International Journal of Security and Its Applications
Vol.9, No.10 (2015), pp.187-200
http://dx.doi.org/10.14257/ijsia.2015.9.10.17

# Constraint Sequential Fault Diagnosis using An Inertial Velocity Differential Evolution Algorithm

Xiaohong Qiu, Yuting Hu and Bo Li

*Software School, Jiangxi University of Science and Technology, Nanchang 330013, PRC;*
*E-mail: jxauqiu@ 163.com, 1016361898@qq.com,libo.jx@163.com*

***Abstract***

*The optimal test sequence design for fault diagnosis is a challenge NP-complete problem. An improved Differential Evolution algorithm with additional inertial weighting item (inertial velocity) is proposed to solve the Optimal Test sequence Problem (OTP) in complicated electronic system. The proposed algorithm called Inertial Velocity Differential Evolution (IVDE) is constructed based on an adaptive differential evolution algorithm. IVDE combined with a new individual fitness function optimizes the test sequence sets with the index of fault isolation rate satisfied in top-down to generate diagnostic decision tree to decrease the test cost and the number of tests used. The simulation results show that IVDE algorithm can cut down the test cost under the satisfied fault isolation rate requirement. Compared with the other algorithm such as PSO (particle swarm optimizer)and GA(genetic algorithm), IVDE can get better solution of the OTP.*

***Keywords:*** *Differential evolution, evolutionary computation, fault isolation rate, testability*

## 1. Introduction

Complex systems with high safety and mission criticality requirements, such as the space shuttle or commercial aircraft with complicated electronic system, consume high maintenance expenditures. The high maintenance cost may often be due to the lack of consideration of testability requirements at the initial design stage and inefficiencies in test strategy. To improve the design for testability, the optimal test sequence design for fault diagnosis is focus on but it is a challenge NP-complete problem [1]. There are dynamic programming and heuristic search algorithm available to solve the problem. The bottom-up dynamic programming with a self-constructed test tree need exponential storage and computational complexity of $O(3^n)$ ($n$ is the number of the test set)[1]. AO* entropy-based heuristic search method is proposed to solve the problem in AND/OR graph search of fault isolation [2,3,4]. The entropy is used to evaluate and select the current best test point in each step of the expansion of the node in the AO* algorithm. This is easy to fall into the local optimal solution. More practical algorithms such as traditional Lagrangian relaxation, near-optimal gradient[5] and bottom-up test sequencing generation methods[6] are studied to solve the middle scale OTP. The intelligent algorithms such as genetic algorithm[7], Discrete Binary Particle Swarm Optimization(DBPSO)[8], quantum-behaved particle swarm optimization[9] application in the optimal diagnostic test strategies are suggested to effectively reduce the testing cost for the large-scale system. After research on the PSO algorithm, a new DBPSO with the adventure factor is used to solve the OTP[10] and has gotten better test sequence. More intelligent algorithms are studied. Firefly algorithm has been also used to solve the software testing problem[11], ant algorithm is used to overcome the computational explosion by setting up ant state transfer rule and feedback[12]. But these algorithms have

ISSN: 1738-9976 IJSIA

to construct a new fitness function which is not directly related to the test cost. And the constraints on the flexible testability index of Fault Detection Rate (FDR) and Fault Isolation Rate(FIR) are not directly concerned with[13].

Differential Evolution (DE) is an effectively random search heuristic evolutionary algorithm. JADE (adaptive differential evolution with optional external archive)[14] and CoDE (composite trial vector generation strategies differential evolution algorithm) [15] are two improved DE with good performance in their test numerical simulation. But they have not been applied to solve the OTP.We try to use the DE to solve this kind of problem.

The organization of the paper is as follows. A general test sequence problem formulation is introduced in section II. In section III, an improved differential evolution called Inertial Velocity Differential Evolution (IVDE) with an additional inertial weighting factor is presented and discussed. And a new fitness function integrated with the test cost with the FDR and FIR is proposed to solve the OTP. IVDE optimizes the test sequence sets with a new individual state representation in top down to generate diagnostic decision tree to decrease the test cost and meet the FDR and FIR requirement. The IVDE used to optimize the examples compared with other algorithm is discussed in section IV. The paper has been concluded in section V.

## 2. Test Sequence Problem Formulation

The Optimal Test sequencing Problem(OTP) known as test planning issues is to design a test sequencing strategy that unambiguously isolates the failure states with minimum expected testing cost to meet the FDR and FIR requirement. The OTP parameters are defined as the five-tuple (*S*,*P*,*T*,*C*,*D*) [1] , where

①$S=\{s_0,s_1,\ldots,s_m\}$ is a set of statistically independent failure states associated with the system. And $s_0$ states that "no fault" state, $s_i(0<i<m+1)$ stands for a different fault state.

②$P=[p(s_0),p(s_1),\ldots,p(s_m)]$ is a priori probability vector associated with the failure states *S*.

③$T=\{t_1,t_2,\ldots,t_n\}$ is a finite set of *n* reliable binary outcome tests, where each test $t_j$ checks a subset of *S*.

④ $C=\{c_1,c_2,\ldots,c_n\}$ is a set of test costs measured in terms of time, manpower requirements, or other economic factors, where $c_j$ is the cost of applying test $t_j$.

⑤$D=[d_{ij}]$ is a binary matrix of dimension $m\times n$ called dependence matrix which represents the relationship between the set of failure states *S* and the set of tests *T*,where $d_{ij}=1$ if test $t_j$ monitors failure state $s_i$; otherwise $d_{ij}=0$.

The OTP parameters of a small scale system are shown in Table 1.

**Table 1. OTP Parameters for Small Scale Example**

| Failure state | Test T | | | | | Probability *P* |
|---|---|---|---|---|---|---|
| | $t_1$ | $t_2$ | $t_3$ | $t_4$ | $t_5$ | |
| $s_0$ | 0 | 0 | 0 | 0 | 0 | 0.01 |
| $s_1$ | 0 | 1 | 0 | 0 | 1 | 0.08 |
| $s_2$ | 0 | 0 | 1 | 1 | 0 | 0.28 |
| $s_3$ | 1 | 0 | 0 | 1 | 1 | 0.20 |
| $s_4$ | 1 | 1 | 0 | 0 | 0 | 0.30 |
| $s_5$ | 1 | 1 | 1 | 1 | 0 | 0.13 |
| Cost *C* | 1.0 | 4.0 | 3.0 | 3.0 | 5.0 | |

In this paper, assumes that there is only one system failure state occurs simultaneously.(Multiple fault diagnosis problem will be discussed in another paper in the future). The problem is to design a test sequence that is able to unambiguously identify

 

the occurrence of any system state in *S* using the tests in the test set *T*, and that minimizes the expected testing cost, *J* , given by (1) under the known condition of (*S*,*P*,*T*,*C*,*D*) with required Fault Isolation Rate(for example, $FIR \geq FIR_{\text{target}}$ ) [1].

$$J = P^T AC = \sum_{i=0}^{m}\sum_{j=1}^{n} a_{ij}\, p(s_i) c_j \tag{1}$$

Where $A=(a_{ij})$ is a binary matrix with the dimension of $(m+1)\times n$ . $a_{ij}=1$ if the failure state $s_i$ identification system used in the course of the test $t_j$, otherwise $a_{ij}=0$ . $FIR_{\text{target}}$ is the lowest requirement of FIR in system.

## 3. Improved Differential Evolution Algorithm to Solve OTP

### 3.1. Differential Evolution (DE) Algorithm

We suppose that the minimized objective function is $f(x_i)$, $x_i=(x_{i,1},\cdots,x_{i,d},\cdots,x_{i,n}) \in R^n$, the feasible solution space is $\Omega=\prod_{i=1}^{n}[x_{i,\min},x_{i,\max}]$. And the initial population $P=\{x_1,...,x_i,\cdots,x_{Np}\}$ is randomly sampled from $\Omega$ , where $N_P$ is the population size and *n* is state space dimension. Differential Evolution (DE) algorithm can be used to deal with the optimization problem. At *k*-th generation, DE creates a mutant vector $v_i=(v_{i,1},\cdots,v_{i,d},\cdots,v_{i,n}) \in R^n$ for each individual $x_i$ in current population. Different mutation operation will play key role to decide the DE performance. One widely used DE mutation operator named *DE/current-to-best/1/bin* [14] is shown as (2)：

$$v_{i,d}^{k+1} = x_{i,d}^{k} + F_i^k \times ((x_{best,d}^{k} - x_{i,d}^{k}) + \lambda \times (x_{r1,d}^{k} - x_{r2,d}^{k})) \tag{2}$$

Where the superscript *k* stands for the *k*-th iteration or *k*-th generation, the subscript *i* stands for the *i*-th individual, and the subscript *d* stands for the *d*-th subdimension. Namely $x_{i,d}^{k}, v_{i,d}^{k}$ is the position and mutant vector in *d*-th subdimension of the *i*-th individual in the *k*-th iteration(or generation). *r*1, and *r*2 are the distinct integers randomly selected from the range [1, $N_P$] and are also different from *i*. The parameter $F_i$ is called the mutation factor, which amplifies the difference vectors.

$x_{best,d}$ is the *d*-th element of the best individual in the current population. After mutation, DE performs a binomial crossover operator on $x_{i,d}$ and $v_{i,d}$ to generate a trial vector $u_{i,d}$ :

$$u_{i,d}^{k} = \begin{cases} v_{i,d}^{k} & if \quad rand(1) \leq C_R \quad or \quad d = rand(n) \\ x_{i,d}^{k} & otherwise \end{cases} \tag{3}$$

Where *i* = 1,2,…, $N_P$, *d* = 1,2,…,*n*, rand(*n*) is a randomly chosen integer in [1,*n*], rand(1) is a uniformly distributed random number between 0 and 1 generated for each individual, and $C_R \in [0,1]$ is called the crossover control parameter. Due to the use of rand(.), the trial vector $u_i$ differs from its target vector $x_i$ .

If the *d*-th element $u_{i,d}$ of the trial vector $u_i$ is out of the boundary, it is reset as follows:

$$u_{i,d} = \begin{cases} \min\{x_{d,\max}, 2x_{d,\min} - u_{i,d}\} & if \quad u_{i,d} < x_{d,\min} \\ \max\{x_{d,\min}, 2x_{d,\max} - u_{i,d}\} & if \quad u_{i,d} > x_{d,\max} \end{cases} \tag{4}$$

The selection operator is performed to select the better one from the target vector $x_i^k$ (*k*-th generation) and the trial vector $u_i^k$ to enter the next generation $x_i^{k+1}$ :

$$x_i^{k+1} = \begin{cases} u_i^k & if \quad f(u_i^k) < f(x_i^k) \\ x_i^k & otherwise \end{cases} \tag{5}$$

### 3.2 Improved DE with Additional Inertial Weighting Item

To combine the good feature of Particle Swarm Optimizer (PSO) with DE to escape from local optimum [16], we suggest an improved DE with additional inertial weighting item of Inertial Velocity Factor called IVDE. After the inertial velocity factor added, the equation (2) is rewritten as (6) and (7).

$$vel_{i,d}^{k+1} = w_{i,d}^{k} \times vel_{i,d}^{k} + F_i^k \times ((x_{best,d} - x_{i,d}^{k}) + \lambda \times (x_{r1,d}^{k} - x_{r2,d}^{k})) \tag{6}$$

$$v_{i,d}^{k+1} = x_{i,d}^{k} + vel_{i,d}^{k} \tag{7}$$

Where the superior $k$ and the subscript $i$ and $d$ have the same meaning as (2). Namely $vel_{i,d}^{k}$ is the speed in $d$-th subdimension of the $i$-th particle in the $k$-th iteration, $w_{i,d}^{k}$ is the inertia weight factor of velocity to escape the local optimum. $F_i$ is the mutation factor, its value estimation method and the "*DE/current-to-pbest/1*" strategy used in JADE algorithm [14] will be used. Namely $x_{best,d}^{p}$ is randomly chosen as one of the top 100 $p$% individuals in the current population with $p \in (0,1]$ instead of $x_{best,d}$. The equation (6) will be:

$$vel_{i,d}^{k+1} = w_{i,d}^{k} \times vel_{i,d}^{k} + F_i^k \times ((x_{best,d}^{p} - x_{i,d}^{k}) + \lambda \times (x_{r1,d}^{k} - x_{r2,d}^{k})) \tag{8}$$

Where $F_i^k$ is associated with $x_i$ and is re-generated at each generation by the adaptation process introduced in (9). At $k$-th generation, the mutation factor $F_i$ of each individual $x_i$ is independently generated according to a Cauchy distribution with location parameter $\mu_F$ and scale parameter 0.1 which is truncated to be 1 if $F_i \geq 1$ or regenerated if $F_i \leq 0$. Suppose $S_F$ as the set of all successful mutation factors in $k$-th generation. The $\mu_F$ is initialized to be 0.5 and then updated at the end of each generation as (10),where $c \in (0,1)$ is a positive constant and $mean_L(\cdot)$ is the Lehmer mean calculated by (11).

$$F_i = randc(\mu_F, 0.1) \tag{9}$$

$$\mu_F = (1-c) \times \mu_F + c \times mean_L(S_F) \tag{10}$$

$$mean_L(S_F) = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F} \tag{11}$$

Similarly, we consider the crossover probability $C_{R,i}$ also has the different weighting feature, and estimated by (12) and (13).

$$C_{R,i} = randn(\mu_{CR}, 0.1) + C_{R,iw} \tag{12}$$

$$\mu_{CR} = (1-c) \times \mu_{CR} + c \times mean_A(S_{CR}) \tag{13}$$

Where $randn(\mu_{CR}, 0.1)$ is a normal distribution of mean $\mu_{CR}$ and standard deviation 0.1, $C_{R,iw}$ is a modified factor with different weighting factor according to their fitness. Denote $S_{CR}$ as the set of all successful crossover probabilities $C_{R,i}$ at $k$-th generation. The mean $\mu_{CR}$ is initialized to be 0.5 and then updated at the end of each generation as (13), where $c \in (0,1)$ is a positive constant and $mean_A(\cdot)$ is the usual arithmetic mean.

Then sort current population in their fitness ascending order $f_{i,sort} = sort_{\substack{Ascending \\ x_i \in P}}\{f(x_i)\}$.

The other parameters will be calculated by the following:

$$C_v(i) = \sin(\frac{f_{i,order}}{N_P}\pi) \tag{14}$$

$$C_{R,iw}(k) = \frac{1+\alpha_1 k}{1+\alpha_2 k} \times 0.04 \times C_v(i) \tag{15}$$

$$w_{i,d}(k) = \frac{1+\alpha_1 k}{1+\alpha_2 k} \times (0.1 rand(0,1) + 0.618) \tag{16}$$

 

$$p(k) = \frac{1+\alpha_1 k}{1+\alpha_2 k} \times p_0 \tag{17}$$

Where $k$ is the iteration times or the objective function evaluations, $\alpha_1, \alpha_2$ are control parameters of the filter to be adjusted with iterations .

### 3.3 Test Sequence Coder for IVDE to Solve OTP

When using IVDE to solve the test sequencing problem, we define individual state $x = (x_1, \cdots, x_i, \cdots, x_n)$ as a test vector which dimension is equal to the number of tests. The value of $x_i$ corresponds to the sequence of the *i*-th test, 1 to *n*. For every individual, the descending order of its vector value can be regarded as a diagnostic sequence design. For example, a five dimension vector *x*={8.1, -6.5, -0.9,6.2,-7.3}, means its diagnostic sequence result is the descending order *T*s={1,4,3,2,5} shown in Figure.1. In the test space, these value denote the test sequence of the test set *T*. Therefore the vector *x* will give a diagnostic strategy { $t_1$->$t_4$->$t_3$ ->$t_2$->$t_5$}.This test sequence can be regarded as a OTP solution of the example mentioned in Table 1.

| | *Test sequence coder* | | | | | |
|---|---|---|---|---|---|---|
| *x* | $x_1$ | $x_2$ | ... | $x_i$ | ... | $x_n$ |
| *Test sequence* | ... | ... | ... | *j* | ... | ... |

| Test set | $t_1$ | $t_2$ | $t_3$ | $t_4$ | $t_5$ |
|---|---|---|---|---|---|
| ***x*** | 8.1 | -6.5 | -0.9 | 6.2 | -7.3 |
| Test sequence | 1 | 4 | 3 | 2 | 5 |

**Figure 1. Individual State for Test Sequence Coder**

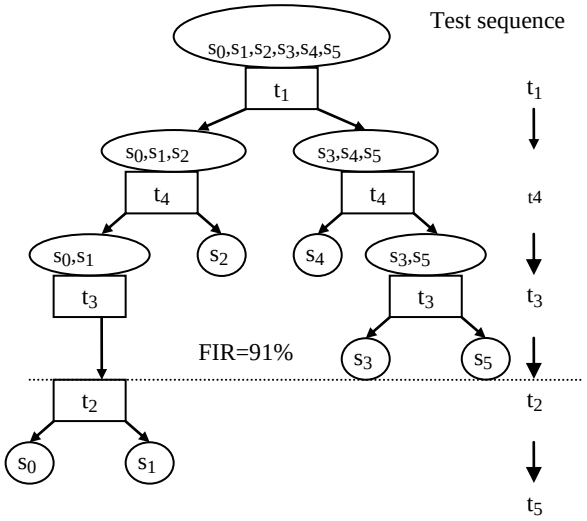


**Figure 2. Fault Isolate Flow Diagram by Test Sequence**

This test sequence { $t_1$->$t_4$->$t_3$ ->$t_2$->$t_5$} is used to isolate the failure states shown in Figure.2. If set $FIR_{\text{target}} = 0.9$ at the beginning, all the failure states are in one group $S$={$s_0$,$s_1$,$s_2$,$s_3$,$s_4$,$s_5$}. Firstly the first test $t_1$ is used to separate $S$ into two ambiguous group, which are {$s_0$,$s_1$,$s_2$} and {$s_3$,$s_4$,$s_5$}. Then use the second test $t_4$ to separate these two ambiguous group, respectively. {$s_0$,$s_1$,$s_2$} is separated into {$s_0$,$s_1$} and {$s_2$}. {$s_3$,$s_4$,$s_5$} is separated into {$s_4$} and {$s_3$,$s_5$}. There are two ambiguous groups and two isolated failure states. So we calculate the FIR=the sum of the probability of isolated failure

states=$p(s_2)+p(s_4)$=0.58. If the $FIR \geq FIR_{\text{target}}$ , the test sequence is enough and terminated to isolate the failure states. Otherwise, the next test is used to separate the ambiguous groups again. $\{s_0,s_1\}$ is failed in separating into two group by the test $t_3$. $\{s_3,s_5\}$ is successful in isolating into failure state $\{s_3\}$ and $\{s_5\}$. Now FIR=$0.58+p(s_3)+p(s_5)$=0.91, it is bigger than $FIR_{\text{target}} = 0.9$ . Therefore , the valid test sequence length $N_t$ is 3 in this OTP example.

### 3.4 The Fitness Function of the IVDE to Solve OTP

A vector $x$ is a diagnostic sequence, which give a diagnostic tree mentioned above, so we can use the cost of diagnostic tree to establish fitness function. Suppose $x$ given, its fault isolation matrix is $A=(a_{ij})$ mentioned in (1), its Fault Detection Rate is FDR and Fault Isolation Rate is FIR. The FDR, FIR and the cost of testing should be integrated into the fitness function of the individual $x_r$ based on (1).

$$f(x_r) = \sum_{i=0}^{m}\sum_{j=1}^{n} a_{ij}\, p(s_i) c_j + \alpha(FDR_{\text{target}} - FDR) + \beta(FIR_{\text{target}} - FIR) + \gamma N_t \qquad (18)$$

Where $N_t$ is the number of the test sequence set $T_s$ used to detect and isolate failure states, $\alpha$ , $\beta, \gamma$ are three weighting coefficients. The factors $\alpha, \beta$ is bigger, the FDR and FIR will be closer to the specified $FDR_{\text{target}}$ and $FIR_{\text{target}}$ . The other symbols are the same meaning as (1). In this paper, $\gamma$ =0.0001 , $FDR_{\text{target}}$=100% , $\alpha$=1(if $FDR_{\text{target}} \geq FDR$), $\alpha$=-1000(if $FDR_{\text{target}} < FDR$) , $\beta$=1(if $FIR_{\text{target}} \geq FIR$), $\beta$=-1000(if $FIR_{\text{target}} < FIR$) . The fitness function calculation detail algorithm is illustrated in Figure.3.

Set $N_t=n, A=(a_{ij})=0$. Set the specified $FDR_{\text{target}}$ and $FIR_{\text{target}}$
Initial failure states set $S=\{s_0,s_1,\ldots,s_i,\ldots s_m\}$, $s_0$ states that "no fault" state, $s_i(1 \leq i \leq m)$ is an independent failure state.
$T_s=\{t_{1s},\ldots,t_{js},\ldots,t_{ns}\}$ is a test sequence determined by gene space vector $x$, $t_{js}$ is the $j$-th test($1 \leq j \leq n$).
For $j$=1 to n
  For each subset which has more than one failure state in $S_{j-1}$ ,create two subset
    -one for those faults $s_i$ which passes the $j$-th test if $d_{ij} = 0$ ;
    -the other for those faults $s_i$ which not passes the $j$-th test if $d_{ij} = 1$.
  If creating two subset is succeeded, the $j$-th test is valid, and set $a_{ij} = 1$.
  $S_j$ =collection of all nonempty subsets.
  Calculate $FIR$ and $FDR$ by judging which subset contains one failure state in $S_j$
  If $FIR>=FIR_{\text{target}}$ and $FDR>=FDR_{\text{target}}$, set $N_t$=j and break
  If each subset in $S_j$ all has only one failure state, set $N_t$=j.
Next $j$.
Calculate the fitness of $T_s$ coded from the vector $x$.

**Figure 3. Fitness Calculation Algorithm in IVDE**

### 3.5 The Complexity of the IVDE to Solve OTP

The complexity of the IVDE to solve OTP is mainly determined by the fitness function calculation. Shown in (18), $A=(a_{ij})$ should be calculated first. A column vector of $A$ is determined by one test. The test sequence will decide the fault isolation matrix $A$ . The detail diagnostic strategy algorithm is illustrated in Figure.3. This complexity of calculating $A=(a_{ij})$ is $O(mn^2)$. Therefore the fitness function calculation complexity is $O(m^2n^3)$. It is smaller than the complexity $O(3^n)$ of dynamic programming[1] when $n$ is

 

large enough, so IVDE can be used to generate optimal test sequence to meet the testability analysis requirement of a complex system .

### 3.6 The IVDE Algorithm to solve OTP

Suppose that $FEs$ is the variable of the objective function evaluations, $FE_{\text{smax}}$ is the limit maximum of $FEs$ for the optimal problem to solve. The Inertial Velocity Differential Evolution (IVDE) algorithm to solve OTP is described as the following:

Step 1: Set the target $FIR_{\text{target}}$, the population size $N_{\text{P}}$, the dimension $n$, the value of $FE_{\text{smax}}$, the percentage $p_0$ of $x_{best}^{p}$ in (17), the parameter value of $\alpha_1, \alpha_2$ used in (15)~ (17).

Step 2: Generate random initialization population $P = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$. For each individual $\mathrm{x}_i$, the descending order of its sub vector value is regarded as a diagnostic sequence design and calculate their fitness as shown in Figure.3. Set iteration control variable $FEs = N_{\text{P}}$.

Step 3: Set $i = 1$.

Step 4: Select individual $\mathbf{x}_i$ in current population, calculate its inertial velocity weighting factor $w_{i,d}$ and crossover probability $C_{R,iw}$ by (16) and (15) respectively. Then calculate $C_R$ by (12) and $F_i$ by (9) .

Step 5: Calculate the variation $v_{\text{i}}$ of individual $x_{\text{i}}$ by (8) and (7) , calculate its fitness as shown in Figure 3 and update $u_{\text{i}}$ by (3) and (4).

Step 6: Select the optimum assignment from $\{u_{\text{i}}, x_{\text{i}}\}$ according to greedy selection mechanism by (5), update individuals to be the next generation of the population, and record the successful individuals to external storage to form $S_{\text{F}}$ and $S_{\text{CR}}$.

Step 7: $i = i + 1$. If $i > N_{\text{P}}$ ,go to Step 8; otherwise, go to Step 4.

Step 8: Set $FEs = FEs + N_{\text{P}}$.

Step 9: All individuals are regarded as the new generation population and form the successful set $S_F$ and $S_{\text{CR}}$, calculate the $\mu_F$ by (10), $F_{\text{i}}$ by (9), $\mu_{\text{CR}}$ by (13).

Step 10: Sort current population according to their fitness in ascending order, randomly select one of the top 100 $p$% individuals as $\mathbf{x}_{best}^{p}$ .

Step 11: When $FEs < FE_{\text{smax}}$, go to Step 3; otherwise, the algorithm stops, the best individual will be regarded as the optimal solution.

## 4. Numerical Experiment Results

IVDE algorithm to solve OTP is programmed in Matlab environment. Experiments run on a PC with 2.93 GHz Intel dual-core CPU, and Win7 operating system. The parameters of IVDE algorithm are set as $n$=the number of test set, the feasible solution space $\Omega$ is limited in $x_{i,\min} = -10, x_{i,\max} = 10$ , the max number of generations is 200, $FE_{\text{smax}}$ =20000, population size $N_{\text{P}} = 100$, $\lambda$=1.123 ,$c = 0.1$, $p_0 = 0.35$, $\alpha_1$=0.00011 , $\alpha_2$=0.00067 . The initial value of $\mu_{CR}$ and $\mu_F$ are 0.5. In this section, three classic examples are chosen to test IVDE.

### 4.1 APOLLO Prelaunch Checkout Example[1]

APOLLO prelaunch checkout example [1] is always used to check computer-based diagnostic aid. In this system, there are 10 failure states and 15 tests with the required $FIR_{\text{target}} = 100\%$ . The system is assumed to be in one of the failure states; that is, the probability of no-fault condition $p(s_0)$. This situation is very common in field maintenance, where the object of fault diagnosis is to identify the failure source, given that the system built-in testing and monitoring aids have detected the presence of a fault during system operation. The given binary test matrix $D$ along with the probability $P$ of

failure states and test costs *C* is shown in Table.2. The optimal solution is shown in Table 3 which IVDE independently run 15 times to solve the OTP in Matlab.

**Table 2. Test Parameters for APOLLO Prelaunch Checkout [1]**

| Failure state | Test T | | | | | | | | | | | | | | | Probability *P* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $t_1$ | $t_2$ | $t_3$ | $t_4$ | $t_5$ | $t_6$ | $t_7$ | $t_8$ | $t_9$ | $t_{10}$ | $t_{11}$ | $t_{12}$ | $t_{13}$ | $t_{14}$ | $t_{15}$ | |
| $s_0$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.0 |
| $s_1$ | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0.1 |
| $s_2$ | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0.1 |
| $s_3$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0.1 |
| $s_4$ | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0.1 |
| $s_5$ | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0.1 |
| $s_6$ | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0.1 |
| $s_7$ | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0.1 |
| $s_8$ | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0.1 |
| $s_9$ | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0.1 |
| $s_{10}$ | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0.1 |
| Cost *C* | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |

**Table 3. Optimal Solution on APOLLO Prelaunch Checkout Example**

| ~~Result~~ | *Optimal test sequencing set Ts* | *Test Cost J* | *Fitness* |
|---|---|---|---|
| *IVDE* | *Ts1={$t_{15}$ ,$t_7$,$t_8$,$t_9$}*<br>*Ts2={$t_{11}$ ,$t_3$,$t_{13}$,$t_{12}$}*<br>*Ts3={$t_5$ ,$t_8$,$t_6$,$t_1$}*<br>*Ts4={$t_{15}$ ,$t_{12}$,$t_{11}$,$t_{10}$}*<br>*Ts5={$t_8$ ,$t_9$,$t_4$,$t_{15}$}*<br>*Ts6={$t_9$ ,$t_{11}$,$t_3$,$t_2$}*<br>*Ts7={$t_{12}$ ,$t_{13}$,$t_9$,$t_{15}$}*<br>*Ts8={$t_5$ ,$t_8$,$t_{10}$,$t_{11}$} ……*<br>*Ts1,Ts2, … have the same test cost and fitness.* | *3.4000±1.81E-15* | *3.4004±1.81E-15* |

The results shown in Table.3 compared with HEF1[1] are the same. And IVDE can almost get 15 different optimal test sequence but the same test cost and fitness in 15 run times.

### 4.2 Anti-tank System [13]

Anti-tank system is a guided missile system primarily designed to hit and destroy heavily armored military vehicle. The system has 13 failure state and 12 tests. The parameters of failure state probability and test cost is shown in Table 4[13]. The optimal solution is shown in Table 5 which IVDE independently run 15 times to solve the OTP in Matlab.

Shown in Table 5,the performance of the proposed method is compared with the self-adaptive test optimizing DPSO algorithm(SADPSO). SADPSO is regarded as the best algorithm discussed in [13]. Compared to SADPSO, the IVDE consumes the least test cost in the same FIR target. IVDE is more efficient than SADPSO.

**Table 4. OTP Parameters for Anti-tank System[13]**

| Failure state | **Test T** | | | | | | | | | | | | Probability P |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $t_1$ | $t_2$ | $t_3$ | $t_4$ | $t_5$ | $t_6$ | $t_7$ | $t_8$ | $t_9$ | $t1_0$ | $t_{11}$ | $t_{12}$ | |
| $s_0$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.10 |
| $s_1$ | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0.12 |

 

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $s_2$ | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0.07 |
| $s_3$ | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0.08 |
| $s_4$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0.09 |
| $s_5$ | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0.08 |
| $s_6$ | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0.11 |
| $s_7$ | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0.13 |
| $s_8$ | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0.09 |
| $s_9$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0.04 |
| $s_{10}$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0.02 |
| $s_{11}$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0.02 |
| $s_{12}$ | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0.05 |
| Cost C | 1.0 | 1.0 | 2.2 | 1.3 | 1.5 | 10 | 1.0 | 2.0 | 3.9 | 2.8 | 0.8 | 2.3 | |

**Table 5. Algorithm Comparison in Anti-tank System [13]**

| | FIR targets | >=95% | >=90% | >=85% | >=80% | >=75% | >=70% |
|---|---|---|---|---|---|---|---|
| SADPSO-C | FIR | 100% | 94% | 89% | 83% | 79% | 73% |
| | Test Number | 11 | 10 | 10 | 9 | 9 | 8 |
| | Cost | 4.86 | 4.63 | 4.56 | 4.32 | 4.14 | 3.9 |
| IVDE | FIR | 100% | 94% | 89% | 83% | 79% | 73% |
| | Test Number | 11 | 10 | 10 | 9 | 9 | 8 |
| | Cost | 4.75 | 4.52 | 4.44 | 4.21 | 4.02 | 3.9 |

### 4.3 Super-heterodyne Receiver Complicated System

The test sequence of the super-heterodyne receiver of the radar system[1] with 36 different tests and 22 failure states is always used to evaluate a new algorithm. Its failure states and test relationship matrix *D*, failure priori probability *P* and test costs *C* are illustrated in detail in Table.IV in[1]. We also use this example to verify IVDE algorithm further.IVDE algorithm use the same parameters mentioned above except $n$=36, $FE_{\text{smax}} = 0.25n \times 10^4$. The optimal solution is shown in Table.6. To compare the results with $FIR_{\text{target}} = 100\%$, the solution of the problem in [7,8,9,10] are shown in Table.7.

**Table 6. Algorithm Comparison in Super-heterodyne Receiver [13]**

| | $FIR_{\text{target}}$ | >=90% | >=80% | >=70% | >=60% |
|---|---|---|---|---|---|
| SADPSO-C | FIR | 94.65% | 80.26% | 74.69% | 60.26% |
| | Test Number | 9 | 13 | 10 | 10 |
| | Cost | 3.26 | 3.13 | 3.04 | 2.68 |
| IVDE | FIR | 93.53% | 80.02% | 74.85% | 60.04% |
| | Test Number | 6 | 13 | 4 | 9 |
| | Cost | 3.25 | 3.16 | 2.84 | 2.68 |

In Table 6, the performance of the proposed method IVDE is compared with the SADPSO [13]. All the comparisons are carried out under the same FIR constrains, i.e.,90%,80%,70,and 60%. The first line of the table shows the FIR requirements(targets). The test sequence generated by all algorithms should have a FIR no less than the required target. The remained part of Table 6 shows all algorithm's performance including the actual FIR(FIR result),different test number and test cost. For example, IVDE generate a test sequence with 3.25 test cost and FIR=93.53% when the required FIR index is no less than 90%,and only need 6 different test but SADPSO need 9 different test. The solution of only 6 different test to achieve the FIR>90% by IVDE is shown in Figure 4.

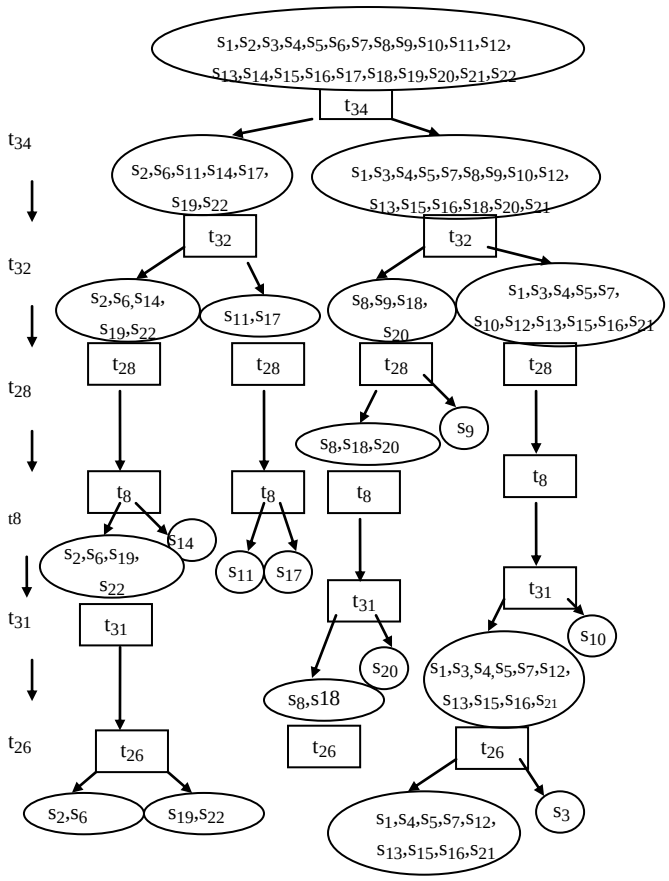


**Figure. 4. The Solution of 6 Test to Achieve the FIR>90% by IVDE**

**Table 7. Optimal Solution on the Superheterodyne Receiver**

| *Result* / *Algorithm* | ***Optimal test sequencing set Ts*** | ***Test Cost*** | ***Fitness*** |
|---|---|---|---|
| *DPSO-AO*[8]* | $t_{30},t_{26},t_{31},t_{34},\mathbf{t_8},t_{11},t_4,t_{21},t_{22},t_{12},t_{28},\mathbf{t_8},t_{19}$ | *3.02* | |
| *GA[7]* | $t_{34},t_{28},t_8,t_{26},t_{31},t_{32},t_{19},t_{21},t_{27},t_5,t_{14},t_{22},t_{12},t_{13}$ | *3.9526* | |
| *DBPSO-AO*[10]* | $t_{34},t_7,t_8,t_{26},t_{32},t_{14},t_{21},t_{19},t_{31},t_{28},t_{13},t_{35},t_{29},t_{22},t_{36}$ | *3.52* | |
| *QPSO[9]* | $t_{32},t_{34},t_{28},t_{31},t_8,t_{26},t_{13},t_{14},t_5,t_{12},t_{21},t_{27},t_{19},t_{22}$ | *3.3904* | |
| *IVDE* | *Ts1=*$\{t_{34},t_8,t_{19},t_{30},t_{28},t_{26},t_{29},t_{32},t_{31},t_7,t_5,t_{21},t_{22},t_{10},t_{14}\}$<br>*Ts2=*$\{t_{34},t_8,t_{19},t_{30},t_{28},t_{29},t_{32},t_{31},t_{26},t_{11},t_{21},t_6,t_{23},t_{15},t_{22}\}$<br>*Ts3=*$\{t_{34},t_8,t_{19},t_{30},t_{28},t_{26},t_{29},t_{32},t_{31},t_{11},t_{21},t_6,t_{22},t_7,t_{35}\}$<br>*Ts1,Ts2 and Ts3 have the same test cost and fitness.* | *3.34735* | *3.34885* |

To achieve 100% FIR, IVDE need 15 test to isolate all the failure states and get the test cost is 3.347. This is better than the test cost=3.9526 in [7] and cost=3.52 in [10]. That means IVDE can get a better solution. Although IVDE have also not gotten a solution test cost less than 3.02 [8], we think this result is unreasonable with optimal test sequencing set because of $t_8$ used twice . We can generate the diagnostic tree by the optimal test sequence calculated by IVDE. If the optimal test sequence is *T*s1 listed in Table.7, its diagnostic decision tree is shown as Figure.5. From the tree in Figure.5, a failure state only need parts of the test set to form the decision tree path to isolate from the others. For example, failure state $s_{19}$ only need test sequence $\{t_{34},t_8,t_{19},t_{26},t_{21}\}$ to isolate from the others. By the tree, the longer path from the root, the failure state priori probability is smaller.

 

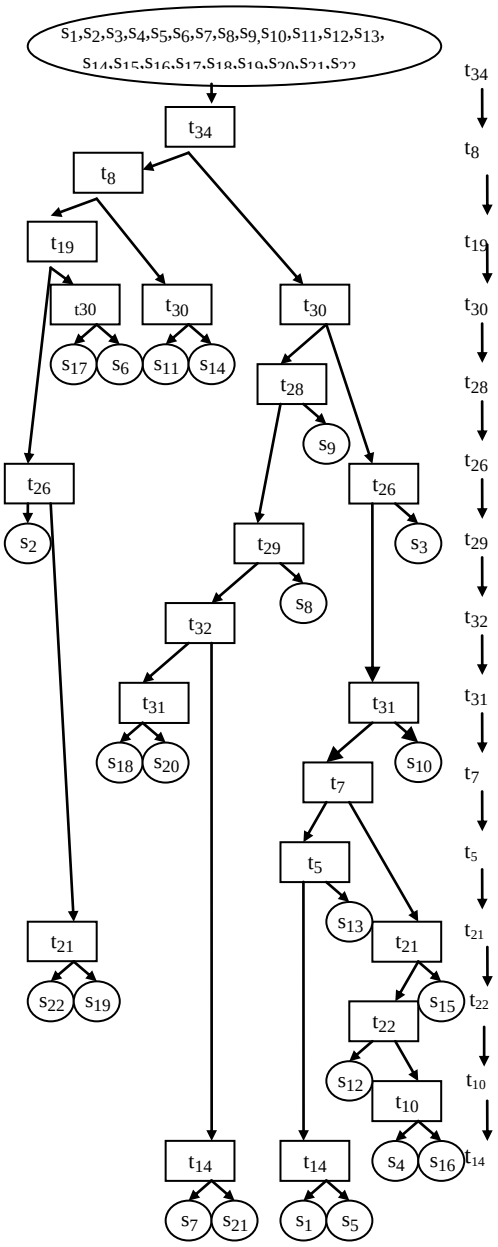


**Figure 5. Diagnostic Decision Tree by Optimal Test Sequence of IVDE**

### 4.4 IVDE Solution Compared with CLPSO, JADE and CoDE

In order to demonstrate the advantage of IVDE, we compare the results of the super heterodyne receiver OTP with that of the Comprehensive Learning Particle Swarm Optimizer (CLPSO)[17] , (JADE)[14] and CoDE[15]. The Matlab code of CLPSO, JADE and CoDE algorithm and their parameter are taken from [15]. Their fitness functions are same as (18). IVDE, CLPSO, CoDE and JADE independently run 15 times to solve the OTP in Matlab R2006 environment,. The average results and standard deviation are shown in Table 8. The convergence graph of IVDE algorithm compared with CLPSO, CoDE and JADE is shown in Figure.6. The convergence of all the three Differential Evolution algorithm (CoDE,JADE and IVDE) is better than the CLPSO . The optimal fitness mean of JADE is not better than CLPSO. But IVDE has gotten better solution and convergence speed than the other's after 2000 *FE*s. That means IVDE has better features than the other algorithm.

**Table 8. Different Method on the Superheterodyne Receiver Example over 15 Independent Runs**

| Algorithm | Optimal Fitness Mean Error±Std Dev |
|---|---|
| *CLPSO* | 3.35311E+00±1.80E-03 |
| *CoDE* | 3.35129E+00±9.06E-16 |
| *JADE* | 3.35452E+00±4.98E-03 |
| *IVDE* | 3.34902E+00±7.56E-04 |

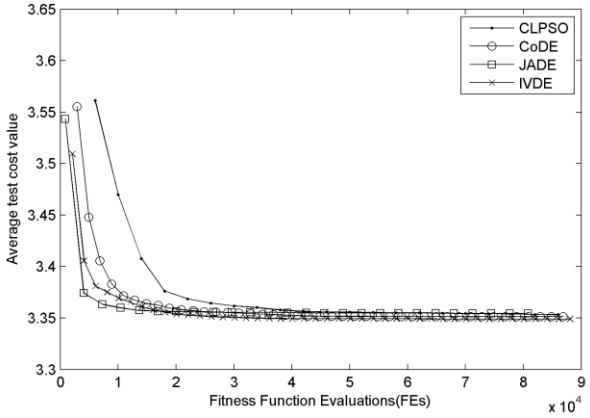


**Figure 6. Convergence of IVDE Compared with CLPSO, CoDE and JADE**

## 5 Conclusions

We have developed a new differential evolution algorithm with additional inertial weighting item called Inertial Velocity Differential Evolution (IVDE) and proposed a new fitness function to solve the optimal test sequencing problems for large scale electronic system. We demonstrated our algorithms on the super heterodyne receiver system[1] and get a better solution with FDR=100% and FIR=100% compared with the other algorithm in the paper [7,9,10]. And compared with SADPSO[13], the IVDE algorithm can get better solution to reduce the test cost and test numbers on the condition that the FIR is predetermined during the system design.

The IVDE algorithm proposed in this paper introduce a inertial velocity factor to escape from local optimum. This is useful to avoid early convergence and increase the probability of searching global optimal solution compared with other DE such as JADE,CoDE. The IVDE used to solve multiple fault diagnosis problem and software testing problem will be discussed in near future.

## Acknowledgments

This work was supported in part by the National Natural Science Foundation of Jiangxi Province, China under Grant No. 20132BAB201044 and in part by Jiangxi Higher Technology Landing Project under Grant No.KJLD12071.The authors would like to thank all anonymous reviewers' valuable comments on this paper.

## References

[1] K R Pattipati, Alexandridis M. Application of Heuristic Search and Information Theory to Sequential Fault Diagnosis. *IEEE Trans. On SMC.* **(1990)** vol.20,no.4,pp.872-887.
[2] V. Raghavan, M. Shakeri, KR Pattipati. Optimal and near optimal test sequencing algorithms with realistic test models. *IEEE Trans. On SMC.* **(1999)** vol.29,no.1,pp.11-27.
[3] M. Shakeri, V. Raghavan, KR Pattipati. Test sequencing Algorithms for Multiple Fault Diagnosis. *IEEE Trans. On SMC.* **(2000)**., vol.30,no.1,pp.1-14

 

[4] Hu Z., Zhang S., Yang Y., Song L., Liu Y., Test sequencing problem considering life cycle cost based on the tests with non-independent cost, *Chemical Engineering Transactions*, **(2013)**,vol.33, pp253-258 DOI: 10.3303/CET1333043.
[5] Fang Tu, KR Pattipati, S. Deb, et al. Computationally Efficient Algorithms for Multiple Fault Diagnosis in Large Graph-Based Systems . *IEEE Trans. On SMC.*, **(2003)** vol.33,no.1,pp.73-85.
[6] O. Erhun Kundakcioglu and Tonguç Ü nlüyurt, Bottom-Up Construction of Minimum-Cost AND / OR Trees for Sequential Fault Diagnosis, *IEEE Trans. On SMC.*, **(2007)** vol.37,no.5,pp.621-629.
[7] Yu Jinsong, Xu Bo, Li Xing-shan. Generation of test strategy for sequential fault diagnosis based on genetic algorithms, *Chinese Journal system simulation*. **(2004)**vol.16,no.4,pp.833-836.
[8] Jiang Rong Hua.Wang Hou Jun. Long Bin. Applying Improved AO* Based on DPSO Algorithm in the Optimal Test Sequencing Problem of Large scale Complicated Electronic System. *Chinese Journal of Computers.* **(2008)** vol. 31,no.10,pp.1835-1840.
[9] Lian Guangyao,Huang Kaoli,Chen Jianhui,Gao Fengqi. Optimization method for diagnostic sequence based on improved particle swarm optimization algorithm[J]. *Chinese Journal of Systems Engineering and Electronics*.**(2009)** vol.20,no.4,pp.899-995.
[10] Qiu Xiaohong, Liu Jun , Xiaohui Qiu, Random DBPSO Algorithm Application in the Optimal Test-sequencing Problem of Complicated Electronic System. 2nd International Conference on Computer and Automation Engineering, ICCAE 2010. Singapore. **(2010)**Feb 26-28,p107-111.
[11] Srivatsava, Praveen Ranjan, Mallikarjun B. , Yang Xin-She. Optimal test sequence generation using firefly algorithm. *Swarm and Evolutionary Computation*,**(2013)** vol.8,no.1, pp. 44-53.
[12] Pan Jia-Liang, Ye Xiao-Hui, Xue Qiang, A New Method for Sequential Fault Diagnosis Based on Ant Algorithm. 2009 Second International Symposium on Computational Intelligence and Design, IEEE,Changsha, (**2009**). vol. 1, pp.44-48.
[13] Yang Chenglin, Yan Junhao, Long Bing, Liu Zhen. A novel test optimizing algorithm for sequential fault diagnosis. *Microelectronics Journal*. **(2014)** vol.45,no.6,pp.719-727.
[14] J. Zhang and A. C. Sanderson. JADE: adaptive differential evolution with optional external archive, *IEEE Trans. Evolut. Comput.*, **(2009)** vol.13,no.5,pp.945-958.
[15] Wang Yong, Cai Zixing, Zhang Qingfu. Differential Evolution With Composite Trial Vector Generation Strategies and Control Parameters. *IEEE Trans. Evolutionary Computation*,**(2011)** vol.15,no.1,pp.55-66.
[16] Kennedy.J, Eberhart.R C. Particle swarm optimization . Proceedings of IEEE Conference on Neural Networks, IV, Piscataway, NJ: IEEE Press, (**1995)**:1942-1948.
[17] J J Liang, A K Qin, P N Suganthan, S Baskar. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Trans. Evolutionary Computation*, **(2006)** vol. 10,no.3,pp. 281~295.

## Authors

**Xiaohong Qiu** was born in Ganzhou City, Jiangxi Province China in 1967. He received the B.S. and M.S. degrees in Automatic Control from the Beijing University of Aeronautics and Astronautics, in 1989,1992 respectively, and the Ph.D degree in flight control, guidance and simulation from Beijing University of Aeronautics and Astronautics in 1995.

From 1995 to 2002, he was a senior engineer and vice general manager with the Institute of Unmanned Vehicle of Beijing University of Aeronautics and Astronautics. Since 2002, he has been Professor of Jiangxi Agricultural University, Jiangxi Normal University. He is the author of three books, more than 70 articles. His research interests include Intelligent Control and Intelligent Computing. He was a recipient of Defense Science and Technology Progress second Award in 2001.

**Yuting Hu** was born in Jiujiang City,Jiangxi Province China in 1990. She received the B.S. degrees in School of Electrical Engineering and Automation from Jiangxi University of Science and Technology in 2013. She is currently a post-graduate student in the Jiangxi University of Science and Technology.

Her current research includes the development of software and intelligent Computing.

**Bo Li** was born in Nanchang City, Jiangxi Province China in 1979. He received the B.S. degrees in School of Electrical Engineering and Automation from Jiangxi University of Science and Technology in 2002, and Master degree from School of Science, Jiangxi University of Science and Technology in 2005.

Since 2005, he is a lecturer of Software School, Jiangxi University of Science and Technology. His current research interest includes the development of software and intelligent algorithm.