Dynamic Differential Evolution for Constrained Real-Parameter Optimization

Youyun Ao[1], Hongqin Chi[2]
[1]School of Computer and Information, Anqing Teachers College, Anqing, China
Email: youyun.ao@gmail.com
[2]Department of Computer, Shanghai Normal University, Shanghai, China
Email: chihq@shnu.edu.cn

***Abstract*—Differential evolution (DE) has been shown to be a simple and effective evolutionary algorithm for global optimization both in benchmark test functions and many real-world applications. This paper introduces a dynamic differential evolution (D-DE) algorithm to solve constrained optimization problems. In D-DE, a novel mutation operator is firstly designed to prevent premature. Secondly, the scale factor *F* and the crossover probability *CR* are dynamic and adaptive to be beneficial for adjusting control parameters during the evolutionary process, especially, when done without any user interaction. Thirdly, D-DE uses orthogonal design method to generate initial population and reinitialize some solutions to replace some worse solutions during the search process. Finally, D-DE is validated on 6 benchmark test functions provided by the CEC 2006 special session on constrained real-parameter optimization. The experimental results obtained by D-DE are explained and discussed, and some conclusions are also drawn.**

***Index Terms*—constrained optimization, mutation scheme, differential evolution, evolutionary algorithm, constraint handling**

I. INTRODUCTION

Many real-world optimization problems in science and engineering involve a number of constraints which the optimal solution must satisfy. These problems are also called constrained optimization problems or nonlinear programming problems. We are most interested in the general constrained optimization problems, which are all transformed into the following format [1], [2], [3], [4]:

$$\begin{aligned} &\text{Minimize } f(\vec{x}),\ \vec{x}=[x_1,x_2,...,x_n]\in\Re^n \\ &\text{Subject to } g_j(\vec{x})\le 0,\ j=1,2,...,q \\ &\qquad\qquad h_j(\vec{x})=0,\ j=q+1,q+2,...,m \\ &\text{Where } L_i\le x_i\le U_i, i=1,2,...,D \end{aligned} \tag{1}$$

Here n is the number of the decision or parameter variables (that is, $\vec{x}$ is a vector of size D), the ith variable x_i varies in the range $[L_i,U_i]$. The function $f(\vec{x})$ is the objective function, $g_j(\vec{x})$ is the jth inequality constraint and $h_j(\vec{x})$ is the jth equality constraint. The decision or search space S is written as $S=\prod_{i=1}^{D}[L_i,U_i]$, and the feasible space F, expressed as $F=\{\vec{x}\in S \mid g_j(\vec{x})\le 0, j=1,2,...,q; h_j(\vec{x})=0, j=q+1,q+2,...,m\}$, is one subset of the parameter space S (obviously, $F\subseteq S$) which satisfies the equality and inequality constraints.

Population-based evolutionary algorithm, mainly due to its ease to implement and use, and its less susceptibleness to the characteristics of the function to be optimized, has become a very popular option to solve constrained optimization problems in benchmark test functions and real-world applications [5]. Muñoz Zavala et al. [6] proposed a new constrained optimization algorithm based on improved particle swarm optimization (COPSO). In order to keep diversity, COPSO introduces a hybrid approach based on a modified ring neighborhood structure with two new perturbation operators for perturbing the particle swarm optimization (PSO) memory. In addition, COPSO adopts a new and special handling technique for equality constraints where a dynamic tolerance value is adjusted to allow the survival of unfeasible particles. Furthermore, COPSO is applied to the solution of state-of-the-art benchmark test functions and various engineering design problems. Liang and Suganthan [7] proposed a dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism (DMS-PSO). DMS-PSO adopts a novel constraint-handling mechanism based on multi-swarm. Different from the existing constraints handling methods, sub-swarms are adaptively assigned to explore different constraints during the search process. Additionally, DMS-PSO introduces Sequential Quadratic Programming (SQP) method to improve local search ability. Finally, DMS-PSO is applied to the solution of constrained real-parameter optimization. Mezura-Montes et al. [8] proposed a modified differential evolution for constrained optimization (MDE). In order to increase the probability of each parent to generate a better offspring, MDE allows each solution to generate more than one offspring but using a different mutation operator which combines information of the current parent to find new search directions. Besides, MDE employs three selection criteria based on feasibility to deal with the constraints and adopts a diversity mechanism to maintain infeasible solutions located in promising areas of the search space. Takahama and Sakai [9] proposed a novel constrained optimization algorithm by the ε constrained differential evolution with gradient-based mutation and feasible elites (εDE). Firstly, εDE applies the ε constrained method to differential evolution. Secondly, to solve problems with

Manuscript received August 24, 2009; revised October 21, 2009.

doi:10.4304/jait.1.1.43-51

many equality constraints faster, which are very difficult problems for numerical optimization, εDE proposes gradient-based mutation and feasible elites preserving strategy. Finally, εDE is tested on 24 benchmark test functions provided by the CEC 2006 special session on constrained real-parameter optimization. Differential evolution (DE) [10], [11], a relatively new evolutionary technique, has been shown to be simple and powerful and has been widely applied to both benchmark test functions and real-world applications [12]. After analyzing the existing evolutionary algorithms, this paper introduces a new dynamic differential evolution (D-DE) algorithm for constrained real-parameter optimization efficiently.

The remainder of this paper is organized as follows. Section II briefly introduces the basic idea of DE. Section III describes in detail the D-DE algorithm. Section IV presents 6 benchmark test functions. Section V presents experimental settings adopted by D-DE, conventional DE and GA, respectively. Section VI provides an analysis of the results obtained from our empirical study. Finally, some conclusions and some possible paths for future research are provided in Section VII.

II. THE BASIC IDEA OF CONVENTIONAL DE

Let us assume that $\vec{x}_i^t = [x_{i,1}^t, x_{i,2}^t, ..., x_{i,D}^t]$ are solutions at generation *t*, $P^t = \{\vec{x}_1^t, \vec{x}_2^t, ..., \vec{x}_N^t\}$ are population, where *D* denotes the dimension of solution space, *N* is population size. In conventional DE, the child population P^{t+1} is generated through following operators [10], [13]:

A. Mutation Operator

For each $\vec{x}_i^t$ in parent population, the mutant vector $\vec{v}_i^{t+1}$ is generated according to the following equation:

$$\vec{v}_i^{t+1} = \vec{x}_{r_1}^t + F \times (\vec{x}_{r_2}^t - \vec{x}_{r_3}^t) \tag{2}$$

Where $r_1, r_2, r_3 \in \{1,2,...,N\} \setminus i$ are randomly chosen and mutually different, the scaling factor *F* is used to control amplification of the differential variation $\vec{x}_{r_2}^t - \vec{x}_{r_3}^t$.

B. Crossover Operator

For each individual $\vec{x}_i^t$, a trial vector $\vec{u}_i^{t+1}$ is generated by the following equation:

$$u_{i,j}^{t+1} = \begin{cases} v_{i,j}^{t+1}, & \text{if}(rand \le CR \,\|\, j = rand[1,D]) \\ x_{i,j}^t, & \text{otherwise} \end{cases} \tag{3}$$

Where *rand* is a uniform random number distributed between 0 and 1, $rand[1,D]$ is a randomly selected index from the set $\{1,2,...,D\}$, the crossover probability $CR \in [0,1]$ is used to control the diversity of individuals.

C. Selection Operator

The child individual $\vec{x}_i^{t+1}$ is selected from each pair of $\vec{x}_i^t$ and $\vec{u}_i^{t+1}$ by using greedy selection criterion:

$$\vec{x}_i^{t+1} = \begin{cases} \vec{u}_i^{t+1}, & \text{if}(f(\vec{u}_i^{t+1}) < f(\vec{x}_i^t)) \\ \vec{x}_i^t, & \text{otherwise} \end{cases} \tag{4}$$

Where the function f is the objective function and the condition $f(\vec{u}_i^{t+1}) < f(\vec{x}_i^t)$ means the individual $\vec{u}_i^{t+1}$ is better than $\vec{x}_i^t$.

1: Generate initial population P^0.
2: Evaluate P^0 and let generation counter $t = 0$.
3: **While** *(the stopping criterion is not satisfied)* **do** {
4: For each individual $\vec{x}_i^t$, its offspring $\vec{x}_i^{t+1}$ is generated
5: by mutation, crossover and selection operators.
6: Evaluate P^{t+1} and let $t = t+1$ }

Figure 1. The general framework of DE.

III. THE PROPOSED ALGORITHM DDE

A. Orthogonal Initial Population

Generally, the initial population $P^0 = \{\vec{x}_1^0, \vec{x}_2^0, ..., \vec{x}_N^0\}$ of evolutionary algorithm is randomly generated as follows:

$$\forall i \le N, \forall j \le D : x_{i,j}^0 = L_j + \mathrm{r}_j \times (U_j - L_j) \tag{5}$$

Where *N* is the population size, *D* is the number of variables, r_j is a random number between 0 and 1, the *j*th variable of $\vec{x}_i^0$, written as $x_{i,j}^0$, is initialized in the range $[L_j, U_j]$. In order to improve the search efficiency, this paper employs orthogonal design method to generate the initial population, which can make some points closer to the global optimal point and improve the diversity of solutions. The orthogonal design method is described as follows [14]:

For any given individual $\vec{x} = [x_1, x_2, ..., x_D]$, the *i*th decision variable x_i varies in the range $[L_i, U_i]$. Here, each x_i is taken as each factor of orthogonal design. Let us assume that each factor holds *Q* levels, namely, quantize the domain $[L_i, U_i]$ into *Q* levels $\alpha_1, \alpha_2, ..., \alpha_Q$. The *j*th level of the *i*th factor $\alpha_{i,j}$ is defined as follows:

$$a_{i,j} = \begin{cases} L_i & , j = 1 \\ L_i + (j-1)(\frac{U_i - L_i}{Q-1}) & , 2 \le j \le Q-1 \\ U_i & , j = Q \end{cases} \tag{6}$$

Thereafter, we create the orthogonal array $M = (b_{i,j})_{N \times D}$ with *D* factors and *Q* levels, where *N* is the number of level combinations. The procedure of generating the orthogonal array $M = (b_{i,j})_{N \times D}$ is described as follows:

1: **for** ($i = 1; i \le N; i++$)
2: { $b_{i,1} =$ **int**($(i-1)/Q$) **mod** Q ; $b_{i,2} = (i-1)$ **mod** Q }
3: **for** ($j = 3; j \le D; j++$)
4: **for** ($i = 1; i \le N; i++$)
5: { $b_{i,j} = (b_{i,1} \times (j-2) + b_{i,2})$ **mod** Q }
6: Increment $b_{i,j}$ by one for $1 \le i \le N, 1 \le j \le D$

Figure 2. Generating orthogonal array $M = (b_j^i)_{N \times D}$.

Therefore, the initial population $P^0 = (x_{i,j}^0)_{N\times D}$ can be generated by using the orthogonal array $M = (b_{i,j})_{N\times D}$, where the jth variable of individual $\vec{x}_i^0$ is $x_{i,j}^0 = a_{j,b_{i,j}}$.

B. Novel Mutation Scheme

According to the different variants of mutation, there are several different DE schemes often used, which are formulated as follows [10]:

"DE/rand/1/bin": $\vec{v}_i^{t+1} = \vec{x}_{r_1}^t + F \times (\vec{x}_{r_2}^t - \vec{x}_{r_3}^t)$ (7)

"DE/best/1/bin": $\vec{v}_i^{t+1} = \vec{x}_{best} + F \times (\vec{x}_{r_1}^t - \vec{x}_{r_2}^t)$ (8)

"DE/current to best/2/bin":

$$\vec{v}_i^{t+1} = \vec{x}_i^t + F \times (\vec{x}_{best} - \vec{x}_i^t) + F \times (\vec{x}_{r_1}^t - \vec{x}_{r_2}^t) \quad (9)$$

"DE/best/2/bin":

$$\vec{v}_i^{t+1} = \vec{x}_{best} + F \times (\vec{x}_{r_1}^t - \vec{x}_{r_2}^t) + F \times (\vec{x}_{r_3}^t - \vec{x}_{r_4}^t) \quad (10)$$

"DE/rand/2/bin":

$$\vec{v}_i^{t+1} = \vec{x}_{r_1}^t + F \times (\vec{x}_{r_2}^t - \vec{x}_{r_3}^t) + F \times (\vec{x}_{r_4}^t - \vec{x}_{r_5}^t) \quad (11)$$

Where $\vec{x}_{best}$ is the best solution of the current population, and the control parameter F is usually set to be a constant. Using $\vec{x}_{best}$ can improve the convergence speed but also increase the probability of getting stuck in the local optimum. In order to overcome the limitations, this paper proposes a novel variant of mutation, which is defined as follows:

$$\vec{v}_i^{t+1} = \vec{x}_{better} + F \times \sum_{k=1}^{K/2} (\vec{x}_{r_{2k-1}}^t - \vec{x}_{2k}^t) \quad (12)$$

Where $r_1, r_2, ..., r_K \in \{1,2,...,N\} \setminus i$, they are K mutually different and randomly chosen integers. The better solution $\vec{x}_{better}$ is a random sample from top N_a solutions after ranking the current population based on the feasibility rule described in the later. The scale factor F is a dynamic control parameter and related to the generation number, which is defined as follows:

$$F = F_{\min} + (F_{\max} - F_{\min}) \times (1 - \frac{t}{T})^{F_b} \quad (13)$$

Here $F_{\min}$ and $F_{\max}$ are the bottom and upper boundaries of F, and usually are set to 0.1 and 0.9 respectively. The exponent F_b is a shape parameter determining the degree of dependency on the generation number and usually is set to 2 or 3. Two parameters t and T are the current generation number and the maximal generation number respectively.

C. Dynamic Control Parameters

In conventional DE, the crossover probability CR is a constant value between 0 and 1. In this study, a dynamic crossover probability CR is defined as follows:

$$CR = CR_{\min} + (CR_{\max} - CR_{\min}) \times (1 - \frac{t}{T})^{CR_b} \quad (14)$$

Here $CR_{\min}$ and $CR_{\max}$ are the bottom and upper boundaries of CR, and usually are set to 0.1 and 0.9 respectively. The exponent CR_b is a shape parameter determining the degree of dependency on the generation number and usually is set to 2 or 3. Two parameters t and T are the current generation number and the maximal generation number respectively.

At the early stage, D-DE uses a bigger scale factor F and a bigger crossover probability CR to search the solution space to preserve the diversity of solutions and prevent premature; at the later stage, D-DE employs a smaller scale factor F and a smaller crossover probability CR to search the solution space to enhance the local search and prevent the better solutions found from being destroyed.

D. Repair Rule

After crossover, if one or more of the variables in the new vector $\vec{u}_i^{t+1}$ are outside their boundaries, the violated variable value $\vec{u}_{i,j}^{t+1}$ is either reflected back from the violated boundary or set to the corresponding boundary value using the repair rule as follows [15], [16]:

$$u_{i,j}^{t+1} = \begin{cases} (L_j + u_{i,j}^{t+1})/2, & \text{if } (p \le 1/3) \wedge (u_{i,j}^{t+1} < L_j) \\ L_j, & \text{if } (1/3 < p \le 2/3) \wedge (u_{i,j}^{t+1} < L_j) \\ 2L_j - u_{i,j}^{t+1}, & \text{if } (p > 2/3) \wedge (u_{i,j}^{t+1} < L_j) \\ (U_j + u_{i,j}^{t+1})/2, & \text{if } (p \le 1/3) \wedge (u_{i,j}^{t+1} > U_j) \\ U_j, & \text{if } (1/3 < p \le 2/3) \wedge (u_{i,j}^{t+1} > U_j) \\ 2U_j - u_{i,j}^{t+1}, & \text{if } (p > 2/3) \wedge (u_{i,j}^{t+1} > U_j) \end{cases} \quad (15)$$

Where p is a probability and uniformly distributed random number in the range $[0,1]$.

E. Constraint Handling Mechanism

In evolutionary algorithms for solving constrained optimization problems, the most common method to handle constraints is to use penalty functions. Usually equality constraints are transformed into inequalities of the form [4]:

$$|h_j(\vec{x})| - \varepsilon \le 0, j = q+1, q+2, ..., m \quad (16)$$

Here ε is a tolerance allowed (a very small value) for the equality constraints.

In general, the constraint violation function of one individual $\vec{x}$ is transformed by m equality and inequality constraints as follows [9], [17], [18]:

$$G(\vec{x}) = \sum_{j=1}^{q} w_j \max(0, g_j(\vec{x}))^{\beta} + \sum_{j=q+1}^{m} w_j \max(0, |h_j(\vec{x})| - \varepsilon)^{\beta} \quad (17)$$

Here the exponent β is a positive number and usually set to 1 or 2, and the coefficient w_j is greater than zero. The function value $G(\vec{x})$ shows that the degree of constraints violation of individual $\vec{x}$. β is set to 2 and w_j is set to 1 in this study.

In this study, a simple and efficient constraint handling technique of feasibility-based rule is introduced, which is also a constraint handling technique without parameters. When two solutions are compared at a time, the following criteria are always applied [3]:

1) If one solution is feasible, and the other is infeasible, the feasible solution is preferred;

2) If both solutions are feasible, the one with the better objective function value is preferred;

3) If both solutions are infeasible, the one with smaller constraint violation function value is preferred.

F. Algorithm Framework

The general framework of the D-DE algorithm is outlined as follows:

1: Generate orthogonal initial population $P^0 = \{\vec{x}_1^0, \vec{x}_2^0, ..., \vec{x}_N^0\}$,
2: initialize parameters, and let $t = 0$.
3: Evaluate P^0, and rank P^0 based on feasibility rule.
4: **repeat**
5: **for** each individual $\vec{x}_i^t$ in the population P^t **do**
6: Generate K random integers $r_1, r_2, ..., r_K \in \{1,2,..., N\} \setminus i$,
7: they are also mutually different.
8: Generate a random integer $j_{rand} \in \{1,2,..., D\}$.
9: Randomly select a sample $\vec{x}_{better}$ from top N_a individuals of
10: the current population P^t.
11: **for** each parameter $j \in \{1,2,..., K/2\}$ **do**
12: $$\vec{u}_{i,j}^{t+1} = \begin{cases} \vec{x}_{better,j} + F \times (\vec{x}_{r_{2k-1},j}^t - \vec{x}_{r_{2k},j}^t), \\ \quad \text{if } (rand \le CR \,\|\, j = rand[1, D]) \\ \vec{x}_{i,j}^t, \quad \text{otherwise} \end{cases}$$
13: **end for**
14: Apply repair rule to repair $\vec{u}_{i,j}^{t+1}$ if required, and evaluate $\vec{u}_{i,j}^{t+1}$.
15: Replace $\vec{x}_i^t$ with the child $\vec{u}_i^{t+1}$ in the population P^{t+1}, if $\vec{u}_i^{t+1}$
16 is better, otherwise $\vec{x}_i^t$ is retained.
17: **end for**
18: Rank P^{t+1} based on the feasibility rule, then replace N_b worse
19: solutions with N_b orthogonal reinitialized solutions.
20: Rank P^{t+1} based on the feasibility rule, and let $t = t + 1$.
21:**until** (*the termination condition is achieved*)

Figure 3. The general framework of the D-DE algorithm.

IV. TEST FUNCTION SUITE

In order to validate D-DE, we employ 6 benchmark test problems $g04$, $g06$, $g08$, $g11$, $g12$, which are provided by the CEC 2006 special session on constrained real-parameter optimization [4], and which are described in the following.

A. Test function g02

$$\text{Minimize } f(\vec{x}) = -\left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right|$$

$$\text{Subject to } g_1(\vec{x}) = 0.75 - \prod_{i=1}^{n} x_i \le 0$$

$$g_2(\vec{x}) = \sum_{i=1}^{n} x_i - 7.5n \le 0$$

Where $n = 20$ and $0 < x_i \le 10\,(i = 1,2,..., n)$. The global minimum $\vec{x}^* =$ (3.16246061572185,3.12833142812967, 3.09479212988791,3.06145059523469,3.02792915885555,2.99382606701730,2.95866871765285,2.92184227312450,0.49482511456933,0.48835711005490,0.48231642711865,0.47664475092742,0.47129550835493,0.46623099264167,0.46142004984199,0.45683664767217,0.45245876903267,0.44826762241853,0.44424700958760,0.44038285956317), the best is $f(\vec{x}^*) = -0.80361910412559$, constraint g_1 is close to being active.

B. Test function g04

$$\text{Minimize } f(\vec{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$$

$$\text{Subject to } g_1(\vec{x}) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \le 0$$

$$g_2(\vec{x}) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \le 0$$

$$g_3(\vec{x}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \le 0$$

$$g_4(\vec{x}) = -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \le 0$$

$$g_5(\vec{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \le 0$$

$$g_6(\vec{x}) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \le 0$$

Where $78 \le x_1 \le 102$, $33 \le x_2 \le 45$ and $27 \le x_i \le 45$ $(i = 3,4,5)$. The optimum solution is $\vec{x}^* =$ (78,33, 29.9952560256815985,45,36.7758129057882073),where $f(\vec{x}^*) = -3.066553867178332e+004$. Two constraints are active (g_1 and g_6).

C. Test function g06

$$\text{Minimize } f(\vec{x}) = (x_1 - 10)^3 + (x_2 - 20)^3$$

$$\text{Subject to } g_1(\vec{x}) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \le 0$$

$$g_2(\vec{x}) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \le 0$$

Where $13 \le x_1 \le 100$ and $0 \le x_2 \le 100$. The optimum solution is $\vec{x}^* =$ (14.09500000000000064,0.84296078921547956 68) where $f(\vec{x}^*) =$ -6961.81387558015. Both constraints are active.

D. Test function g08

$$\text{Minimize } f(\vec{x}) = -\frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$$

$$\text{Subject to } g_1(\vec{x}) = x_1^2 - x_2 + 1 \le 0$$

$$g_2(\vec{x}) = 1 - x_1 + (x_2 - 4)^2 \le 0$$

Where $0 \le x_1 \le 10$ and $0 \le x_2 \le 10$. The optimum solution is $\vec{x}^* =$ (1.22797135260752599,4.24537336612274885) where $f(\vec{x}^*) =$ -0.0958250 414180359.

E. Test function g11

$$\text{Minimize } f(\vec{x}) = x_1^2 + (x_2 - 1)^2$$

$$\text{Subject to } h(\vec{x}) = x_2 - x_1^2 = 0$$

Where $-1 \le x_1 \le 1$, $-1 \le x_2 \le 1$. The optimum solution is $\vec{x}^* =$ (-0.707036070037170616,0.500000004333606807)

where $f(\vec{x}^*) = 0.7499$.

F. Test function g12

Minimize

$$f(\vec{x}) = -(100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2)/100$$

Subject to $g(\vec{x}) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \le 0$

Where $0 \le x_i \le 10\,(i = 1,2,3)$ and $p, q, r = 1,2,...,9$. The feasible region of the search space consists of 9^3 disjointed spheres. A point (x_1, x_2, x_3) is feasible if and only if there exists p, q, r such that the above inequality holds. The optimum solution is $\vec{x}^* = (5,5,5)$ where $f(\vec{x}^*) = -1$. The solution lies within the feasible region.

V. Experimental Settings

A. Parameter Settings of D-DE

In our experimental study, the parameter values used in D-DE are set as follows: the population size $N = 50$, the maximal generation number $T = 5000$, the level number $Q = \lfloor \sqrt{N} \rfloor$, the number of top solutions $N_a = 0.1 \times N = 5$, the number of replaced solutions $N_b = 0.1 \times N = 5$, the minimal and maximal values of scale factor F are set to $F_{\min} = 0.1$ and $F_{\max} = 0.9$ respectively, $K = 6$, the minimal and maximal values of crossover probability CR are set to $CR_{\min} = 0.1$ and $CR_{\max} = 0.9$ respectively, the shape parameter values $F_a = 3$ and $F_b = 3$ respectively, the exponent $\beta = 2$, the tolerant value $\varepsilon = 0.0001$. The number of function evaluations (FES) is equal to $(1 + N_b) \times N \times T$ =275,000. The achieved solution at the end of $(1 + N_a) \times N \times T$ FES is used to measure the performance of D-DE. D-DE is independently run 30 times on each test function.

B. Parameter Settings of Conventional DE

In our experimental study, the parameter values adopted by the conventional DE are set as follows: the population size $N = 50$, the maximal generation number $T = 55000$, the crossover probability $CR = 0.9$, the scale factor $F = 0.6$, the tolerant value $\varepsilon = 0.0001$. The number of function evaluations (FES) is equal to $N \times T$ =275,000. The achieved solution at the end of $N \times T$ FES is used to measure the performance of the conventional DE. The DE employs the repair rule and constraint handling mechanism described in Section III and is independently run 30 times on each test function.

C. Parameter Settings of Conventional GA

As a computing technique and method, population-based genetic algorithm (GA) [20] has been shown to be an effective evolutionary algorithm [21]. In our experimental study, the conventional GA uses simulated binary crossover (SBX) [22], polynomial mutation operator [23], tournament selection between the parent and its child, the repair rule and constraint handling mechanism described in Section III. The parameter values employed by the real-coded GA are set as follows: the population size $N = 50$, the maximal generation number $T = 55000$, the crossover probability $P_c = 0.9$ and a mutation probability $P_m = 1/n$ (where n is the number of decision variables for real-coded GA), the distribution indexes for crossover and mutation operators as $\eta_c = 20$ and $\eta_m = 20$; the tolerant value $\varepsilon = 0.0001$. The number of function evaluations (FES) is equal to $N \times T = 275{,}000$. The obtained solution at the end of $N \times T$ FES is used to measure the performance of the conventional GA. The GA is independently run 30 times on each test function.

VI. Experimental Results and Discussions

A. Comparison with Respect to Conventional DE and GA on 6 Benchmark Test Problems

Using the above experimental settings, the best, mean and worst results obtained by D-DE, DE, and GA are given in Tables I-III. As shown in Table I, D-DE, DE and GA all can find the best solution for each test problems $g08$, $g11$ and $g12$, respectively. For test problems $g04$ and $g06$, D-DE and DE can find the best solution and is better than that of GA. For test problem $g02$, the best result obtained by DE is slightly better than that of D-DE, and obviously better than that of GA. According to the mean results given in Table II, the mean result obtained by D-DE is better than or not worse than that of DE and GA for all test problems $g02$, $g04$, $g06$, $g08$, $g11$ and $g12$, while the mean result obtained by DE is better than or not worse than that of GA for test problems $g02$, $g04$, $g06$, $g08$ and $g12$ except for test problem $g11$ where the mean result obtained by GA is better than that of DE. Table II shows that the worst result obtained by D-DE is better than or not worse than that of DE and GA for all test problems $g02$, $g04$, $g06$, $g08$, $g11$ and $g12$, while the worst result obtained by DE is better than or not worse than that of GA for test problems $g02$, $g04$, $g06$, $g08$ and $g12$, except for test problem $g11$ where the worst result obtained by GA is better than that of DE. Additionally, according to Tables I-III, we can find that D-DE can almost find the optimum solution for each test problems $g02$, $g04$, $g06$, $g08$, $g11$ and $g12$ in one single run, and that D-DE is robust and can outperform DE and GA on a set of test problems.

TABLE I.
THE BEST RESULTS OBTAINED BY D-DE WITH RESPECT TO THOSE OBTAINED BY DE, GA ON 6 BENCHMARK PROBLEMS.

Function	Optimal	D-DE	DE	GA
$g02$	-0.803619	-0.80356676	-0.80361902	-0.80254846
$g04$	-30665.539	-30665.53867	-30665.53867	-30664.86146
$g06$	-6961.814	-6961.81388	-6961.81388	-6958.24296
$g08$	-0.095825	-0.09582504	-0.09582504	-0.09582504
$g11$	0.7499	0.749900000	0.749900000	0.749900232
$g12$	-1	-1	-1	-1.00000000

TABLE II.
THE MEAN RESULTS OBTAINED BY D-DE WITH RESPECT TO THOSE OBTAINED BY DE, GA ON 6 BENCHMARK PROBLEMS.

Function	Optimal	D-DE	DE	GA
$g02$	-0.803619	-0.80150890	-0.79516035	-0.78941723
$g04$	-30665.539	-30665.53867	-30665.53867	-30661.64864
$g06$	-6961.814	-6961.81388	-6961.81388	-6925.57354
$g08$	-0.095825	-0.09582504	-0.09582504	-0.09582504
$g11$	0.7499	0.749900000	0.897159582	0.750805569
$g12$	-1	-1	-1	-1.00000000

TABLE III.
THE WORST RESULTS OBTAINED BY D-DE WITH RESPECT TO THOSE OBTAINED BY DE, GA ON 6 BENCHMARK PROBLEMS.

Function	Optimal	D-DE	DE	GA
$g02$	-0.803619	-0.79253455	-0.77041284	-0.75131833
$g04$	-30665.539	-30665.53867	-30665.53867	-30655.21039
$g06$	-6961.814	-6961.81388	-6961.81388	-6887.72373
$g08$	-0.095825	-0.09582504	-0.09582504	-0.09582504
$g11$	0.7499	0.749900000	1.000000000	0.755040572
$g12$	-1	-1	-1	-1.00000000

B. Comparison with Respect to Some State-of-the-art Approaches on 6 Benchmark Test Problems

In this section, we present the experimental results in detail and compare D-DE with respect to state-of-the-art algorithms. The experimental results are given in Table IV. The optimized objective function values (of 30 runs) arranged in ascending order and the 15th value in the list is called the median optimized function value.

According to the summary of statistical results of test problems $g04$, $g06$, $g08$, $g11$, $g12$ given in Table IV, it is clearly seen that D-DE, A-DDE [19], COPSO [6] and SRES [18] all can find the optimum or near-optimum, when D-DE uses 275,000 FES, A-DDE 180,000 FES, COPSO 350,000 FES and SRES 500,000 FES, respectively. For test problem $g02$, the mean, worst and standard derivation of values obtained by D-DE are the best when compared with other algorithms, while the median value obtained by D-DE is better than that of A-DDE and SRES, and is slightly worse than that of COPSO. Besides, for test problem $g02$, the best value obtained by D-DE is slightly worse than that of the other algorithms. As shown in Table V, the best, median, mean, worst and standard derivation of values obtained by D-DE when set to 550,000 FES are obviously better than those when set to 275,000 FES. The best, median values obtained by D-DE when set to 550,000 FES are almost convergent to the optimum or near-optimum. Therefore, for test problem $g02$, D-DE is not still convergent to the optimum when set to 275,000 FES. In conclusion, the performance of D-DE is stable and better than or not worse than some state-of-the-art evolutionary algorithms on a set of test problems.

TABLE IV.
COMPARISON D-DE WITH RESPECT TO ALGORITHMS A-DDE [19], COPSO [6], SRES [18] ON 6 BENCHMARK TEST FUNCTIONS

Function	Optimal	Method	Best	Median	Mean	Worst	Std	FES
$g02$	-0.803619	D-DE	-0.80356676178	-0.803457738386	-0.801508902296	-0.79253454688	4.00E-03	275,000
		A-DDE	-0.803605	-0.777368	-0.771090	-0.609853	3.66E-02	180,000
		COPSO	-0.803619	-0.803617	-0.801320	-0.786566	4.59E-03	350,000
		SRES	-0.804	-0.793	-0.788	-0.746	1.3E-02	500,000
$g04$	-30665.539	D-DE	-30665.53867178	-30665.53867178	-30665.53867178	-30665.53867178	1.16E-011	275,000
		A-DDE	-30665.539	-30665.539	-30665.539	-30665.539	3.20E-13	180,000
		COPSO	-30665.538672	-30665.538672	-30665.538672	-30665.538672	0	350,000
		SRES	-30665.539	-30665.539	-30665.539	-30665.539	0.0E+00	500,000
$g06$	-6961.814	D-DE	-6961.81387558	-6961.81387558	-6961.81387558	-6961.81387558	4.63E-012	275,000
		A-DDE	-6961.814	-6961.814	-6961.814	-6961.814	2.11E-12	180,000
		COPSO	-6961.813876	-6961.813876	-6961.813876	-6961.813876	0	350,000
		SRES	-6961.814	-6961.814	-6961.814	-6961.814	1.9E-12	500,000
$g08$	-0.095825	D-DE	-0.095825041418	-0.095825041418	-0.095825041418	-0.095825041418	2.82E-017	275,000
		A-DDE	-0.095825	-0.095825	-0.095825	-0.095825	9.10E-10	180,000
		COPSO	-0.095825	-0.095825	-0.095825	-0.095825	0	350,000
		SRES	-0.096	-0.096	-0.096	-0.096	0.0E+00	500,000
$g11$	0.7499	D-DE	0.749900000000	0.749900000000	0.749900000000	0.749900000000	1.13E-016	275,000
		A-DDE	0.75	0.75	0.75	0.75	5.35E-15	180,000
		COPSO	0.749999	0.749999	0.749999	0.749999	0	350,000
		SRES	0.750	0.750	0.750	0.750	1.1E-16	500,000
$g12$	-1	D-DE	-1	-1	-1	-1	0	275,000
		A-DDE	-1.000	-1.000	-1.000	-1.000	4.10E-11	180,000
		COPSO	-1.000000	-1.000000	-1.000000	-1.000000	0	350,000
		SRES	-1.000	-1.000	-1.000	-1.000	0.0E+00	500,000

TABLE V.
EXPERIMENTAL RESULTS OBTAINED BY D-DE WHEN FES=275,000 , FES=550,000 FOR TEST FUNCTION $g02$ OVER 30 RUNS

FES	Best	Median	Mean	Worst	Std
275,000	-0.80356676178	-0.803457738386	-0.801508902296	-0.79253454688	4.00E-03
550,000	-0.803610090279	-0.8036058890	-0.802935868229	-0.79259834414	2.55E-03

C. Convergence Graphs Obtained by D-DE for 6 Benchmark Test Problems

In order to provide a more intuitive comprehension, we present the convergence graphs obtained by D-DE for test problems $g02$, $g04$, $g06$, $g08$, $g11$ and $g12$. Figures 4-9 depict the convergence graphs for test problems $g02$, $g04$, $g06$, $g08$, $g11$ and $g12$, respectively. It is clearly seen that D-DE has a trend to find the optimum solution for test problem $g02$ within 300,000 FES, that D-DE can find the optimum solution for each test problem $g06$, $g08$, $g11$, $g12$ within 50,000 FES, and that D-DE can obtain the optimum solution for test problem $g04$ within 100,000 FES.

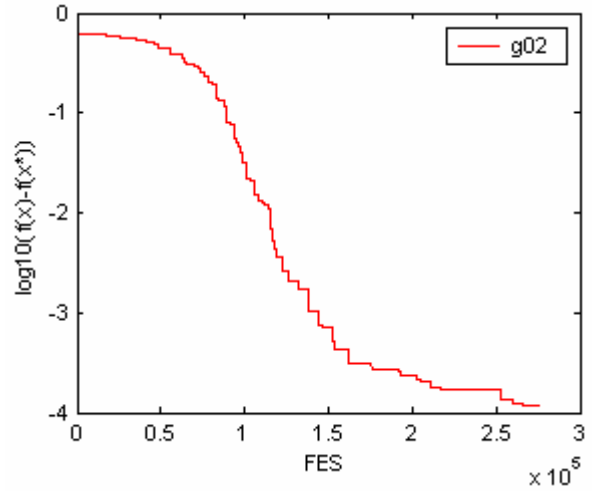

Figure 4. Convergence graph for g02.

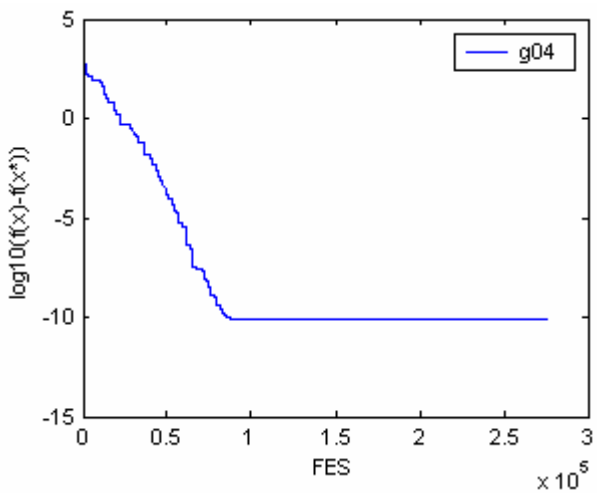

Figure 5. Convergence graph for g04.

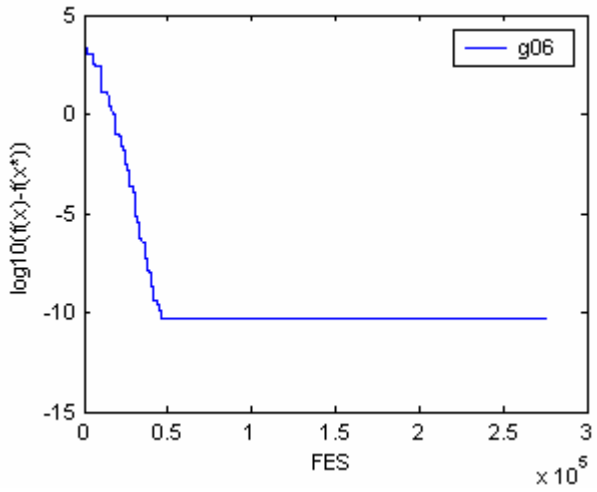

Figure 6. Convergence graph for g06.

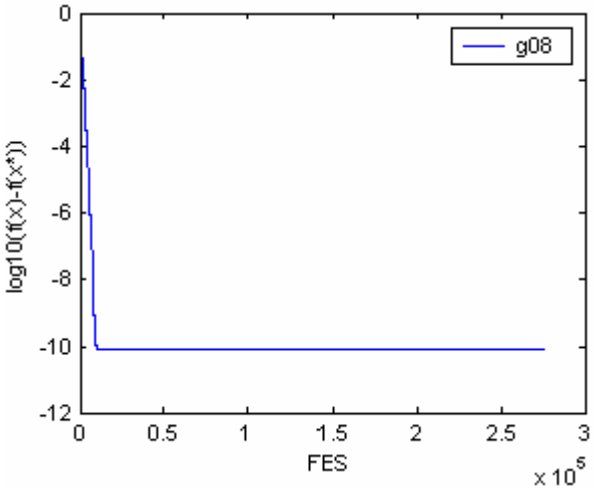

Figure 7. Convergence graph for g08.

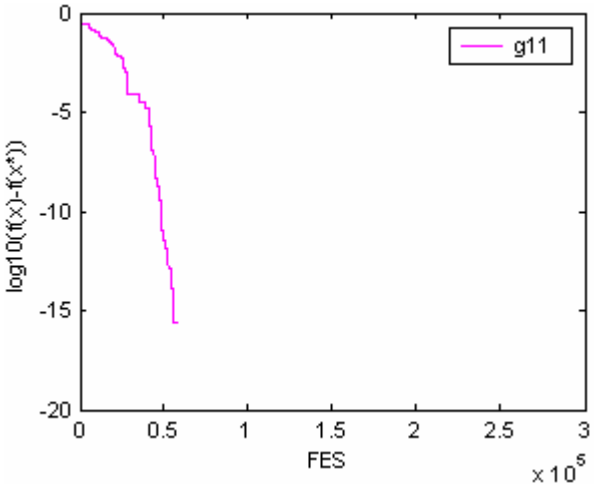

Figure 8. Convergence graph for g11.

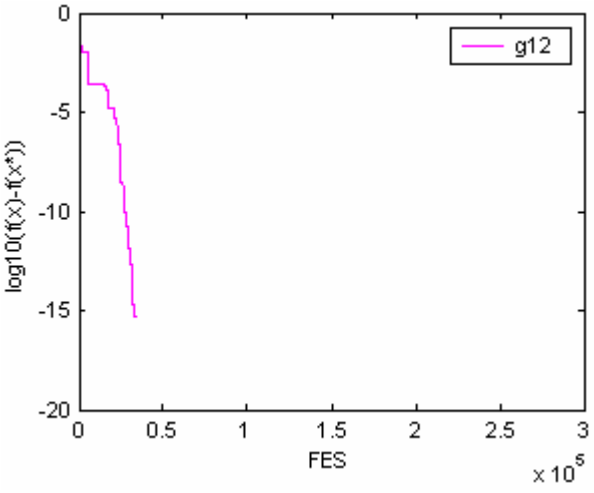

Figure 9. Convergence graph for g12.

VII. CONCLUSIONS AND FUTURE WORK

In this study, we present a dynamic differential evolution algorithm (D-DE) for solving constrained real-parameter optimization problems. In this model of D-DE, there exist at least three important contributions as follows:

1) The first contribution is the novel mutation scheme, which can improve the convergence speed, prevent premature and preserve the diversity of solutions.

2) The second contribution is two important control parameters (i.e., the scale factor F and the crossover probability CR), which are dynamic and beneficial for adjusting control parameters during the evolutionary and search process, especially, when done without any user interaction.

3) The third contribution is that D-DE can prevent premature and enhance the search performance mainly due to replacing some relatively worse solutions with reinitialized solutions during the evolutionary process.

In addition, D-DE employs orthogonal design method to generate initial population to improve the diversity of solutions and introduces a constraint handling technique based on the feasibility rule and the sum of constraints violation.

Finally, D-DE is tested on 6 benchmark test functions provided by the CEC 2006 special session on constrained real-parameter optimization. Through comparing D-DE with respect to state-of-the-art evolutionary algorithms, the experimental results show that D-DE is highly competitive and can obtain good results in terms of a test set of constrained real-parameter optimization problems. However, in the future, there are still many aspects to do. Firstly, in order to further validate D-DE, we are considering of the possibility of testing more benchmark test functions (especially, highly dimensional problems) and real-world constrained optimization problems. Secondly, for some test functions, there exists the phenomenon of slow evolutionary at the later stage. In order to overcome the limitation, we will incorporate some local search techniques into D-DE to improve the convergence speed. Additionally, improving constraint handling technique is another future work.

REFERENCES

[1] V. L. Huang, A. K. Qin, and P. N. Suganthan, "Self-adaptative differential evolution algorithm for constrained real-parameter optimization," in *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, pp. 324-331, IEEE, Vancouver, BC, Canada, July 2006.

[2] A. E. Muñoz-Zavala, A. Hernández-Aguirre, E. R. Villa-Diharce, and S. Botello-Riond, "PESO+ for Constrained Optimization," in *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, pp. 935-942, IEEE, Vancouver, BC, Canada, July 2006.

[3] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, Vol. 186, No. 2, pp. 311-338, 2000.

[4] J. J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, P. N. Suganthan, C. A. Coello Coello, and K. Deb, "Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization," *Technical Report*, Nanyang Technological University, Singapore, 2006.

[5] R. Landa-Becerra, C. A. Coello Coello, "Cultured differential evolution for constrained optimization," *Computer Methods in Applied Mechanic sand Engineering*, Vol. 195, No. 33-36, pp. 4303-4322, 2006.

[6] A. E. Muñoz Zavala, A. Hernández Aguirre, E. R. Villa Diharce, and S. Botello Rionda, "Constrained optimization with an improved particle swarm optimization algorithm," *International Journal of Intelligent Computing and Cybernetics*, Vol. 1, No. 3, pp. 425-453, 2008.

[7] J. J. Liang, P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism," in *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, pp. 316-323, IEEE, Vancouver, BC, Canada, July 2006.

[8] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello, "Modified differential evolution for constrained optimization," in *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, pp. 332-339, IEEE, Vancouver, BC, Canada, July 2006.

[9] T. Takahama, and S. Sakai, "Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites," in *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, pp. 308-315, IEEE, Vancouver, BC, Canada, July 2006.

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

[11] K. Price, R. Storn, J. Lampinen, *Differential Evolution: A Practical Approach To Global Optimization*, Berlin: Springer-Verlag, 2005.

[12] Z. Y. Yang, K. Tang, X. Yao, "Self-adaptive differential evolution with neighborhood search," *2008 Congress on Evolutionary Computation (CEC'2008)*, pp. 1110-1116, 2008.

[13] H. A. Abbass, R. Sarker, C. Newton, "PDE: a Pareto-frontier differential evolution approach for multiobjective optimization problems," in *Proceedings of IEEE Congress on Evolutionary Computation*, Vol. 2, pp. 971-978, 2001.

[14] Y. W. Leung, Y. P. Wang, "An orthogonal genetic algorithm with quantization for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, Vol. 5, No. 1, pp. 40-53, 2001.

[15] J. Brest, V. Zumer, and M. S. Maucec, "Self-adaptative differential evolution algorithm in constrained real-parameter optimization," in *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, pp. 919-926, IEEE, Vancouver, BC, Canada, July 2006.

[16] Y. Wang, Z.X Cai, "A Hybrid Multi-Swarm Particle Swarm Optimization to Solve Constrained Optimization Problems," *Frontiers of Computer Science in China*, Vol. 3, No. 1, pp. 38-52, 2009.

[17] C. A. Coello Coello, "Theoretical and Numerical Constraint-Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art," *Computer Methods in Applied Mechanics and Engineering*, Vol. 191, No. 11-12, pp. 1245-1287, 2002.

[18] T. P. Runarsson, "Approximate evolution strategy using stochastic ranking," in *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, pp. 2760-2767, IEEE, Vancouver, BC, Canada, July 2006.

[19] E. Mezura-Montes, A. G. Palomeque-Qrtiz, "Parameter control in differential evolution for constrained optimization," in *2009 IEEE Congress on Evolutionary Computation (CEC'2009)*, pp. 1375-1382, 2009.

[20] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs,* Springer-Verlag, third edition, 1996.
[21] E. Mezura-Montes, C. A. Coello Coello, "A simple multimembered evolution strategy to solve constrained optimization problems," *IEEE Transactions on Evolutionary Computation*, Vol. 9, No. 1, pp. 1-17, 2005.
[22] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Systems*, Vol. 9, No. 2, pp. 115-148, 1995.
[23] K. Deb and M. Goyal, "A robust optimization procedure for mechanical component design based on genetic adaptive search," *Transactions of the ASME: Journal of Mechanical Design*, Vol. 120, No. 2, pp. 162-164, 1998.

Youyun Ao was born in 1973. He received his B.S. degree in Computer Science from Jiangxi Normal University, Nanchang, China in 1999. He received his M.S. degree in Computer Software and Theory from Shanghai Normal University, Shanghai, China in 2006. He is currently a lecturer in Computer Science at Anqing Teachers College, Anqing, Anhui, China. His research interests include evolutionary computation, intelligent information processing, etc.