An New Crossover Operator for Real-coded Genetic Algorithm with Selective Breeding Based on Difference Between Individuals

Zhi-Qiang Chen
School of Computer Science and Information Engineering, Chongqing Technology and Business University
Chongqing, China

Yuan-Fu Yin
School of Economic and Trade,
Chongqing Technology and Business University
Chongqing, China

Abstract—**In this paper, we present an efficient crossover operator for real-coded genetic algorithm that breeds offspring based on difference between individuals. In the proposed crossover operator, offspring are generated following a promising direction with Laplace distribution based on the center of mass of parents. A set of 15 test problems available in the global optimization literature is used to evaluate the performance of proposed genetic algorithm. The comparative study shows that the proposed genetic algorithm performs quite well and outperforms other algorithms. (Abstract)**

Keywords- Genetic Algorithm; Function Optimization; Real-Coded

I. INTRODUCTION

A variety of application problems in engineering, science and technology can be formulated as nonlinear global optimization problems having local as well as global optima. For finding near global minima, many stochastic optimization techniques like evolutionary algorithms, simulated annealing etc. have been developed which rely heavily on computational power. Among these, Genetic algorithm (GA) method, a robust and efficient search technique, has been used to many engineering applications since it was introduced. The popularity of this method is based on simply solving multidimensional and multimodal optimization problems without requiring any additional information such as the gradient of an objective function. Although the origin of this method proposed binary number for encoding, over the past ten years, there have been a surge of studies related to real-coded genetic algorithms (RCGA) for continue space problem [1~15]. In RCGAs, crossover has always been considered to be the fundamental search operator. A lot of efforts have been put into the development of sophisticated real-coded crossover operators to improve the performances of RCGAs for real-parameter optimization. A serial of crossover operators have been presented such as Heuristic crossover [1], Flat crossover [2], Arithmetical crossover [3], Blend crossover (BLX-α) [4], Simulated binary crossover (SBX) [7], Unimodal normal distribution crossover (UNDX) [8], Simplex crossover (SPX) [9], Parent centric crossover (PCX) [10], Laplace Crossover [11] and Real-coded ensemble crossover (REX^{star}) [12].

We also developed several real-coded genetic algorithms that breeds offspring based on difference between individuals [13~15]. In [13], the real-coded genetic algorithm with selective breeding based on difference between individuals was firstly proposed and called *rc*-CGA. BLX-α and UNDX crossover operators were applied to the *rc*-CGA in [14]. In [15], we proposed new crossover operators called FPDD-LX for *rc*-CGA. The simulation results in [13~15] showed the *rc*-CGA have excellent performance for continuous function optimization. In this paper, we add a neighbor search mechanism to the *rc*-CGA and rename the real-coded genetic algorithm as real-coded genetic algorithm with selective breeding based on difference between individuals (*rc*-SBGA). In additional, we extend the crossover operator FPDD-LX that generates offspring following a promising direction with Laplace distribution based on the center of mass of parents (*EX*-FPDD-LX). By combining *EX*-FPDD-LX with *rc*-SBGA we define a new real-coded genetic algorithm called *rc*-SBGA+*EX*-FPDD-LX. A set of 15 test problems available in the global optimization literature [11~12] are used to evaluate the performance of the proposed genetic algorithm.

II. PROPOSED REAL–CODED GENETIC ALGORITHM

A. Real-coded Genetic Algorithm with Selective Breeding based on Difference Between Individuals

The original genetic algorithm with selective breeding based on difference between individuals was proposed as a binary-coded GA called CGA, and thus it was applied to some combinatorial optimization problems [16~18]. In [13], we modify and extend the CGA to real-coded genetic algorithm for parameter optimization and called the real-coded CGA as *rc*-CGA. In the CGA, only these parents whose *difference-degree* is larger than a given *threshold* produce offspring; other parents are performed mutation to keep the diversity. The *difference-degree* is an important parameter denoting the similarity between two chromosomes.

978-1-4577-2133-5/10/$26.00 ©2012 IEEE

In this section, we add a new neighbor search mechanism to the *rc*-CGA model: If the *difference-degree* of parents is smaller than a given *threshold*, one neighbor search operator is applied to one of parents to produce a child. We rename the *rc*-CGA with neighbor search mechanism as real-coded genetic algorithm with selective breeding based on difference between individuals (*rc*-SBGA). In the *rc*-SBGA, The difference of two parents is designed as following: Given two chromosomes $\vec{x}_1$ and $\vec{x}_2$, the *difference-degree* between $\vec{x}_1$ and $\vec{x}_2$ is defined as:

$$d_i = \frac{\left\|\vec{e}_1 - \vec{e}_2\right\|}{2} \tag{1}$$

where $\left\|\vec{e}_1 - \vec{e}_2\right\|$ is the distance between the vectors $\vec{e}_1$ and $\vec{e}_2$, $\vec{e}_1$ and $\vec{e}_2$ are computed as follows:

$$\begin{aligned} \vec{e}_1 &= \vec{x}_1 / \left\|\vec{x}_1\right\| \\ \vec{e}_2 &= \vec{x}_2 / \left\|\vec{x}_2\right\| \end{aligned} \tag{2}$$

Another important parameter in the *rc*-SBGA is the given threshold D_s. D_s is decreased in every generation and defined as:

$$D_S(t+1) = \mu D_S(t) \tag{3}$$

where t expresses t^{th} generation, $\mu \in (0,1)$ is a constant called cooling ratio.

Let N_P be the population size, the *rc*-SBGA is intertwined in the following manner:

Step1. Set the cooling ratio μ, and the initial value of the *threshold* D_s; randomly generate N_P initial individuals as population seeding.

Step2. Create mating pool P_M with N_C individuals using tournament selection to generate offspring.

Step3. Produce N_C offspring according to the following sub-procedure:

①Set *Counter* to 0 as a counter of offspring.

②Select uniformly at random (N_C - *Counter*)/2 pairs of individual as parents from the mating pool P_M.

③ Calculate *difference-degree* d_i of each pair of parents using (1).

④ If the *difference-degree* d_i of two parents is larger than *threshold* D_s, one crossover operator is applied on the parents to generate offspring.

⑤ Else if the *difference-degree* d_i of the parents is smaller than *threshold* D_s, a neighbor search operator is performed to one of two parents and generate an offspring; a mutation operator is applied to another parent.

⑥ Evaluate offspring, then update the current best solution and the *Counter*. If N_C = *Counter*, all offspring replace last generation as a new generation and terminate this sub-procedure.

⑦If $N_C < N_P$, Go to sub step ②.

Step4. Decrease *threshold* D_s by using (3).

Step5.Terminate this procedure if termination criterion is reached.

Step6. Go to step2.

In terms of the neighbor search operator above mentioned, we define as follow:

$$\vec{y} = \vec{x} + diag(\omega_1^t, \omega_2^t, ..., \omega_D^t)(\vec{x}_B - \vec{x}) \tag{4}$$

where $\vec{y}$ is offspring of $\vec{x}$, $\omega_i^t \in [0,t]$ is a random number uniformly distributed; t is a parameter defined by user which decides the movement towards the promising direction $\vec{x}_B - \vec{x}$; D is the dimension of $\vec{x}$. Michalewicz's Non-Uniform Mutation (NUM) [3] is uses as mutation operators above mentioned.

B. Extended FPDD-LX Crossover Operators(EX-FPDD-LX)

A proper trade-off between exploration and exploitation is necessary for the efficient and effective operation of a population-based stochastic search technique like GA. An efficient search technique always hopes that it is enabled to be guided quickly to search the attracted space and then exploit the space. In [15], we proposed a crossover operator called FPDD-LX based on the idea which to create offspring following some promising descent directions. FPDD-LX is a parent-centric recombination operator. Here we use mean-centric concept and modify the FPDD-LX as follows. Given two parents $\vec{x}_1$ and $\vec{x}_2$, the offspring $\vec{x}_o$ is created following a promising direction with Laplace distribution based on the center of mass of parents as follows:

$$\vec{x}_m = \frac{\vec{x}_1 + \vec{x}_2}{2} \tag{5}$$

$$\begin{aligned} \vec{x}_o = \vec{x}_m &+ diag(\omega_1^t, \omega_2^t, ..., \omega_D^t)(\vec{x}_P - \vec{x}_m) \\ &+ \xi(\vec{x}_1 - \vec{x}_2) \end{aligned} \tag{6}$$

$$\xi = \begin{cases} -b \log_e r & if\ r \le 0.5 \\ b \log_e r & if\ r > 0.5 \end{cases} \tag{7}$$

$$f(\xi) = \frac{1}{2b} \exp(-\frac{|\xi|}{b}) \tag{8}$$

where $\vec{x}_P$ represents an attracted point. We use the champion of the mating pool P_M as the attracted point in this paper. $\omega_i^t \in [0,t]$ and t are the same as that in (4), which decide the movement towards the promising direction $\vec{x}_P - \vec{x}_m$; ξ is a random generated number using Laplace(0, b) distribution with probability density function (8).

III. SIMULATIONS

By combining *rc*-SBGA with EX-FPDD-LX, we define a real-coded genetic algorithm called *rc*-SBGA+EX-FPDD-LX, and evaluate its performance by simulating a larger number of

benchmark functions.

A. Test Bed

We used a test-bed of 15 traditional benchmark functions (f_1~f_{15}) available in the global optimization literature [11~12] to evaluate the performance of the proposed scheme. They are reported as following:

1. Sphere function (f_1)

$$\min_x f(x) = \sum_{i=1}^{n} x_i^2,$$

$$-100 \le x_i \le 100,\ x^* = (0,0,...,0),\ f(x^*) = 0.$$

2. Ellipsoid function (f_2)

$$\min_x f(x) = \sum_{i=1}^{n} (1000^{i-1/n-1} x_i)^2,$$

$$-100 \le x_i \le 100,\ x^* = (0,0,...,0),\ f(x^*) = 0.$$

3. k-tablet function (f_3)

$$\min_x f(x) = \sum_{i=1}^{k} x_i^2 + \sum_{i=k+1}^{n} (100x_i)^2,$$

$$-5.12 \le x_i \le 5.12,\ x^* = (0,0,...,0),\ f(x^*) = 0.$$

4. Schewefel problem 3 (f_4)

$$\min_x f(x) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i|,$$

$$-10 \le x_i \le 10, x^* = (0,0,...,0)\ \textit{and}\ f(x^*) = 0.$$

5. Schewefel problem 4 (f_5)

$$\min_x f(x) = \max_x \{|x_i|, 1 \le i \le n\},$$

$$-100 \le x_i \le 100, x^* = (0,0,...,0)\ \textit{and}\ f(x^*) = 0.$$

6. Axis parallel hyper ellipsoid (f_6)

$$\min_x f(x) = \sum_{i=1}^{n} i x_i^2,$$

$$-100 \le x_i \le 100, x^* = (0,0,...,0)\ \textit{and}\ f(x^*) = 0.$$

7. Zakharov's function (f_7)

$$\min_x f(x) = \sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} \frac{i}{2} x_i)^2 + (\sum_{i=1}^{n} \frac{i}{2} x_i)^4,$$

$$-100 \le x_i \le 100,\ x^* = (0,0,...,0)\ \textit{and}\ f(x^*) = 0.$$

8. Exponential problem (f_8)

$$\min_x f(x) = -\exp(0.5 \sum_{i=1}^{n} x_i^2),$$

$$-1 \le x_i \le 1,\ x^* = (0,0,...,0)\ \textit{and}\ f(x^*) = -1.$$

9. Ellipsoidal function (f_9)

$$\min_x f(x) = \sum_{i=1}^{n} (x_i - i)^2,$$

$$-n \le x_i \le n,\ x^* = (1,2,...,n)\ \textit{and}\ f(x^*) = 0.$$

10. Ackley's problem (f_{10})

$$\min_x f(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2})$$

$$-\exp(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)) + 20 + e,$$

$$-30 \le x_i \le 30,\ x^* = (0,0,...,0)\ \textit{and}\ f(x^*) = 0.$$

11. Cosine mixture problem (f_{11})

$$\min_x f(x) = \sum_{i=1}^{n} x_i^2 - 0.1\sum_{i=1}^{n} \cos(5\pi x_i),$$

$$-1 \le x_i \le 1,\ x^* = (0,0,...,0)\ \textit{and}\ f(x^*) = -0.1n.$$

12. Levy and Montalvo problem 2 (f_{12})

$$\min_x f(x) = 0.1(\sin^2(3\pi x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]),$$

$$-5 \le x_i \le 5,\ x^* = (0,0,...,0)\ \textit{and}\ f(x^*) = 0.$$

13. Generalized penalized function 1 (f_{13})

$$\min_x f(x) = \frac{\pi}{n}(10\sin^2(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2) + \sum_{i=1}^{n} u(x_i, 10, 100, 4),$$

$$\textit{where}\ y_i = 1 + \tfrac{1}{4}(x_i + 1),$$

$$-50 \le x_i \le 50,\ x^* = (0,0,...,0)\ \textit{and}\ f(x^*) = 0.$$

14. Generalized penalized function 2 (f_{14})

$$\min_x f(x) = 0.1(\sin^2(3\pi x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2)[1 + \sin^2(2\pi x_n)] + \sum_{i=1}^{n} u(x_i, 10, 100, 4),$$

$$-50 \le x_i \le 50,\ x^* = (0,0,...,0)\ \textit{and}\ f(x^*) = 0.$$

In the problem 15 and 16, the penalty function u is given by the following expression:

$$u(x,a,k,m) = \begin{cases} k * pow((x-a),m) & if\ x > a, \\ -k * pow((x-a),m) & if\ x < -a, \\ 0 & otherwise. \end{cases}$$

15. Bohachevsky function (f_{15})

$$\min_x f(x) = \sum_{i=1}^{n-1} (x_i^2 + 2x_{i+1}^2 - 0.3\cos(3\pi x_i) - 0.4\cos(4\pi x_{i+1}) + 0.7),$$

$$-5.12 \le x_i \le 5.12,\ x^* = (0,0,...,0),\ f(x^*) = 0.$$

TABLE I NUMBER OF FES TO ACHIEVE THE FIXED ACCURACY LEVEL 10^{-7} USING RC-SBGA+EX-FPDD-LX, D=30 TO 100

Function	D	*Max*FEs	$N_P(N_C)$	*Least*FEs	*Mean*FEs	*Most*FEs	Standard deviation	No. of successful runs
f_1	30	3×10^5	D(D)	1.43E+4	1.62E+4	1.78E+4	904.10	25
	50	5×10^5	D(D)	3.40E+4	3.67E+4	3.91E+4	1536.11	25
	100	1×10^6	D(D)	1.25E+5	1.32E+5	1.41E+5	3971.36	25
f_2	30	3×10^5	D(D)	2.88E+4	3.80.E+4	4.33E+5	3715.70	25
	50	5×10^5	D(D)	5.13E+4	5.93E+4	7.06E+4	5238.11	25
	100	1×10^6	D(D)	2.01E+5	2.46E+5	2.99E+5	20642.88	25
f_3	30	3×10^5	D(D)	2.80E+4	3.91E+4	6.23E+4	8625.18	25
	50	5×10^5	D(D)	4.32E+4	5.29E+4	6.27E+5	5100.22	25
	100	1×10^6	D(D)	1.88E+5	2.14E+5	2.59E+5	18616.53	25
f_4	30	3×10^5	D(D)	2.88E+4	3.23E+4	3.79E+4	2118.34	25
	50	5×10^5	D(D)	5.51E+4	6.38E+4	7.42E+4	4840.24	25
	100	1×10^6	D(D)	1.82E+5	2.06E+5	2.36E+5	15098.43	25
f_5	30	1×10^6	10D(10D)	4.41E+5	5.43E+5	6.29E+5	49151.90	25
	50	2×10^6	12D(D)	7.59E+5	8.62E+5	9.71E+5	48709.03	25
f_6	30	3×10^5	D(D)	5.75E+4	7.35E+4	8.96E+4	8803.71	25
	50	5×10^5	D(D)	8.55E+4	1.36E+5	2.56E+5	43852.05	25
	100	1×10^6	D(D)	2.13E+5	3.26E+5	6.11E+5	122493.80	25
f_7	30	3×10^5	D(D)	9.18E+4	1.05E+5	1.26E+5	8583.41	25
	50	1×10^6	D(D)	4.50E+5	5.16E+5	5.58E+5	27959.96	25
	100	2×10^6	D(D)	8.69E+5	9.34E+5	1.04E+6	33572.91	25
f_8	30	3×10^5	D(D)	7.95E+3	9.12E+3	1.06E+4	676.32	25
	50	5×10^5	D(D)	1.96E+4	2.17E+4	2.36E+4	1011.73	25
	100	1×10^6	D(D)	7.41E+4	7.88E+4	8.43E+5	2659.76	25
f_9	30	3×10^5	4D(4D)	2.98E+4	4.34E+4	1.17E+5	15954.46	25
	50	5×10^5	6D(6D)	1.24E+5	1.54E+5	3.23E+5	37007.72	25
	100	1×10^6	D(D)	7.41E+4	7.88E+4	8.43E+5	2659.76	25
f_{10}	30	3×10^5	D(D)	4.00E+4	5.51E+4	6.93E+4	8776.90	25
	50	5×10^5	D(D)	9.09E+4	1.49E+5	2.63E+5	37456.73	25
	100	1×10^6	6D(D)	3.37E+5	3.67E+5	3.99E+5	15964.11	25
f_{11}	30	3×10^5	D(D)	3.49E+4	7.34E+4	1.39E+5	28468.83	25
	50	1×10^6	D(D)	1.64E+5	3.29E+5	7.01E+5	120604.73	25
	100	2×10^6	D(D)	6.45E+5	1.25E+6	1.63E+6	255529.41	25
f_{12}	30	3×10^5	D(D)	2.35E+4	3.98E+4	6.44E+4	10405.40	25
	50	3×10^5	D(D)	4.54e+4	9.41e+4	1.83E+5	40087.13	25
	100	1×10^6	D(D)	1.45E+5	2.52E+5	5.60E+5	104341.31	25
f_{13}	30	3×10^5	D(D)	2.21E+4	4.03E+4	8.83E+4	15209.10	25
	50	5×10^5	D(D)	5.53E+4	7.83E+4	1.30E+5	18491.29	25
	100	1×10^6	D(D)	1.75E+5	2.40E+5	4.33E+5	57150.23	25
f_{14}	30	3×10^5	D(D)	3.75E+4	4.73E+4	6.38E+4	6354.69	25
	50	5×10^5	2D(2D)	6.94E+4	9.27E+4	2.30E+5	32643.92	25
	100	1×10^6	D(D)	1.79E+5	2.51E+5	4.32E+5	66881.56	25
f_{15}	30	3×10^5	D(D)	1.75E+4	6.44E+4	1.41E+5	33309.79	25
	50	5×10^5	D(D)	7.26E+4	1.72E+5	4.19E+5	93781.10	25
	100	2×10^6	D(D)	3.96E+5	6.84E+5	1.26E+6	232065.75	25

B. Experiment

We test D = 30 to 100 for each problem using 25 independent runs, where D represents the number of dimension of the variable in the entire problems. Initial individual is initialized uniformly at random within the search space. All experiments are performed to evaluate the performance of the proposed genetic algorithm at a PC station. The evaluation criterion is the number of the function evaluations (*FEs*) to achieve the fixed accuracy (within the error value 10^{-7}). A run is terminated before reaching the max number of function evaluations if the error value $f(\boldsymbol{x}) - f(\boldsymbol{x}^*)$ is 10^{-7} or less, where $\boldsymbol{x}^*$ is the global optimal solution.

Finding the most appropriate combination of parameters occurring in a GA is termed as parameter tuning and is considered to be the most important and perhaps most difficult task. In case of real-coded GA, parameter tuning is generally more difficult as compared to binary coded GA due to the

Table II COMPARISONS WITH OTHER ALGORITHMS

Func.	*rc*-SBGA+ EX-FPDD-LX	MMG+ UNDX	MMG+ SPX	SGA+ LX+NUM
f_1	**1.62E+4**	8.03e+4	6.81e+5	2.22e+5
f_2	**3.80.E+4**	-- --	1.94e+6	3.39e+5
f_3	**3.91E+4**	-- --	1.95e+6	3.39e+5
f_4	**3.23E+4**	4.07e+6	1.65e+6	3.41e+5
f_5	**5.43E+5**	1.32e+6	1.79e+6	-- --
f_6	**7.35E+4**	3.09e+5	9.08e+5	2.53e+5
f_7	**1.05E+5**	2.56e+5	9.11e+5	-- --
f_8	**9.12E+3**	6.11e+4	6.32e+5	1.70e+5
f_9	**4.34E+4**	1.12e+5	-- --	2.71e+5
f_{10}	**5.51E+4**	3.63e+5	1.54e+6	8.14e+5
f_{11}	**7.34E+4**	1.58e+5	7.83e+5	2.16e+5
f_{12}	**3.98E+4**	3.58e+5	7.42e+5	1.80e+5
f_{13}	**4.03E+4**	9.84e+4	6.89e+5	1.75e+5
f_{14}	**4.73E+4**	1.95e+5	7.41e+5	1.93e+5
f_{15}	**6.44E+4**	2.41e+6	9.74e+5	2.73e+5

simple reason that the numbers of tunable parameters occurring in a real-coded GA are usually more than that occurring in binary GA. To achieve this goal, we have carried out extensive experiments for the proposed *rc*-SBGA+EX-FPDD. There exist seven important parameters in the proposed approach: (1) The size of population size N_P; 2)the size of mating pool N_C; (3) t deciding the movement towards the promising descent direction; (4)Laplace(0, b) distribution parameter b; (5) cooling ratio μ; (6) Tournament size; (7)the initial value of *threshold* D_s. Here N_P and N_C are set according to Table I. The other parameters settings are given as follows: t, b, the initial value of *threshold* D_s, μ and tournament size are fixed to 2.5, 0.5, 0.1, 0.9999, 3 for all problems, respectively.

All 25 runs for every test problem achieve within the error value 10^{-7}. The *least, mean* and *most FEs* in 25 runs for every problem using the proposed *rc*-SBGA+EX-FPDD are recorded in the Table I, respectively. As shown in Table I, the proposed approach has excellent performance for the problem of parameter optimization in case of D=30~100.

To evaluate the performance of the proposed algorithm, we compare our method with three existing real-coded genetic algorithms: MMG+UNDX [8], MMG+SPX [9], SGA +LX+NUM [11], whose parameter settings are kept same as [15]. Table II record the mean FEs in 25 runs to achieve the fixed accuracy 10^{-7} using different algorithms in case of D=30. The comparing result expresses that the proposed algorithm is superior to its competitor. From Table I and Table II, we can know that the proposed algorithm performs quite well and outperforms other algorithms in 15 tested problems.

IV. CONCLUSIONS

In this study we defined an efficient real-coded genetic algorithm called *rc*-SBGA+EX-FPDD-LX. The experiments were performed on a set of 15 benchmark problems available in global optimization literature. Three different algorithms were employed to compare the performance of the proposed algorithm. Simulation results showed that the proposed *rc*-SBGA+EX-FPDD-LX has excellent performance for the problem of parameter optimization.

ACKNOWLEDGEMENTS

This work was supported by the startup scientific research funds of Chongqing Technology and Business University under Contract 2011-56-05.

REFERENCES

[1] A.H. Wright, "Genetic algorithms for real parameter optimization", in: G.J.E. Rawlins (Ed.), Foundations of Genetic Algorithms I, Morgan Kaufmann, San Mateo, 1991, pp. 205-218.

[2] N.J. Radcliffe, "Equivalence class analysis of genetic algorithms", Complex Systems 2 (5) , 1991, pp.183-205.

[3] Z. Michalewicz, "Genetic Algorithms + Data Structures = Evolution Programs", Springer-Verlag, New York, 1992.

[4] L.J. Eshelman, J.D. Schaffer, "Real-coded genetic algorithms and interval schemata", in: D.L. Whitley (Ed.), Foundation of Genetic Algorithms II, Morgan Kaufmann, San Mateo, CA, 1993, pp. 187-202.

[5] H. Muhlebein, D. Schlierkamp-Voosen, "Predictive models for breeder genetic algorithms in continuous parameter optimization", Evolutionary Computation 1 (1) , 1993, pp. 25-49.

[6] H.M. Voigt, H. Muhlenbein, D. Cvetkovic, "Fuzzy recombination for the breeder genetic algorithms", in: L.J. Eshelmen (Ed.), Proceedings of the 6th International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, 1995, pp. 104-111.

[7] K. Deb, R.B. Agrawal, "Simulated binary crossover for continuous search space", Complex Systems 9 , 1995, pp. 115-148.

[8] I. Ono, S. Kobayashi, "A real-coded genetic algorithm for function optimization using unimodal normal distribution crossover", in: T. Back (Ed.), Proceedings of the Seventh International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA,1997, pp. 246−253.

[9] S. Tsutsui, M. Yamamura, T. Higuchi, "Multi-parent recombination with simplex crossover in real-coded genetic algorithms", in: W. Banzhaf, J. Daida, A. Eiben, M. Garzon, V. Honavar, M. Jakiela, R. Smith (Eds.), Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1 1999), 1999, pp. 657−664.

[10] K. Deb, A. Anand, D. Joshi, "A computationally efficient evolutionary algorithm for real-parameter evolution", Evolutionary Computation Journal 10 (4) , 2002, pp. 371-395.

[11] K. Deep, M. Thakur, "A new crossover operator for real coded genetic algorithms", Applied Mathematics and Computation 188 , 2007, 895-911.

[12] S.kobayashi, "The Frontiers of Real-Coded Genetic Algorithms", Journal of Japanese Society for Artificial Intelligence , Vol.24, No.1, 2009, pp.128-143. In Japanese.

[13] Z.Q.CHEN, R.L.Wang, "An Efficient Real-coded Genetic Algorithm for Real-Parameter Optimization", The 6th International Conference on Natural Computation (ICNC'10), 8.2010, pp2276-2280.

[14] Z.Q.CHEN, R.L.Wang, "Two Efficient Real-Coded Genetic Algorithms for Real Parameter Optimization, International Journal of Innovative Computing", Information and Control, Vol.7, No.8, 2011.

[15] Z.Q.CHEN, R.L.Wang, "A New Framework with FDPP-LX Crossover for Real-Coded Genetic Algorithm", IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E94.A (2011) , No. 6 pp.1417-1425.

[16] R. L. Wang, "A genetic algorithm for subset sum problem", Neurocomputing, vol.57, 2004, pp.463-468.

[17] Z.Q.CHEN, R.L.Wang, "Solving the Bipartite Subgraph Problem using Genetic Algorithm with Conditional Genetic Operators", IEEJ Transactions on Electrical and Electronic Engineering, Vol.4, 2009, pp.663-667.

[18] Z.Q.CHEN, R.L.Wang, "Solving m-Way Graph Partitioning Problem Using Genetic Algorithm", IEEJ Transactions on Electrical and Electronic Engineering, Vol. 6. No. 5, 2011, pp.483-489.