WCCI 2012 IEEE World Congress on Computational Intelligence
June, 10-15, 2012 - Brisbane, Australia IEEE CEC

An Improved Differential Evolution for Constrained Optimization with Dynamic Constraint-Handling Mechanism

Zhihui Li, Zhigang Shang, J.J.Liang
School of Electrical Engineering
Zhengzhou University
Zhengzhou, China
lizhrain@zzu.edu.cn, zhigang_shang@zzu.edu.cn

Ben Niu
College of Management
Shenzhen University
Shenzhen, China
liangjing@zzu.edu.cn, drniuben@gmail.com

Abstract—**In this paper, an improved Differential Evolution (DE) with a self-adaptive strategy to determine the control parameters is proposed to solve constrained real-parameter optimization, combined with the dynamic constraint-handling mechanism. It is implemented by restating the single-objective constrained optimization as a set of single-objective unconstrained problems and dynamically assigning to the individual adaptively as its fitness, and the self-adaptive strategy of control parameters based on the intrinsic structure analysis of differential vectors is use to solve each unconstrained optimization problem individually. This approach is tested on a suit of test problems proposed for CEC2010 competition and special session on single objective constrained real-parameter optimization. The result indicates the combination of dynamic constraint-handling mechanism and self-adaptation of control parameters in DE will outperform using the former solely for constrained optimization.**

Keywords- Differential Evolution; control parameters; self-adaptive; dynamic constraint-Handling constrained optimization; intrinsic triangle

I. INTRODUCTION

Most of the practical problems can be reduced to optimization problems for solution. Therefore, solving optimization problems has now become a hot research topic. Optimization of constrained problems is an important area in the optimization field.

The so-called constrained optimization problem with inequality, equality constraints, generally, can be described as:

$$\begin{aligned} &\text{Minimize} && f(\mathbf{x}) \\ &\text{subject to:} && g_i(\mathbf{x}) \le 0, i = 1,...,q \\ & && h_j(\mathbf{x}) = 0, j = q+1,...,m \end{aligned} \quad (1)$$

Where x is the solution vector $\mathbf{x} = [x_1, x_2, ..., x_D]$ and for each variable x_i it satisfies a constrained boundary $x_i \in [x_{\min}, x_{\max}]^D$, D is dimensionality of the problem, q is the number of inequality constraints and m-q is the number of equality constraints. If we denote the whole search space as S, x is a feasible solution, if x satisfies all the constrained functions $g_i(x)$ and $h_j(x)$, thus all of the feasible solutions form the feasible region (FR). Otherwise x is an infeasible solution and the infeasible region (IFR) is the set of all the infeasible solutions. If we denote the feasible region *as F*, then it is clear that $F \in S$. If $g_i(\mathbf{x}) = 0$ and **x** is a feasible solution, the constraint $g_i(\mathbf{x})$ is called as *active constraints* at **x**. Equality constraints $h_j(\mathbf{x})$ are active at all feasible solutions. For the convenience of practice, the equality constraints are always transformed into the inequality form:

$$|h_j(\mathbf{x})| - \varepsilon \le 0 \quad (2)$$

where ε is the allowed tolerance and is set to 1e-4 in this paper.

Then the formula of the constrained optimization problem can be simplified as:

$$\begin{aligned} &\text{Minimize} && f(\mathbf{x}),\ \mathbf{x} \in [\mathbf{x}_{\min}, \mathbf{x}_{\max}]^D \\ &\text{subject to:} && g_i(\mathbf{x}) \le 0, i = 1,...,m \end{aligned} \quad (3)$$

Differential Evolution (DE) was first introduced by Rainer Storn and Kenneth Price to solve optimization problems mainly in continuous search spaces [1] [2]. DE is a heuristic approach based on genetic algorithm (GA) and mimics the principle of natural evolution for minimizing possibly nonlinear and non-differentiable continuous space functions.

Because of the virtues such as simple in principle, controlled by few parameters, good robustness etc, DE has attracted a growing number of scholars to concern about and has been successfully and widely applied in constrained optimization, nonlinear optimal control, neural network optimization, filter design, array antenna pattern and other diverse fields [3][4][5].

We have already proposed the Differential Evolution with Dynamic constraint-handling (DCDE), which has a better global search ability, to solve the constrained problems provided by CEC2010 [8]. But the parameter in DE is fixed and pre-set according to the test experiments. Some researchers have begun to consider some techniques such as self-adaptation to automatically find an optimal set of control parameters for DE, the typical example is work of Qin et al [6][7].

In this paper, an improved DE is incorporated into Dynamic constraint-handling to solve the constrained problem. Different

This paper is supported by National Natural Science Foundation of China (Grant NO. 60905039,71001072)

from the previous method, the parameter in DE is not fixed but self-adaptive.

The remainder of this paper is organized as follows: Section II gives description of DCDE. A detailed description of our approach to determine the control parameters of DE is given in Section III. After that, experimental results over a suite of 18 constrained numerical optimization problems provided by CEC2010 [9] are presented in Section IV. Finally, we conclude the paper with discussion of results and suggestions for future work in Section V.

II. Differential Evolution with Dynamic Constraint-Handling

As the DE algorithm is method suit to solve unconstrained optimization problem essentially, so in order to solve constrained optimization problems, Differential Evolution with Dynamic constraint-handling (DCDE) algorithm using rules-based constraint handling mechanism to restate the single-objective constrained optimization as a set of single-objective unconstrained problems, then the constrained optimization will be much simpler. The objective and constraint functions are assigned to the individuals adaptively as its fitness according to which has the maximal pressure needed to be optimized to guide the population to search for feasible region.

The algorithm description of DCDE is as follow:

A. Dynamic and random DE (drDE)

a) Initialization

The initial population chosen randomly with a uniform distribution for all the individuals within the search space constrained by the prescribed minimum and maximum parameter bounds.

For example, the initial value of the *j*th parameter in *i*th individual is generated by

$$x_j = x_{\min} + rand(0,1)\cdot(x_{\max} - x_{\min}) \quad j = 1,...,D \tag{4}$$

Where $rand(0,1)$ is a random variable with uniform distribution in range $[0,1]$.

b) Mutation

For each mutation operation, the modal is selected randomly from the strategy pool which include the strategy "DE/rand/2" "DE/best/2" and "DE/rand to best/2" as listed in Table I. drDE randomly utilize difference strategy, For each target vector x_i ($i = 1,2,...,NP$), a new mutation vector v_i is generated by adding the weighted difference to another vector, just as :

TABLE I. CHOOSEN DIFFERENTIAL STRATEGIES

Symbol	Differential Expression
DE/rand/2/bin	$\mathrm{v}_i = \mathrm{x}_{r1} + F(\mathrm{x}_{r2} + \mathrm{x}_{r3} - \mathrm{x}_{r4} - \mathrm{x}_{r5})$
DE/best/2/bin	$\mathrm{v}_i = \mathrm{x}_{best} + F(\mathrm{x}_{r1} + \mathrm{x}_{r2} - \mathrm{x}_{r3} - \mathrm{x}_{r4})$
DE/rand to best/bin	$\mathrm{v}_i = \mathrm{x}_i + F(\mathrm{x}_{best} - \mathrm{x}_i) + F(\mathrm{x}_{r2} - \mathrm{x}_{r3})$

c) Crossover

Crossover operation is to mix the mutated vector with the target vector to yield the so-called trial vector. The trial vector can be generated by

$$\mathrm{u}_{ij} = \begin{cases} \mathrm{v}_{ij} & if(rand(0,1) \le CR) or (j = j_{rand}) \\ \mathrm{x}_{ij} & if(rand(0,1) > CR) or (j \ne j_{rand}) \end{cases} \tag{5}$$

In drDE, for each crossover, CR is randomly selected from array [0.9,0.3] or [0.3,0.2,0.1] in different stage of the algorithm, j_{rand} is a randomly chosen index $\in [1, D]$ which can ensure at least one parameter be copied from v_i .

d) Selection

The greedy criterion is used to determine whether the trail vector u_i generated in generation G come into the population of next generation (G+1) or not. It is selected by using our followed mechanism.

B. Dynamic constraint-handling mechanism

Suppose that there are *m* constraints, and then the number of objective functions needed to be optimized is $m+1$, i.e. $F = [f(\mathrm{x}), g_1(\mathrm{x}), g_2(\mathrm{x}), ..., g_m(\mathrm{x})]$

Define the pressure

$$p_i = \frac{\sum_{j=1}^{NP} (g_i(\mathbf{x}_j) > 0)}{NP} \quad , i=1, 2,..., m \tag{6}$$

$$fp = 1 - \bar{\mathrm{p}}, \mathbf{p} = [p_1, p_2, ..., p_m] \tag{7}$$

$$gp_i = \bar{\mathrm{p}} \cdot (p_i / \sum_{i=1}^{m} p_i) \tag{8}$$

thus $fp + \sum_{i=1}^{m} gp_i = 1$

The first step is using roulette selection according to *fp* and *gp_i* to assign the objective function or a single constraint as a target to each individual. Then according to the target to do the optimize operations [10].

While updating the individual, the following comparison criteria is used：

a) If the target of optimization is *k*th constraint $g_k(\mathrm{x})$ and the constraint hasn't been satisfied, the offspring wins if $g_k(\mathrm{u}_i) < g_k(\mathbf{x}_i)$ or $sumg(\mathrm{u}_i) < sumg(\mathbf{x}_i)$ or $sumg(\mathrm{u}_i) = sumg(\mathbf{x}_i)\ \&\ f(\mathrm{u}_i) < f(\mathbf{x}_i)$ (9)

where $$sumg(\mathbf{x}_i) = \sum_{k=1}^{m} (ratio_k \cdot (g_k(\mathbf{x}_i) > T_k) \cdot g_k(\mathbf{x}_i)) \tag{10}$$

$$ratio_k = \frac{1/g_k \max}{\sum_{k=1}^{m} (1/g_k \max)} \qquad k = 1,...,m \tag{11}$$

$$T_k = 0.5 \cdot (1 - \frac{fitcount}{0.5 \cdot Max\_FEs}) \cdot g_k \max \tag{12}$$

(*Max_FEs:* Max fitness evaluations, stop criterion)

The effect of Ratio is to balance the impacts of different constraints. The goal of setting parameter T is to find an acceptable region (AR) with a relaxed constraint which include the feasible region (FR) with a strict constraint, such that the AR can be shrunk.

In this way, the constraints that are more difficult will have more individuals work for it, while the easier ones will have less or even no individual working for it. So the search will focus on finding feasible solutions firstly, and then concentrate on improving the objective function. There will be more individuals evolve along the fitness increasing direction if more constraints are satisfied.

b) If the target of optimization is *k*th constraint $g_k(\mathrm{x})$ and the constraint has been satisfied, the offspring wins if

$$sumg(\mathrm{u}_i) < sumg(\mathbf{x}_i) \quad (13)$$

$$\text{or} \quad sumg(\mathrm{u}_i) = sumg(\mathbf{x}_i) \;\&\; f(\mathrm{u}_i) < f(\mathbf{x}_i) \quad (14)$$

c) If the target of optimization is $f(\mathrm{x})$, the offspring wins if

$$sumg(\mathrm{u}_i) < sumg(\mathbf{x}_i) \quad (15)$$

$$\text{or} \quad sumg(\mathrm{u}_i) = sumg(\mathbf{x}_i) \;\&\; f(\mathrm{u}_i) < f(\mathbf{x}_i) \quad (16)$$

III. INTRINSIC TRIANGLE IN DE (ITDE)

A. Intrinsic Triangle

In standard DE, each vector of population could be selected equally because of uniform sampling. In an arbitrary triangle composed by randomly selected three points (here we call them as base points), there are two directions to be chosen for the one base point is selected as the base vector, which determined by difference of other two points, so there are two possible target vector are formed and each with fifty percent possibility. If the mutation scale factor F is set to 1, there are three new points formed (we call them as target points, See Fig.1). The new triangle composed by the new three target points is defined as intrinsic triangle. It has some attractive and interesting character: the each base point is lies in the middle of differential vector between its two target points. So for the fixed region in the intrinsic triangle, these six points could ensure to sample uniformly, that is accord with maximum entropy criteria and ensure the acquired information of sampling is maximized. That is the reason why DE is so simple but has a outstanding perform to optimization question.

In Fig.1, the triangle ABC is composed by three base points, and the triangle A'B'C' is the intrinsic triangle of DE, in which each side is twice as long as the differential vector formed by the corresponding two base points

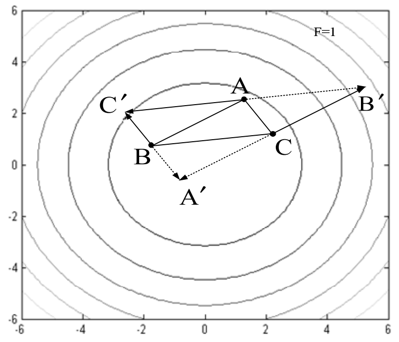

Figure 1. the intrinsic triangle of standard DE

B. Mutation Scale Facter Self-Adaption

When the mutation scale factor F is not equal to 1, there are six new target points formed so the intrinsic triangle could be destroyed. In order to use the sampling uniformity provided by the intrinsic triangle, which just represented the local space structure of sample points, the fitness function information of these local region should be integrated to determine the mutation direction and suitable step length for each base point.

Suppose the intrinsic triangle is very small, so the fitness function at three base points could be seemed as differentiable. For each base point, its directional derivatives at the direction of differential vector can be estimated by the projection of directional derivatives of fitness function at other known direction on the base vector mutation direction approximately. In Fig.2, the shadow zone could be thought as the confident interval of fitness function differential information at that direction, so the projection point could be as the candidate of mutation vector $V_{i,g}$. So we can consider the estimated directional derivatives information comprehensively to select the mutation vector. For example, the point P1 is the projection of vector $\overrightarrow{AB}$ on vector $\overrightarrow{AC'}$, the point P2 is the projection of vector $\overrightarrow{AC}$ on vector $\overrightarrow{AB'}$. Because the distance from $\overrightarrow{AB}$ to $\overrightarrow{AC'}$ is equal to the distance from $\overrightarrow{AC}$ to $\overrightarrow{AB'}$, so the difference of fitness function values at point A and B divided by the length of $\overrightarrow{AP_1}$ could be treated approximately as the directional derivative on direction $\overrightarrow{AC'}$ at the point A. The directional derivative on direction $\overrightarrow{AB'}$ at the point A could be estimated in same way. If the directional derivative on direction $\overrightarrow{AC'}$ is smaller than on direction $\overrightarrow{AB'}$ at point A, the point P1 could be selected as the mutation vector, i.e. the mutation scale factor F is equal to the length of $\overrightarrow{AP_1}$, so the control parameter F is self adaptive.

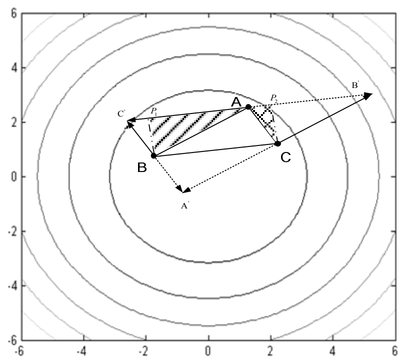

Figure 2. the projection of differentical vector on the search direction

If the intrinsic triangle is small enough and the fitness function is differentiable, the error between the estimated directional derivative information and the real one is small, so this strategy can ensure the good convergence speed. However, it seems that this method exist the shortcoming of easy to fall into the local optimal, this could be solved by the crossover operator according to the local structure information and population behavior.

C. Crossover Ratio Facter Self-Adaption

For any intrinsic triangle, the centre point of it is identical to the ones of the triangle composed by the three base points, the area of these two triangle is proportioned. So we can use the area of triangle composed by the three base points to estimate the confident area. Compared with the search range, if the confident area is small, so the big crossover ratio is reasonable, that means this estimation has a big probability to accept; reversely, if the confident area is relatively bigger, that means this estimation contain the big uncertainty so the small crossover ration should be determined. In order to self-adaptive determine the crossover, a simple strategy is proposed, taking the ratio of the distance of the lower and upper bounds of the parameter vectors to the max distance between the three base points as the crossover ration. Our numerical pre-experiment shows the suitable Cr in the range [0.5,0.9] for most test function, so in order to avoid the searching processing falling to the local optimal, so we map linearly the above mentioned ratio into the interval [0.5,0.9] as the Cr value.

D. Algorithm Detail

Here we give the detail implement of our strategy to self adaptive defining the control parameters based on ITDE.

NP: Population size
FEs(fitcount): fitness evaluations used.
Max_FEs: Max fitness evaluations, stop criterion

Step 1: Initialization

Set the generation number G=0, and create randomly initial population of NP individuals.

Calculate $f(\mathbf{X})$, for each individual vector.

Step 2: Update the individual by our strategy

For each individual,

1) Mutation ($X_i(G) \rightarrow V_i(G)$)

Randomly select three member of differences randomly from the interval [1,NP] and ensure that the indices:

$$r1 \neq r2 \neq r3 \neq i$$

Select the X_{r1} as the base vector, and use the differential of X_{r2} and X_{r3} as differential vector. At the same time, calculate the difference vectors between the X_{r1} and X_{r2}, X_{r1} and X_{r3}, i.e.:

$$\begin{aligned} V_{base} &= V_{23} = X_{r2} - X_{r3} \\ V_{12} &= X_{r2} - X_{r1} \\ V_{13} &= X_{r3} - X_{r1} \end{aligned} \tag{17}$$

Calculate the fitness function difference of X_{r1} and X_{r2}, X_{r1} and X_{r3} separately, i.e.:

$$\begin{aligned} f_{12} &= f(X_{r1}) - f(X_{r2}) \\ f_{13} &= f(X_{r1}) - f(X_{r3}) \end{aligned} \tag{18}$$

Calculate the projection of V_{12} and V_{13} on the direction V_{23} separately, i.e.:

$$\begin{aligned} p_{12} &= V_{12} * V_{base}^{\ T} \\ p_{13} &= V_{13} * V_{base}^{\ T} \end{aligned} \tag{19}$$

Get the possible two target vectors:

$$\begin{aligned} V_{p1} &= p_{12} * unit(V_{base}) + X_{r1} \\ V_{p2} &= p_{13} * unit(V_{base}) + X_{r1} \end{aligned} \tag{20}$$

Where the unit means to get unit vector of the vector.

Select the target vector:

$$V_{i,g} = \begin{cases} V_{p1} & if\left(\dfrac{f12}{p12} < \dfrac{f13}{p13}\right) \\ V_{p2} & otherwise \end{cases} \tag{21}$$

2) Crossover ($V_i(G) \rightarrow U_i(G)$)

Calculate the distance between X_{r1} and X_{r2}, X_{r1} and X_{r3}, X_{r2} and X_{r3} separately, i.e. d_{r1r2}, d_{r1r3} and d_{r2r3}. Get the maximum distance of them, and then the Cr is determined as followed:

$$\begin{aligned} d_{\max} &= \max(d_{r1r2}, d_{r2r3}, d_{r1r3}) \\ Cr &= 1 - d_{\max}/(\mathrm{xu} - \mathrm{xl}) * 0.4 - 0.1 \end{aligned} \tag{22}$$

In order to avoid the search falling to the local extreme, so we map the ratio between $d_{\max}$ to the parameter's range into the interval [0.5, 0.9].

3) Selection ($U_i(G) \xrightarrow{?} X_i(G+1)$)

The selection is operated by using our mechanism in DCDE [8].

Step 3: Stop searching: if FEs >Max_FEs, otherwise repeat step 2 .

IV. EXPERIMENTS AND RESULTS

<u>*PC Configuration:*</u>

System: Windows XP; CPU: 3.40GHz; RAM: 2.00 GB;

Language: Matlab 7.6.0

Algorithm: ITDE, DCDE

Parameters Setting:

a) Parameters to be adjusted:

P, L, L_FEs

b) Corresponding dynamic ranges: self-adaptation

c) Guidelines on how to adjust the parameters

NP can be small for 10 dimensions problems and large for 30 dimensions problems. NP=3*D is a recommended choice.

d) Estimated cost of parameter tuning in terms of FEs.

e) Actual parameter values used.

P=100; *L*=500;

L_FEs=1,000;*Max_FEs*=600,000

Experiments are progressed over a suite of 18 single objective constrained real parameter optimization problems provided in [9]. For each problem, the ITDE is run 25 times. Tables II-VII reports the results. The predefined tolerance value for the 18 test functions is 1e-4.

TABLE I. FUNCTION VALUES ACHIEVED WHEN FES= 2×105 FOR 10D PROBLEMS 1-6 C01-C06

	C01		C02		C03		C04		C05		C06	
	DCDE	ITDE	DCDE	ITDE	DCDE	ITDE	DCDE	ITDE	DCDE	ITDE	DCDE	ITDE
Mean	-0.62	-0.607	-2.262	-2.265	4.29E-25	1.51E-22	0.01	1.02E-4	-483.61	-483.61	-578.654	-578.657
Std	0.102	0.093	0.015	0.015	6.53E-25	1.83E-22	0.015	5.59E-4	0.002	0.001	0.014	0.008

TABLE II. FUNCTION VALUES ACHIEVED WHEN FES= 2×105 FOR 10D PROBLEMS 7-12 C07-C12

	C07		C08		C09		C10		C11		C12	
	DCDE	ITDE	DCDE	ITDE	DCDE	ITDE	DCDE	ITDE	DCDE	ITDE	DCDE	ITDE
Mean	0.159	0.318	23.345	29.007	5.99E-01	0.494	6.008	13.47	-0.015	-0.032	-36.132	-127.522
Std	0.797	1.103	63.16	48.139	2.26E+00	2.47	14.798	44.207	0.032	0.042	177.341	278.24

TABLE III. FUNCTION VALUES ACHIEVED WHEN FES= 2×105 FOR 10D PROBLEMS 13-18 C13-C18

	C13		C14		C15		C16		C17		C18	
	DCDE	ITDE	DCDE	ITDE	DCDE	ITDE	DCDE	ITDE	DCDE	ITDE	DCDE	ITDE
Mean	-56.169	-59.693	0.159	15.016	2.57E-24	3795.176	0.05	0.164	0.075	0.305	0.139	1.04E-15
Std	6.533	3.489	0.797	72.602	1.29E-23	18968.39	0.139	0.157	0.214	0.496	0.698	3.61E-15

TABLE IV. FUNCTION VALUES ACHIEVED WHEN FES= 6×105 FOR 30D PROBLEMS 1-6 C01-C06

	C01		C02		C03		C04		C05		C06	
	DCDE	ITDE	DCDE	ITDE	DCDE	ITDE	DCDE	ITDE	DCDE	ITDE	DCDE	ITDE
Mean	-0.602	-0.498	-2.157	-2.249	0.052	6.976	0.099	0.054	-310.895	-472.306	-523.932	-530.342
Std	0.121	0.029	0.087	0.013	0.264	9.047	0.042	0.034	350.434	29.745	27.272	0.26

TABLE V. FUNCTION VALUES ACHIEVED WHEN FES= 6×105 FOR 30D PROBLEMS 7-12 C07-C12

	C07		C08		C09		C10		C11		C12	
	DCDE	ITDE	DCDE	ITDE	DCDE	ITDE	DCDE	ITDE	DCDE	ITDE	DCDE	ITDE
Mean	0.159	0.569	8.762	1.139	2.13E-24	7.877	0.485	0.578	0.004	0.007	0.476	-37.903
Std	0.797	1.506	30.233	1.945	8.60E-24	20.841	1.342	1.53	0.008	0.01	3.37	65.40

TABLE VI. FUNCTION VALUES ACHIEVED WHEN FES= 6×105 FOR 30D PROBLEMS 13-18 C13-C18

	C13		C14		C15		C16		C17		C18	
	DCDE	ITDE	DCDE	ITDE	DCDE	ITDE	DCDE	ITDE	DCDE	ITDE	DCDE	ITDE
Mean	-63.613	-60.298	0.637	1.139	0.505	1.806	0.006	0.024	1.95	0.51	0.139	4.45E-5
Std	3.122	2.887	1.491	1.945	1.397	2.253	0.028	0.046	7.996	0.659	0.617	9.891E-5

The results show that our approach is able to find a very good feasible solution among those 18 test functions. Compared with our previous test results based on the DCDE, the ITDE which has self-adaptation control parameters integrated with dynamic constraint search leads to that there are 30% percent of evaluation result has been improved, and some functions has almost same result while ITDE performs more robust. It means the self-adaptation control parameters strategy is effective for some function. For other functions without obvious improved and even worse, it should be considered that how to avoid premature and keep the diversity of population.

V. Conclusion

An improved Differential Evolution algorithm which adopts self-adaptive strategy to control the parameter F and CR (ITDE), integrated with the dynamic constraint-handling mechanism, is proposed to solve constrained real-parameter optimization problems in this paper. According to the constraint-handling mechanism, the individuals are adaptively assigned to explore different constraints in the search process. By analyzing the intrinsic triangle in Differential Evolution a better performance is generated by the self adaptive determine the control parameters according to the local information by intrinsic triangle projection estimation. The algorithm proposed here indicates some advantage in fast convergence, while should be improved by adjusting the selection strategy. Any algorithm could not be good at all of the performance, but need to find a good balance between the explorative and exploitative capability. In the future, we will compare the result with other control parameter self adaptive algorithm, and find the reason why ITDE don't have good performance for some test constrained problems, in order to find a new approach to solve constrained problems without pre-setting parameters or less.

References

[1] Storn R, Price K "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces,".*Journal of Global Optimization*, 1997, 11, pp. 341-359..

[2] Storn R , Price K.Differential evolution for multi -objective optimization [J].Evolutionary Computation, 2003, 4 : 8-12

[3] Efre′n Mezura-Montes,Jesu′s Vela′zquez-Reyes and Carlos A.Coello Coello "Modified Differential Evolution for Constrained Optimization" , IEEE Congress on Evolutionary Computation (CEC 2006) ,Canada ,July 16-21,2006

[4] Storn R "System Design by Constraint Adaptation and Differential Evolution" IEEE Transactions on Evolutionary Computation, 3(1): pp:22-34, April 1999

[5] J. Ilonen, J.-K. Kamarainen, and J. Lampinen, "Differential evolution training algorithm for feed-forward neural networks," Neural Process. Lett., vol. 7, no. 1, pp. 93–105, 2003.

[6] S. Das and P. N. Suganthan, "Differential Evolution: A Survey of the State-of-the-Art," IEEE. Trans. Evolutionary Computation, vol. 15, No.1, pp. 4–31, 2011.

[7] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," IEEE Trans. Evol. Comput., vol. 13, no. 2, pp. 398–417,Apr. 2009.

[8] Li Zhihui, J. J. Liang, He Xi and Shang Zhigang, "Differential Evolution with Dynamic Constraint-Handling Mechanism" Proc. of IEEE Congress on Evolutionary Computation (CEC 2010), July. 2010

[9] R. Mallipeddi, P. N. Suganthan, "Problem Definitions and Evaluation Criteria for the CEC 2010 Competition on Constrained Real- Parameter Optimization", Technical Report, Nanyang Technological University, Singapore, 2009

[10] J.J.Liang , and P.N. Suganthan,"Dynamic Multi-Swarm Particle Swarm Optimizer with a Novel Constraint-Handling Mechanism" Proc. of IEEE Congress on Evolutionary Computation (CEC 2006) ,pp.9-16, July 16-21,2006