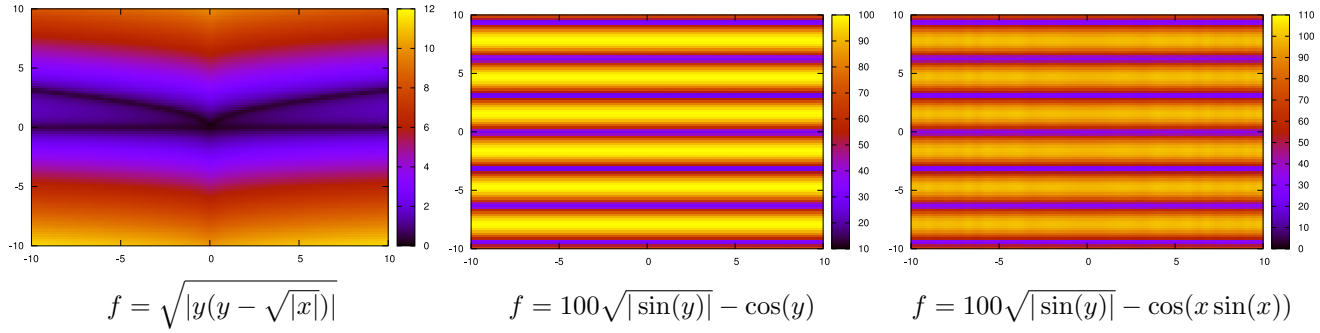


Fig. 4. Examples of the generated search spaces by the MOEA. The number of active nodes is 7, 9, and 12, respectively. BLX-α performs better than UNDX in these search spaces.

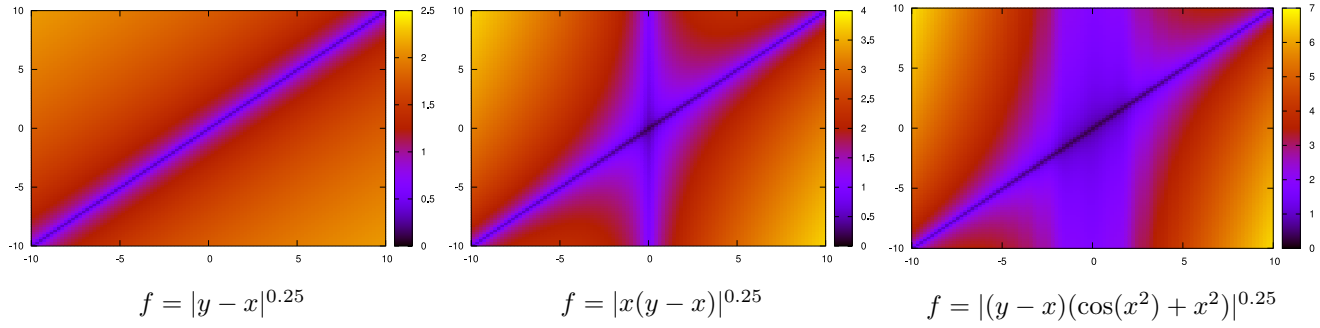


Fig. 5. Examples of the generated search spaces by the MOEA. The number of active nodes is 6, 7, and 10, respectively. UNDX performs better than BLX-α in these search spaces.

TABLE V

FUNCTIONS OF THE NONDOMINATED SOLUTIONS OBTAINED BY THE MOEA. BLX-α PERFORMS BETTER THAN UNDX IN THESE SEARCH SPACES.

Function	# nodes	Fitness
$\sqrt{\lvert y\rvert}$	3	1570
$\sqrt{\lvert 2y\rvert}$	4	1593
$\sqrt{\lvert xy\rvert}$	5	1704
$\sqrt{\lvert y(y + \cos(y))\rvert}$	6	1775
$\sqrt{\lvert y(y - \sqrt{\lvert x\rvert})\rvert}$	7	2122
$100\sqrt{\lvert \sin(y)\rvert} - \cos(y)$	9	4272
$100\sqrt{\lvert \cos(y)\rvert} - \cos(y) - \cos(\sqrt{\lvert \cos(y)\rvert})$	10	4750
$100\sqrt{\lvert \cos(y)\rvert} - \cos(y) - \cos(x)$	11	4940
$100\sqrt{\lvert \sin(y)\rvert} - \cos(x\sin(x))$	12	5782

TABLE VI

FUNCTIONS OF THE NONDOMINATED SOLUTIONS OBTAINED BY THE MOEA. UNDX PERFORMS BETTER THAN BLX-α IN THESE SEARCH SPACES.

Function	# nodes	Fitness
$f = 10 - y$	4	37
$f = \sqrt{\lvert y - x\rvert}$	5	2286
$f = \lvert y - x\rvert^{0.25}$	6	2640
$f = \lvert x(y - x)\rvert^{0.25}$	7	4268
$f = \sqrt{\lvert (y - x)(\cos(x^2) + x^2)\rvert}$	9	6092
$f = \lvert (y - x)(\cos(x^2) + x^2)\rvert^{0.25}$	10	7144

and 12, respectively. These search spaces have little variable dependence and are the multimodal landscapes. Fig. 5 shows the evolved landscapes to which UNDX performs better than BLX-α when the number of active nodes is 6, 7, and 10, respectively. These search spaces have the minimum values in $x = y$, and they have variable dependence. The area where the evaluation values are low is formed around $x = 0$ as the node increases.

V. GENERATION OF SEARCH SPACES TO EMPHASIZE PERFORMANCE DIFFERENCE WITH RANDOM SEARCH

In this section, we attempt the evolution of search spaces to which the random generation of individuals (random search) performs better than real-coded crossovers. The setting of the experiment is the same as that in section III-A.

As a result, the search spaces to which the random search performs better than each real-coded crossover could be generated. Fig. 6 shows the example of the search space to which the random search performs better than BLX-α. The fitness value of this search space calculated by Equation (1) is 9901. The favorable evaluation values are macroscopically obtained when the value of x is small in this search space. However, the areas that have a minimum value actually exist around the value of x from 9 to 10. This search space belongs to a so-called deceptive problem. BLX-α fails to find the global optimum for such a deceptive problem, because the population converges to coordinates with small values of x. In this situation, the probability that the random search

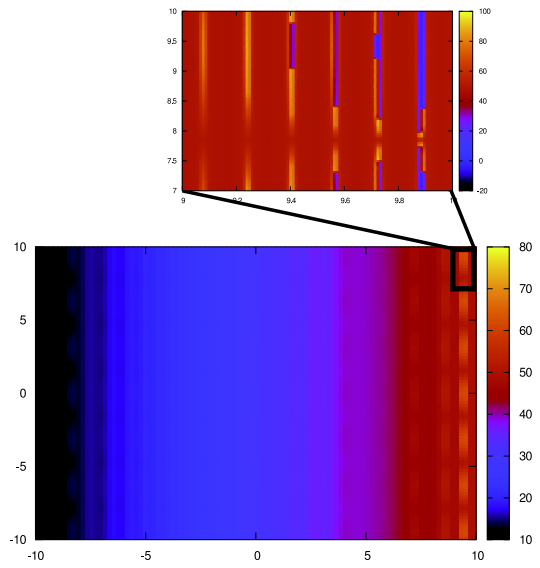


Fig. 6. Example of the search space to which the random search performs better than BLX-α.

discovers the optimum solution compared to RCGAs is high. The search spaces to which the random search performs better than UNDX and SPX have the tendency to a deceptive structure similar to Fig. 6. Although the search space shown in Fig. 6 is a typical problem for which the EC techniques fail in the search, we think that being able to generate such a search space by our method is one of the valuable results.

VI. Conclusions and Future Work

In this paper, the search spaces that emphasize the performance difference of real-coded crossovers were evolved. We used the combination of three types of real-coded crossovers in the experiments: BLX-α, UNDX, and SPX. In the experimental results, the search spaces that have the largest performance difference between two crossovers could be generated for all the combinations. After that, the number of nodes in CGP was added to the objective function, and we evolved a variety of search spaces simultaneously using a multiobjective evolutionary algorithm. Moreover, we confirmed that our method generates the search spaces to which the random search performs better than the search of the real-coded crossovers.

In future, we plan to evolve the search spaces not only to use the difference of the real-coded crossovers, but also to use the difference of the generation alternation models and each parameter. Search spaces can be generated using the performance difference between other evolutionary computation methods and the multiobjective evolutionary algorithms. Furthermore, higher-dimensional search spaces will be evolved through the search spaces of the two dimensions that were evolved in this paper. Finally, we expect to better understand the features and the behavior of evolutionary computation techniques from the analysis of the search spaces generated by our method. We also expect that the results of our method will contribute to the development of a new evolutionary algorithm and the discovery of a new problem domain.

References

[1] L. J. Eshelman and J. D. Schaffer, "Real coded genetic algorithms and interval-schemata," in *Foundations of Genetic Algorithms 2*. Morgan Kaufmann Publishers, 1993, pp. 187–202.

[2] I. Ono and S. Kobayashi, "A real-coded genetic algorithm for function optimization using unimodal normal distribution crossover," in *Proceedings of the 7th International Conference on Genetic Algorithms (ICGA '97)*. Morgan Kaufmann Publishers, 1997, pp. 246–253.

[3] H. Kita, I. Ono, and S. Kobayashi, "Theoretical analysis of the unimodal normal distribution crossover for real-coded genetic algorithms," in *Proceedings of the 1998 IEEE Congress on Evolutionary Computation (CEC '98)*, 1998, pp. 529–534.

[4] S. Tsutsui, M. Yamamura, and T. Higuchi, "Multi-parent recombination with simplex crossover in real coded genetic algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference 1999 (GECCO '99)*. Morgan Kaufmann Publishers, 1999, pp. 657–664.

[5] K. Deb, A. Anand, and D. Joshi, "A computationally efficient evolutionary algorithm for real-parameter optimization," *Evolutionary Computation*, vol. 10, no. 4, pp. 371–395, 2002.

[6] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[7] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the 1995 IEEE International Conference on Neural Networks*, 1995, pp. 1942–1948.

[8] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[9] M. Oltean, "Searching for a practical evidence of the no free lunch theorems," in *Biologically Inspired Approaches to Advanced Information Technology: First International Workshop, BioADIT 2004*, ser. LNCS, vol. 3141. Springer-Verlag, 2004, pp. 472–483.

[10] W. B. Langdon and R. Poli, "Evolving problems to learn about particle swarm and other optimisers," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC '05)*, vol. 1, 2005, pp. 81–88.

[11] W. B. Langdon and R. Poli, "Evolving problems to learn about particle swarm optimizers and other search algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 5, pp. 561–578, 2007.

[12] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[13] R. Poli, W. B. Langdon, and N. F. McPhee, *A Field Guide to Genetic Programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`, 2008, (With contributions by J. R. Koza).

[14] J. I. van Hemert, "Evolving combinatorial problem instances that are difficult to solve," *Evolutionary Computation*, vol. 14, no. 4, pp. 433–462, 2006.

[15] J. F. Miller and P. Thomson, "Cartesian genetic programming," in *Genetic Programming: 3rd European Conference, EuroGP 2000*, ser. LNCS, vol. 1802. Springer-Verlag, 2000, pp. 121–132.

[16] J. F. Miller and S. L. Smith, "Redundancy and computational efficiency in cartesian genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 2, pp. 167–174, 2006.

[17] T. Yu and J. F. Miller, "Neutrality and the evolvability of boolean function landscape," in *Genetic Programming: 4th European Conference, EuroGP 2001*, ser. LNCS, vol. 2038. Springer-Verlag, 2001, pp. 204–217.

[18] H. Satoh, M. Yamamura, and S. Kobayashi, "Minimal generation gap model for considering both exploration and exploitations," in *Proceedings of the IIZUKA'96*, 1996, pp. 494–497.

[19] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm," Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Zurich, Switzerland, Tech. Rep. 103, 2001.