Novel Multimodal Problems and Differential Evolution with Ensemble of Restricted Tournament Selection

Bo-Yang Qu, Ponnuthurai Nagaratnam Suganthan, *Snr Member, IEEE*

***Abstract*—Multi-modal optimization refers to locating not only one optimum but a set of locally optimal solutions. Niching is an important technique to solve multi-modal optimization problems. The ability of discover and maintain multiple niches is the key capability of these algorithms. In this paper, differential evolution with an ensemble of restricted tournament selection (ERTS-DE) algorithm is introduced to perform multimodal optimization. The algorithms is tested on 15 newly designed scalable benchmark multi-modal optimization problems and compared with the crowding differential evolution (Crowding-DE) in the literature. As shown by the experimental results, the proposed algorithm outperforms the Crowding-DE on the novel scalable benchmark problems.**

I. Introduction

In real world optimization, many engineering problems can be classified as multi-modal problems, such as classification problems in machine learning [1] and inversion of teleseismic waves [2]. The aim is to locate several globally or locally optimal solutions and then to choose the most appropriate solution considering practical issues. In recent years, evolutionary algorithms (EA) with various niching techniques have been successfully applied to solve multi-modal optimization problems. The earliest niching approach was proposed by Cavicchio [3]. Subsequently, many other niching methods, such as crowding [4] and clearing [5], have also been proposed.

Differential evolution is a very powerful optimization technique compared with other EAs such as genetic algorithms and evolutionary programming. Like other EAs, DE is also a population-based algorithm. Although DE has been proven to be effective in locating one globally optimal solution [6], the basic DE is not efficient for solving multi-modal optimization problems [7]. Some work has been done to extend the DE to solve multi-modal problems [8]-[9]. Thomsen proposed a Crowding-DE [7] and showed that Crowding-DE outperformed a DE based fitness sharing algorithm. In this paper, DE with an ensemble of crowding and restricted tournament selection (ECRTS-DE) is proposed and compared with the Crowding-DE on a set of newly designed scalable multi-modal optimization problems.

The remainder of this paper is structured as follows. Section II provides a brief overview of differential evolution, crowding and restricted tournament selection as well as the Crowding-DE algorithm. In Section III, the proposed ERTS-DE algorithm is introduced. The definition of newly developed problems and the results of the experiments are presented in Sections IV and V, respectively. Finally, the paper is concluded in Section VI.

Manuscript received February 7, 2010. This work was supported by the Nanyang Technological University.

Authors are with Nanyang Technological University, School of Electrical and Electronic Engineering, Singapore, 639798 (phone: 65-67905404; fax: 65-67933318; emails: E070088@ntu.edu.sg , epnsugan@ntu.edu.sg).

II. Crowding Differential Evolution

This section introduces the differential evolution algorithm, crowding and restricted tournament selection based niching algorithms and crowding differential evolution algorithm which is a DE and crowding based multimodal optimization algorithm.

A. Differential Evolution

The differential evolution (DE) algorithm was first introduced by Storn and Price [10] and widely used in different areas [11]-[13]. The four major steps involved in DE are known as, initialization, mutation, recombination and selection. In the mutation operation, one of the following strategies is used [14]:

DE/rand/1: $v_p = x_{r1} + F \cdot (x_{r2} - x_{r3})$

DE/best/1: $v_p = x_{best} + F \cdot (x_{r1} - x_{r2})$

DE/current-to-best/2:

$v_p = x_p + F \cdot (x_{best} - x_p + x_{r1} - x_{r2})$

DE/best/2: $v_p = x_{best} + F \cdot (x_{r1} - x_{r2} + x_{r3} + x_{r4})$

DE/rand/2: $v_p = x_{r1} + F \cdot (x_{r2} - x_{r3} + x_{r4} - x_{r5})$

where *r1*, *r2*, *r3*, *r4*, *r5* are mutually different integers randomly generated in the range [1, *NP* (population size)], *F* is the scale factor used to scale differential vectors. x_{best} is the solution with the best fitness value in the current population.

The crossover operation is applied to each pair of the generated mutant vector and its corresponding parent vector using the following equations:

$$u_{p,i} = \begin{cases} v_{p,i} & \text{if } rand_i \le CR \\ x_{p,i} & \text{otherwise} \end{cases}$$

where u_p is the offspring vector. *CR* is the crossover rate which is a user-specified constant.

B. Crowding and Restricted Tournament Selection

Crowding **Error! Reference source not found.** was introduced by De Jong in 1975 and extended to restricted tournament selection by Harik [15]. It differs from a simple evolutionary algorithm in the way of replacing individuals in the current population by offspring. For crowding and restricted tournament selection, in order to compare the offspring with the current population, a random set of *w* (window size) individuals are selected from the current population and the nearest to the offspring is determined by Euclidean distance measure. Finally, this nearest individual is replaced by the offspring if its fitness value is worse than the offspring's fitness value. This process is repeated for all the offspring in each generation. Crowding and restricted tournament selection methods are effective in maintaining the diversity of the population, which is important in multi-modal optimization.

C. *Crowding DE*

Crowding DE was first introduced by Thomsen to solve multi-modal optimization problems [7]. In Crowding DE, the fitness value of an offspring is compared with that of the nearest individual in the current population (*w* is same as the population size). The steps of Crowding DE are shown in Table I.

Table I. Crowding DE algorithm

Step 1	Use the basic DE to produce *NP* (population size) offspring.
	For *i*=1:*NP*
Step 2	Calculate the Euclidean distance values of the offspring(*i*) to the other individuals in the DE Population.
Step 3	Compare the fitness value of offspring(*i*) and the fitness value of the individual that has the smallest Euclidean distance. The offspring will replace the individual if it is fitter than the individual.
	Endfor
Step 4	Stop if the termination criterion is met, otherwise go to step 1.

III. ERTS-DE

As we know, there is one key parameter *w* that controls the performance of restricted tournament selection (Crowding DE is a special case with *w*=*NP*). According to the "No free lunch" theorem [16], it is impossible to find one parameter value that can be better than all other parameter values for all problems. Motivated by this observation, an ensemble of restricted tournament selection DE is proposed using parallel populations with different window sizes. In other words, different populations are used. In this paper, two populations with two different window sizes are used. More populations can be used, if additional different parameters or different niching algorithms are used. Each population will generate its own offspring population. The populations need not only compete with their own offspring, but also the offspring generated by the other population. In this way, the algorithm will always keep the offspring that was generated by the more suitable parameter leading to a better performance. The flowchart of the ERTS-DE algorithm is shown in Fig. 1.

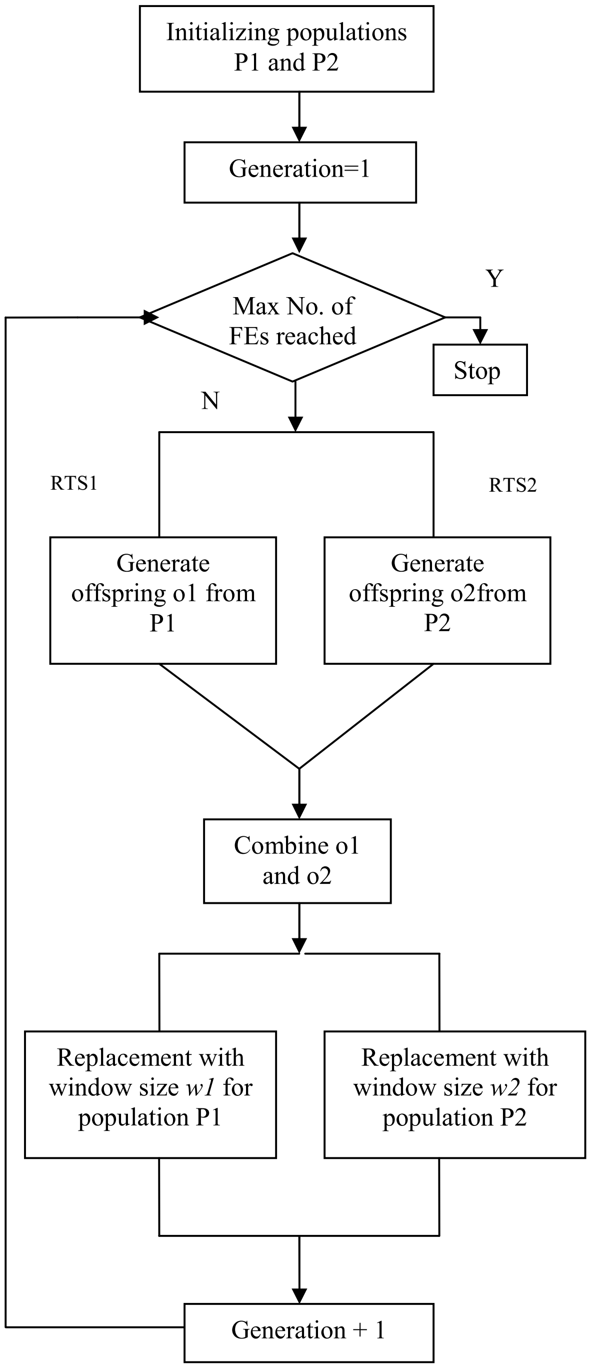
Initializing populations P1 and P2
Generation=1
Max No. of FEs reached
Y
Stop
N
RTS1
RTS2
Generate offspring o1 from P1
Generate offspring o2from P2
Combine o1 and o2
Replacement with window size *w1* for population P1
Replacement with window size *w2* for population P2
Generation + 1

Fig. 1. Flowchart of the ERTS-DE algorithm

IV. PROBLEM DEFINITIONS

There are several multi-modal benchmark problems available in the literature. However, these problems are relatively easy and many algorithms can solve them perfectly. There is also a lack of scalable multi-modal problems. Therefore, it is difficult to differentiate the performance of advanced algorithms. To overcome these problems, a new set of scalable multi-modal problems is designed in this article by making use of composition functions in [17]. All the test functions are maximization problems with equal globally optimal fitness value of 0. The composition functions are defined as follows:

$F(x)$: new composition function

$f_i(x)$: ith basic function used to construct the composition function.

n: number of basic functions (number of optima)

D: dimensions (can be chosen from 1-100)

M_i : linear transformation matrix for each $f_i(x)$

o_i : new shifted optima position for each $f_i(x)$

$$F(x)=\sum_{i=1}^{n}\left\{w_i*[f_i'((x-o_i)/\lambda_i*M_i)]\right\}$$

w_i : weight value for each $f_i(x)$, calculated as follow:

$$w_i=\exp\left(-\frac{\sum_{k=1}^{D}(x_k-o_{ik})}{2D\sigma_i^2}\right)$$

$$w_i=\begin{cases} w_i & w_i==\max(w_i)\\ w_i*(1-\max(w_i).\hat{}10) & w_i\neq\max(w_i)\end{cases}$$

then normalize the weight $w_i=w_i/\sum_{i=1}^{n}w_i$

σ_i : used to control each $f_i(x)$'s coverage range.

λ_i : used to stretch compress the function.

$f_i'(x)=C*f_i(x)/\left|f_{\max i}\right|$, C is a predefined constant.

$\left|f_{\max i}\right|$ is estimated using: $\left|f_{\max i}\right|=f_i((x'/\lambda_i)*M_i), x'=[5,5,...,5]$

Composition Function 1 (F1, *n*=8)

$f_{1-2}(x)$: Rastrigin's Function

$$f_i(x)=\sum_{i=1}^{D}(x_i^2-10\cos(2\pi x_i)+10)$$

$f_{3-4}(x)$: Weierstrass Function

$$f_i(x)=\sum_{i=1}^{D}\left(\sum_{k=0}^{k\max}[a^k\cos(2\pi b^k(x_i+0.5))]-D\sum_{k=0}^{k\max}[a^k\cos(2\pi b^k\bullet 0.5)]\right)$$

a=0.5, b=3, $k_{\max}$=20

$f_{5-6}(x)$: Griewank's Function

$$f_i(x)=\sum_{i=1}^{D}\frac{x_i^2}{4000}-\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$$

$f_{7-8}(x)$: Sphere Function

$$f_i(x)=\sum_{i=1}^{D}x_i^2$$

$\sigma_i=1$ for all i

λ = [1, 1, 10, 10, 5/60, 5/60, 5/32, 5/32]

M_i : are all identity matrices

These formulas are basic functions; shift and rotation should be added to these functions. Take f_1 as an example, the following function should be evaluated:

$$f_i(z)=\sum_{i=1}^{D}(z_i^2-10\cos(2\pi z_i)+10)$$

where $z=((x-o_i)/\lambda_1)*M_1$.

Composition Function 2 (F2 *n*=6)

$f_{1-2}(x)$: Griewank's Function

$f_{3-4}(x)$: Weierstrass Function

$f_{5-6}(x)$: Sphere Function

$\sigma_i=1$ for all i

λ = [1, 1, 10, 10, 5/60, 5/60,]

M_i : are all identity matrices

Composition Function 3 (F3 *n*=6)

$f_{1-2}(x)$: Rastrigin's Function

$f_{3-4}(x)$: Griewank's Function

$f_{5-6}(x)$: Sphere Function

$\sigma_i=1$ for all i

λ = [1, 1, 10, 10, 5/60, 5/60,]

M_i : are all identity matrices

Composition Function 4 (F4 *n*=6)

$f_{1-2}(x)$: Rastrigin's Function

$f_{3-4}(x)$: Weierstrass Function

$f_{5-6}(x)$: Griewank's Function

$\sigma_i=1$ for all i

λ = [1, 1, 10, 10, 5/60, 5/60,]

M_i : are all identity matrices

Composition Function 5 (F5 *n*=6)

$f_{1-2}(x)$: Rastrigin's Function

$f_{3-4}(x)$: Weierstrass Function

$f_{5-6}(x)$: Sphere Function

$\sigma_i=1$ for all i

λ = [1, 1, 10, 10, 5/60, 5/60,]

M_i : are all identity matrices

Composition Function 6 (F6 *n*=6)

$f_{1-2}(x)$: F8F2 Function

$$F8(x)=\sum_{i=1}^{D}\frac{x_i^2}{4000}-\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$$

$$F2(x)=\sum_{i=1}^{D-1}(100(x_i^2-x_{i+1})^2+(x_i-1)^2)$$

$$f_i(x) = F8(F2(x_1, x_2)) + F8(F2(x_2, x_3)) + ... + F8(F2(x_{D-1}, x_D) + F8(F2(x_{D,} x_1))$$

$f_{3-4}(x)$: Weierstrass Function

$f_{5-6}(x)$: Griewank's Function

$\sigma = [1,1,1,1,1,2]$,

$\lambda = [5*5/100;5/100;5*1;1;5*1;1]$

M_i: are all orthogonal matrix

Composition Function 7 (F7 *n*=6)

$f_{1-2}(x)$: Rotated Expanded Scaffer's F6 Function

$$F(x, y) = 0.5 + \frac{(\sin^2(\sqrt{x^2 + y^2}) - 0.5)}{(1 + 0.001(x^2 + y^2))^2}$$

$$f_i(x) = F(x_1, x_2) + F(x_2, x_3) + ... + F(x_{D-1}, x_D) + F(x_D, x_1)$$

$f_{3-4}(x)$: F8F2 Function

$f_{5-6}(x)$: Weierstrass Function

$\sigma = [1,1,1,1,1,2]$,

$\lambda = [5;10;5;1;5*5/100;5/100]$

M_i: are all orthogonal matrix

Composition Function 8 (F8 *n*=6)

$f_{1-2}(x)$: Rotated Expanded Scaffer's F6 Function

$f_{3-4}(x)$: F8F2 Function

$f_{5-6}(x)$: Griewank's Function

$\sigma = [1,1,1,1,1,2]$,

$\lambda = [5*5/100;5/100;5*1;1;5*1;1]$

M_i: are all orthogonal matrix

Composition Function 9 (F9 *n*=6)

$f_{1-2}(x)$: Rotated Expanded Scaffer's F6 Function

$f_{3-4}(x)$: Weierstrass Function

$f_{5-6}(x)$: Griewank's Function

$\sigma = [1,1,1,1,1,2]$,

$\lambda = [5;10;5*5/100;5/100;5;1]$

M_i: are all orthogonal matrix

Composition Function 10 (F10 *n*=6)

$f_{1-2}(x)$: Rastrigin's Function

$f_{3-4}(x)$: F8F2 Function

$f_{5-6}(x)$: Weierstrass Function

$\sigma = [1,1,1,1,1,2]$,

$\lambda = [5;10;5*5/100;5/100;5;1]$

M_i: are all orthogonal matrix

Composition Function 11 (F11 *n*=8)

$f_{1-2}(x)$: Rastrigin's Function

$f_{3-4}(x)$: F8F2 Function

$f_{5-6}(x)$: Weierstrass Function

$f_{7-8}(x)$: Griewank's Function

$\sigma = [1,1,1,1,1,2,2,2]$,

$\lambda = [5;1;5;1;50;10;5*5/200;5/200]$

M_i: are all orthogonal matrix

Composition Function 12 (F12 *n*=8)

$f_{1-2}(x)$: Rotated Expanded Scaffer's F6 Function

$f_{3-4}(x)$: F8F2 Function

$f_{5-6}(x)$: Weierstrass Function

$f_{7-8}(x)$: Griewank's Function

$\sigma = [1,1,1,1,1,2,2,2]$,

$\lambda = [5*5/100;5/100;5;1;5;1;50;10]$

M_i: are all orthogonal matrix

Composition Function 13 (F13 *n*=10)

$f_{1-2}(x)$: Rotated Expanded Scaffer's F6 Function

$f_{3-4}(x)$: Rastrigin's Function

$f_{5-6}(x)$: F8F2 Function

$f_{7-8}(x)$: Weierstrass Function

$f_{9-10}(x)$: Griewank's Function

$\sigma = [1,1,1,1,1,2,2,2,2,2]$,

$\lambda = [5*5/100;5/100;5;1;5;1;50;10;5*5/200;5/200]$

M_i: are all orthogonal matrix

Composition Function 14 (F14 *n*=10)

All settings are the same as F13, except M_i's condition numbers are [10 20 50 100 200 1000 2000 3000 4000 5000]

Composition Function 15 (F15 *n*=10)

$f_1(x)$: Weierstrass Function

$f_2(x)$: Rotated Expanded Scaffer's F6 Function

$f_3(x)$: F8F2 Function

$f_4(x)$: Ackley's Function

$$f_i(x) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + e$$

$f_5(x)$: Rastrigin's Function

$f_6(x)$: Griewank's Function

$f_7(x)$: Non-Continuous Expanded Scaffer's F6 Function

$$F(x, y) = 0.5 + \frac{(\sin^2(\sqrt{x^2 + y^2}) - 0.5)}{(1 + 0.001(x^2 + y^2))^2}$$

$$f_i(x) = F(y_1, y_2) + F(y_2, y_3) + ... + F(y_{D-1}, y_D) + F(y_D, y_1)$$

$$y_i = \begin{cases} x_j & |x_j| < 1/2 \\ round(2x_j)/2 & |x_j| > 1/2 \end{cases} \quad for\ j = 1, 2, ..., D$$

$$round(x) = \begin{cases} a - 1 & if\ x \le 0 \,\&\, b \ge 0.5 \\ a & if\ b < 0.5 \\ a + 1 & if\ x > 0 \,\&\, b \ge 0.5 \end{cases}$$

$f_8(x)$: Non-Continuous Rastrigin's Function

$$f_i(x) = \sum_{i=1}^{D}(y_i^2 - 10\cos(2\pi y_i) + 10)$$

$$y_i = \begin{cases} x_j & |x_j| < 1/2 \\ round(2x_j)/2 & |x_j| > 1/2 \end{cases} for\ j = 1,2,...,D$$

$f_9(x)$: High Conditioned Elliptic Function

$$f(x) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} x_i^2$$

$f_{10}(x)$: Sphere Function with Noise in Fitness

$$f_i(x) = (\sum_{i=1}^{D} x_i^2)(1 + 0.1|N(0,1)|)$$

n=10

$\sigma_i = 2$ for all i

$\lambda = [10; 5/20; 1; 5/32; 1; 5/100; 5/50; 1; 5/100; 5/100]$

M_i are all rotation matrices, condition number are [100 50 30 10 5 5 4 3 2 2];

Table II. Test function properties

Test Function	No. of equal global optima	No. of basic function used	Rotation used (Y/N)
F1	8	4	N
F2	6	3	N
F3	6	3	N
F4	6	3	N
F5	6	3	N
F6	6	3	Y
F7	6	3	Y
F8	6	3	Y
F9	6	3	Y
F10	6	3	Y
F11	8	4	Y
F12	8	4	Y
F13	10	5	Y
F14	10	5	Y
F15	10	10	Y

V. Experiments and Results

For the simulations, Matlab 7.1 is used as the programming language. The configurations of the computer are Intel Pentium® 4 CPU 3.00 GHZ, 2 GB of memory. As the test problems are relatively complex and the number of optima is large, a large population size should be used. The population size is set 600 for D=10 and 1200 for D=30. The maximum number of generation is 500 for D=10 and 1000 for D=30. Therefore, the maximum number of function evaluations will be the population size multiplied by number of generations for both algorithms. The parameters used in the algorithms are list as below:

F=0.9, CR=0.1,

Two experiments are conducted as follows:

1. D=10, Test Functions: F1-F15
2. D=30, Test Functions: F1-F5

For comparison, the following two criteria are used:

1. Number of optima found [18]
2. The best value found

An optimum is considered to be found if there exists a solution in the population within the tolerated Euclidean distance to that optimum. The tolerance for all problems is set to 0.1. All problems are run for 25 times. The results are shown in Tables III-V. Since for D=30, both algorithms are not able to locate any global optimum, the number of optima found for these problems will be zero. As can been seen from the results, the proposed algorithm outperforms the Crowding-DE on all benchmark problems.

Table III Comparison of number of optima found (D=10)

Test Function		Crowding DE	
F1	Best	1	2
	Worst	0	0
	Mean	**0.1**	**1.2**
	Std	0.3162	0.7888
F2	Best	3	3
	Worst	1	3
	Mean	**2**	**3**
	Std	0.6667	0
F3	Best	1	3
	Worst	0	1
	Mean	**0.1**	**2.1**
	Std	0.3162	0.5676
F4	Best	1	2
	Worst	0	1
	Mean	**0.7**	**1.5**
	Std	0.4831	0.5270
F5	Best	2	4
	Worst	0	2
	Mean	**1.4**	**2.9**
	Std	0.6992	0.5677
F6	Best	0	3
	Worst	0	1
	Mean	**0**	**2.4**
	Std	0	0.6992
F7	Best	0	0
	Worst	0	0
	Mean	**0**	**0**
	Std	0	0
F8	Best	0	1
	Worst	0	0
	Mean	**0**	**0.2**
	Std	0	0.4216
F9	Best	1	2
	Worst	0	0
	Mean	**0.1**	**1.1**
	Std	0.3162	0.8756
F10	Best	0	0
	Worst	0	0
	Mean	**0**	**0**
	Std	0	0
F11	Best	1	1
	Worst	0	1
	Mean	**0.4**	**1**
	Std	0.5164	0
F12	Best	0	1
	Worst	0	0
	Mean	**0**	**0.2**
	Std	0	0.4216
F13	Best	0	1
	Worst	0	0
	Mean	**0**	**0.2**
	Std	0	0.4216
F14	Best	0	0
	Worst	0	0
	Mean	**0**	**0**
	Std	0	0
F15	Best	0	2
	Worst	0	0
	Mean	**0**	**0.5**
	Std	0	0.7071

Table IV. Comparison of best value found (*D*=10)

Test Function		Crowding DE	ERTS-DE
F1	Best	-0.3218	-0.1777
	Worst	-1.9420	-0.8395
	Mean	**-1.1618**	**-0.4367**
	Std	0.4773	0.1912
F2	Best	-0.0480	-0.0074
	Worst	-0.1316	-0.0339
	Mean	**-0.0911**	**-0.0239**
	Std	0.0272	0.0077
F3	Best	-0.0987	-0.0306
	Worst	-0.3566	-0.0904
	Mean	**-0.1954**	**-0.0615**
	Std	0.0834	0.0225
F4	Best	-26.4690	-11.5970
	Worst	-39.4740	-27.5910
	Mean	**-31.7634**	**-18.4755**
	Std	4.5066	5.5951
F5	Best	-0.0999	-0.0137
	Worst	-0.2112	-0.0832
	Mean	**-0.1292**	**-0.0377**
	Std	0.0345	0.0215
F6	Best	-2.2706	-0.1270
	Worst	-6.5206	-0.9615
	Mean	**-4.6309**	**-0.5758**
	Std	1.2340	0.2838
F7	Best	-43.1750	-3.6100
	Worst	-114.7600	-18.0540
	Mean	**-64.6664**	**-11.0893**
	Std	20.8064	4.8449
F8	Best	-7.2632	-1.8996
	Worst	-20.1200	-5.8679
	Mean	**-13.1706**	**-3.7509**
	Std	3.7754	1.2317
F9	Best	-2.7016	-0.7779
	Worst	-10.0240	-2.6185
	Mean	**-5.9759**	**-1.7434**
	Std	2.0342	0.6195
F10	Best	-21.2510	-1.6850
	Worst	-40.4930	-3.6430
	Mean	**-29.6469**	**-2.5746**
	Std	6.5543	0.5745
F11	Best	-14.2310	-2.6533
	Worst	-20.9890	-11.0240
	Mean	**-17.7898**	**-5.7436**
	Std	2.2209	2.3804
F12	Best	-3.9807	-1.1163
	Worst	-20.2220	-4.8248
	Mean	**-14.1562**	**-2.0342**
	Std	5.6613	1.1223
F13	Best	-9.2783	-3.0312
	Worst	-30.0480	-12.7220
	Mean	**-23.6060**	**-6.2305**
	Std	6.6086	3.2660
F14	Best	-38.1120	-3.2552
	Worst	-81.2650	-75.5100
	Mean	**-56.5465**	**-30.5935**
	Std	16.0574	56.4823
F15	Best	-9.4756	-1.2842
	Worst	-46.7710	-5.0021
	Mean	**-21.6305**	**-2.9632**
	Std	11.6107	1.0729

Table V. Comparison of best value found (*D*=30)

Test Function		Crowding DE	ERTS-DE
F1	Best	-5.1271	-2.9273
	Worst	-7.8151	-4.5686
	Mean	**-6.2561**	**-3.8060**
	Std	0.8332	0.6433
F2	Best	-2.6091	-1.1209
	Worst	-3.7313	-1.6555
	Mean	**-3.1617**	**-1.3865**
	Std	0.4195	0.1907
F3	Best	-2.4809	-0.7774
	Worst	-4.3563	-2.0588
	Mean	**-3.7140**	**-1.6436**
	Std	0.6220	0.3615
F4	Best	-72.2550	-59.2930
	Worst	-74.8780	-72.5570
	Mean	**-73.9101**	**-64.6000**
	Std	0.9569	4.5664
F5	Best	-2.2144	-1.0255
	Worst	-3.9658	-1.6979
	Mean	**-3.0425**	**-1.4246**
	Std	0.5437	0.1853

VI. Conclusion

In this paper, differential evolution algorithm with an ensemble of restricted tournament selection-based niching algorithm is proposed to overcome the difficulty of choosing window size parameter when solving multi-modal optimization problems. The proposed algorithm is compared with the Crowding-DE on a set of newly designed scalable multi-modal problems. As we can see from the result, the proposed algorithm outperforms the Crowding-DE on all the test problems.

References

[1] S. W. Mahfoud, "Niching methods for genetic algorithms," Ph.D. dissertation, Urbana, IL, USA, 1995. [Online]. Available: citeseer.ist.psu.edu/mahfoud95niching.html.

[2] K. Koper and M. Wysession, "Multimodal function optimization with a niching genetic algorithm: A seis-mological example," *Bulletin of the Seismological Society of America*, vol. 89, pp. 978-988, 1999.

[3] D. J. Cavicchio, "Adaptive search using simulated evolution," Ph.D. dissertation, University of Michigan, Ann Arbor, 1970.

[4] K. A. De Jong, "An analysis of the behavior of a class of genetic adaptive systems, Ph.D. dissertation, University of Michigan, 1975.

[5] A. Pétrowski, "A clearing procedure as a niching method for genetic algorithms", *Proc. of the IEEE Int. Conf. on Evolutionary Computation*, New York, USA, 1996, pp. 798–803.

[6] K. Price. "An introduction to differential evolution." *New Ideas in Optimization,* pages 79-108, 1999.

[7] R. Thomsen. "Multi-modal optimization using crowding-based differential evolution. In *Proceedings of the 2004 Congress on Evolutionary Computation,* volumn 2, pages 1382-1389, 2004.

[8] D. Zaharie. "Extensions of differential evolution algorithms for multimodal optimization," *In Proceedings of SYNASC'04, 6th International Symposium of Symbolic and Numeric Algorithms for Scientific Computing*, pages 523{534, 2004.

[9] Z. Hendershot. "A differential evolution algorithm for automatically discovering multiple global optima in multidimensional, discontinues spaces. *In Proceedings of MAICS 2004, Fifteenth Midwest Artificial Intelligence and Cognitive Sciences Conference*, pages 92{97, 2004.

[10] R. Storn and K. V. Price, "Differential evolution-Asimple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization,* vol. 11, pp. 341-359, 1995.

[11] S. Das, A. Konar and U. K. Chakraborty, " Annealed differential evolution," In *Proc. IEEE Congress on Evolutionary Computation, Singapore,* 2007.

[12] U. K. Chakraborty, S. Das and A. Konar, "Differential evolution with local neighborhood, In *Proc. IEEE Congress on Evolutionary Computation,* Canada, 2006.

[13] V. L. Huang, A. K. Qin and P. N. Suganthan, "Self-adaptive Differential Evolution Algorithm for Constrained Real-Parameter Optimization", In *Proc. IEEE Congress on Evolutionary Computation,* Canada, 2006.

[14] X. Q. Chen, Z. X. Hou, and J. X. Liu, "Multi-objective optimization with modified pareto differential evolution," in *International Conference on Intelligent Computation Technology and Automation,*, Piscataway, NJ 08855-1331, United States, 2008.

[15] G. Harik, "Finding multiple solutions in problems of bounded difficulty," University of Illinois at Urbana-Champaign, 1994, IIIiGAL Report No. 94002.

[16] D H. Wolpert, W. G. Macready, No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation 1 (1997) 67–82.

[17] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. –P. Chen, A. Auger and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," *Technical Report,* Nanyang Technological University, May 2005.

[18] J. Gan, K. Warwick, "A variable radius niching technique for speciation in genetic algorithms," In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2000)*, San Francisco, USA, 2000, pp. 96–103.