International Journal of Automation and Computing 16(3), June 2019, 389-397
DOI: 10.1007/s11633-016-1008-0

Sequential Fault Diagnosis Using an Inertial Velocity Differential Evolution Algorithm

Xiao-Hong Qiu[1] Yu-Ting Hu[1,2] Bo Li[1]

[1]Software School, Jiangxi University of Science and Technology, Nanchang 330013, China

[2]College of Information Engineering, Jiangxi University of Science and Technology, Ganzhou 341000, China

Abstract: The optimal test sequence design for fault diagnosis is a challenging NP-complete problem. An improved differential evolution (DE) algorithm with additional inertial velocity term called inertial velocity differential evolution (IVDE) is proposed to solve the optimal test sequence problem (OTP) in complicated electronic system. The proposed IVDE algorithm is constructed based on adaptive differential evolution algorithm. And it is used to optimize the test sequence sets with a new individual fitness function including the index of fault isolation rate (FIR) satisfied and generate diagnostic decision tree to decrease the test sets and the test cost. The simulation results show that IVDE algorithm can cut down the test cost with the satisfied FIR. Compared with the other algorithms such as particle swarm optimization (PSO) and genetic algorithm (GA), IVDE can get better solution to OTP.

Keywords: Differential evolution (DE), evolutionary computation, fault isolation rate (FIR), testability, fault diagnosis.

1 Introduction

Complex systems with high safety and mission critical requirements, such as the space shuttle or commercial aircraft, consume high maintenance expenditures annually. The high maintenance cost can 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 focused on, but it is a challenging NP-complete problem[1]. There are two kinds of algorithms available to solve the problem: the dynamic programming and heuristic search algorithm. The bottom-up dynamic programming with a self-constructed test tree needs exponential storage and it has 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[2−4]. The entropy is used to select the current best test point for 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], bottom-up test sequencing generation methods[6] and dynamic uncertain causality graph method to model complex behaviors of real-world systems under uncertainties[7] are studied to solve the middle scale optimal test sequence problem (OTP). The intelligent algorithms such as genetic algorithm (GA)[8], discrete binary particle swarm optimization (DBPSO)[9], quantum-behaved particle swarm optimization[10] application in the optimal diagnostic test strategies is suggested to effectively reduce the testing cost for large-scale system. After research on the particle swarm optimization (PSO) algorithm, the swarm behavior should be determined not only by the previous velocity, own experience and the others′ experience, but also by the adventure factor, and DBPSO with this new idea has gotten better test sequence to solve the OTP[11]. More intelligent algorithms are studied, e.g., firefly algorithm has been also used to solve the software testing problem[12], ant algorithm is used to overcome the computational explosion by setting up ant state transfer rule and feedback[13], cultural algorithm is used to optimize the detection of stuck-at faults and crosstalk faults in digital circuits[14]. But these algorithms have 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 and fault isolation rate are not directly concerned with[15]. Differential evolution (DE) is an effective population-based random search heuristic evolutionary algorithm. Adaptive differential evolution with optional external archive (JADE)[16] and composite trial vector generation strategies differential evolution algorithm (CoDE)[17] are two improved DE with good performance in their test numerical simulation. But they have not been applied to solve the OTP.

The organization of the paper is as follows. A general test sequence problem formulation is introduced in Section 2. In Section 3, an improved differential evolution called inertial velocity differential evolution (IVDE) with an additional inertial velocity factor is presented and discussed. And a new fitness function is proposed to integrate the test cost

Research Article
Manuscript received October 27, 2014; accepted July 24, 2015; published online September 2, 2016
Recommended by Associate Editor Chandrasekhar Kambhampati

with the fault detection rate (FDR) and fault isolation rate (FIR) to solve the OTP. IVDE optimizes the test sequence sets to isolate the failure states by a new individual state representation of test sequence and generate diagnostic decision tree to decrease the test cost and meet the FDR and FIR requirements. The IVDE used to optimize the examples compared with other algorithms is discussed in Section 4. The paper has been concluded in Section 5.

2 Test sequence problem

The optimal test sequencing problem 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 requirements. The OTP parameters are defined as the five-tuple (S, P, T, C, D)[1], where $S=\{s_0,\ s_1,\cdots,s_m\}$ is a set of statistically independent failure states associated with the system. And s_0 states that "no fault" state, s_i $(0<i<m+1)$ is a different fault condition.

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

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

$C=\{c_1,c_2,\cdots,c_n\}$ 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.

$D=[d_{ij}]$ is a binary matrix of dimension $(m+1)\times n$ 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

S	Test T					P
	t1	t2	t3	t4	t5	
s0	0	0	0	0	0	0.01
s1	0	1	0	0	1	0.08
s2	0	0	1	1	0	0.28
s3	1	0	0	1	1	0.20
s4	1	1	0	0	0	0.30
s5	1	1	1	1	0	0.13
Cost C	1.0	4.0	3.0	3.0	5.0	

This paper assumes that there is only one system failure state occurs. (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 of T, and that minimizes the expected testing cost J, given by (1) under the known condition of (S, P, T, C, D) with required FIR named FIR_{target}[1].

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

where $A=(a_{ij})$ is a binary matrix with the dimension of $(m+1)\times n$. Let $a_{ij}=1$ if the failure state s_i identification system used in the course of the test t_j, otherwise $a_{ij}=0$. FIR_{target} is the lowest requirement of FIR in system. In general, A is derived from D according to the test sequence.

3 Inertial velocity differential evolution

3.1 Differential evolution algorithm

We suppose that the minimized objective function is $f(x_i)$, $x_i=(x_{i,1},\cdots,x_{i,d},\cdots,x_{i,n})\in\mathbf{R}^n$, the feasible solution space is $\Omega=\prod_{i=1}^{n}[x_{i,\min},x_{i,\max}]$. And the initial population $Pop=\{x_1,\cdots,x_i,\cdots,x_{Npop}\}$ is randomly sampled from Ω, where N_{pop} is the population size. Differential evolution is an effective population-based random search heuristic evolutionary algorithm and can be used to deal with the optimization problem. At the k-th generation, DE creates a mutant vector $v_i=(v_{i,1},\cdots,v_{i,d},\cdots,v_{i,n})\in\mathbf{R}^n$ 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[16] is shown as

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

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, $x_{i,d}^k$ and $v_{i,d}^k$ are the position and mutant vector in the d-th subdimension of the i-th individual in the k-th iteration(or generation). $r1$ and $r2$ are the distinct integers randomly selected from the range [1, N_{pop}] and are also different from i. The parameter F_i is called the mutation factor, which amplifies the difference vectors. λ is a scale factor. $x_{best,d}$ is the d-th element of the best individual in the current population. After mutation, DE applies a binomial crossover operator on $x_{i,d}$ and $v_{i,d}$ to generate a trial vector $u_{i,d}$ as

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

where $i=1,2,\cdots,N_{pop}$, $d=1,2,\cdots,n$, rand(n) is a randomly chosen integer in [1, n], rand(1) is a uniformly distributed random number between 0 and 1 which is generated for each individual, and $C_R\in$[0, 1] is called the crossover control parameter. Due to the use of $rand(\cdot)$, 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

$$u_{i,d} = \begin{cases} \min\{x_{d,\max}, 2x_{d,\min} - u_{i,d}\}, \text{ if } u_{i,d} < x_{d,\min} \\ \max\{x_{d,\min}, 2x_{d,\max} - u_{i,d}\}, \text{ if } u_{i,d} > x_{d,\max}. \end{cases} \quad (4)$$

The selection operator is applied 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} as

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

3.2 Improved DE with additional inertial velocity factor

To combine the good feature of particle swarm optimizer with DE to escape from local optimum[16], we suggest an improved DE with additional inertial velocity factor. After the inertial velocity factor added, (2) is rewritten as (6) and (7).

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

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

where the superscript k and the subscripts i and d have the same meaning as in (2). Namely, $vel_{i,d}^k$ is the speed in d-th subdimension of the i-th particle in the k-th iteration (or generation), $w_{i,d}^k$ is the inertial weight factor of velocity to escape from the local optimum. F_i is the mutation factor, its value is estimated by the method used in JADE algorithm[16], and the "DE/current-to-pbest/1" strategy instead of only adopting the best individual in the "current-to-best/1" strategy. Namely $x_{best,d}^p$ is randomly chosen as one of the top $100\times p\%$ individuals in the current population with $p \in (0, 1]$ instead of $x_{best,d}$. Equation (6) will be

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

$$F_i = randc(\mu_F, 0.1) \quad (9)$$

$$\mu_F = (1 - c) \times \mu_F + c \times mean_L(S_F) \quad (10)$$

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

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 μ_F and scale parameter 0.1, and then truncated to be 1 if $F_i \geq 1$ or regenerated if $F_i \leq 0$. Denote S_F as the set of all successful mutation factors in the k-th generation. The location parameter μ_F of the Cauchy distribution 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).

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

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

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

where $randn(\mu_{CR}, 0.1)$ is a normal distribution of mean μ_{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 $C'_{R,i}$s at generation k. The mean μ_{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 $f_{i,order} = \underset{x_i \in Pop}{sort_{Ascending}}\{f(x_i)\}$. The other parameters will be calculated as

$$C_v(i) = \sin(\frac{f_{i,order}}{N_{pop}}\pi) \quad (14)$$

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

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

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

where k is the iteration times, evaluations α_1 and α_2 are control parameters of the filter to be adjusted with iterations, and p_0 is the initial value of the top percentage of best individuals.

3.3 Test sequence coder for IVDE

When using IVDE to solve the test sequencing problem, we should map the state space to test sequence. We define individual state $x = (x_1, \cdots, x_i, \cdots, x_N)$ as a test vector whose dimension is equal to the number of tests. The value of x_i corresponds to the test number of the i-th test in the test sequence, 1 to N. For every individual, the descending order of its sub vector value can be regarded as a diagnostic sequence design.

Shown in Fig. 1, for example, a five dimension vector is $x = \{8.1, -6.5, -0.9, 6.2, -7.3\}$, its descending order is T_s=$\{1, 4, 3, 2, 5\}$ represented by $\{t_1, t_4, t_3, t_2, t_5\}$. That means it is a diagnostic sequence result in the test space. Therefore, the vector x will give a diagnostic strategy T_s = $\{\ \mathrm{t}_1 \longrightarrow \mathrm{t}_4 \longrightarrow \mathrm{t}_3 \longrightarrow \mathrm{t}_2 \longrightarrow \mathrm{t}_5\}$. This test sequence can be regarded as an OTP solution of the example whose parameters are mentioned in Table 1. If this test sequence is used to isolate the failure states, the diagnostic tree is shown in Fig. 2.

	Test sequence coder					
x	x_1	x_2	…	x_i	…	x_N
Descending order	…	i	…	j	…	…
Test sequence	…	t_i	…	t_j	…	…

Test set	$\{t_1, t_2, t_3, t_4, t_5\}$				
x	8.1	–6.5	–0.9	6.2	–7.3
Descending order	1	4	3	2	5
Test sequence	t_1	t_4	t_3	t_2	t_5

Fig. 1 Individual state vector for test sequence coder

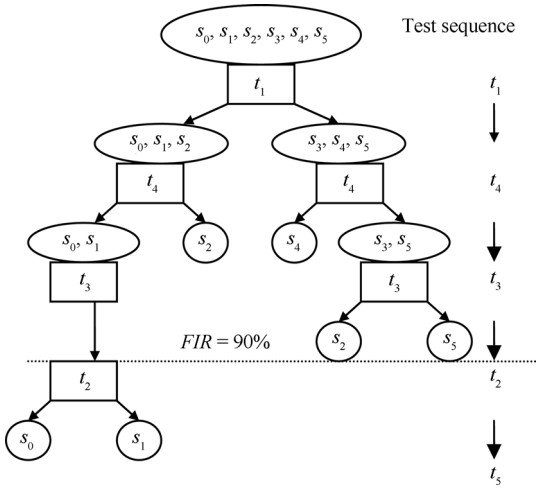


Fig. 2 Fault isolation flow diagram by test sequence

Shown in Fig. 2, if we set $FIR_{\text{target}} = 0.9$, at the beginning, all the failure states are in one group $S = \{s_0, s_1, s_2, s_3, s_4, s_5\}$. The first test t_1 of $\{ t_1 \longrightarrow t_4 \longrightarrow t_3 \longrightarrow t_2 \longrightarrow t_5\}$ is used to separate S into two ambiguous groups, which are $\{s_0, s_1, s_2\}$ and $\{s_3, s_4, s_5\}$. Then the second test t_4 is used to separate these two ambiguous groups. $\{s_0, s_1, s_2\}$ 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 actual $FIR \geq FIR_{\text{target}}$, the test sequence is enough and terminated to isolate the failure states. Otherwise, the next test is used to separate the ambiguous groups again. The group $\{s_0, s_1\}$ cannot be further separated into two groups by the test t_3, but $\{s_3, s_5\}$ can be separated 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 $\boldsymbol{x}$ stands for a diagnostic sequence, which gives a diagnostic tree as mentioned above, so we can use the cost of diagnostic tree to establish fitness function. Suppose $\boldsymbol{x}$ is 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).

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

Set N_t=n, A=(a_{ij})=0.

Set the specified FDR_{target} (default 100%) and FIR_{target}

Initial failure states S=$\{s_0, s_1, \cdots, s_i, \cdots, s_m\}$, where s_0 means "no fault" state, s_i ($1 \leq i \leq m$) is an independent failure state.

T_s=$\{t_{1s}, \cdots, t_{js}, \cdots, t_{ns}\}$ 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 subsets

1)one for those faults s_i which pass the j-th test if $d_{ij} = 0$;

2)the other for those faults s_i which do not pass the j-th test if $d_{ij} = 1$.

If creating two subsets 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_{target} and FDR>=FDR_{target}, set N =j and break

If all subsets in S_j have only one failure state, set N_t=j.

Next j.

Calculate the fitness of T_s coded from the vector x according to (18).

Fig. 3 Fitness calculation algorithm pseudocode

where N_t is the number of the test sequence set of T_s used to detect and isolate failure states, α, β, γ are three weighting coefficients. In (18), N_t is added to minimize the test set. A bigger N_t means that it needs more test points to detect and isolate failure states. The bigger are the factors of α and β, the FDR and FIR will be closer to the specified FDR_{target} and FIR_{target}. The other symbols have the same meaning as in (1). In this paper, α=1, β=1, γ=0.000 1, FDR_{target}=100% . The fitness function calculation detailed algorithm pseudocode is illustrated in Fig. 3. The complexity of the IVDE to solve OTP is mainly determined by the fitness function calculation. As 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 detailed diagnostic strategy algorithm is illustrated in Fig. 3. The 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.

3.5 IVDE algorithm to solve OTP

Suppose that FEs is the variable of the objective (or fitness) function evaluations, FEsmax is the maximum limit of FEs for the optimal problem to solve. The inertial velocity differential evolution algorithm is described as follows:

Step 1. Set the value of FIR_{target}, the population size N_{pop}, the dimension n, the value of FEsmax, the percentage p_0 of $\mathbf{x}_{best}^p$ in (17), the parameter values of α_1, α_2 used in (15)−(17).

Step 2. Generate random initialization population $Pop = \{x_1, \cdots, x_i, \cdots, x_{Npop}\}$. For each individual x_i, the descending order of its sub vector value is regarded as a diagnostic sequence design and their fitness can be calculated as shown in Fig. 3. Set iteration control variable $FEs = N_{pop}$.

Step 3. Set $i = 1$.

Step 4. Select individual 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_i of individual x_i by (8) and (7), calculate its fitness as shown in Fig. 3 and update u_i by (3) and (4).

Step 6. Select the optimum assignment from $\{u_i,\ x_i\}$ according to greedy selection mechanism by (5), update individuals to be the next generation of the population, and record the successful individuals in an external storage to form S_F and S_{CR}.

Step 7. $i = i + 1$. If $i > N_{pop}$, go to Step 8. Otherwise, go to Step 4.

Step 8. Set $FEs = FEs + N_{pop}$.

Step 9. All individuals are regarded as the new generation population, calculate the μ_F by (10), F_i by (9), μ_{CR} by (13).

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

Step 11. When $FEs < FE$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 elements of test set, the feasible solution space is $\Omega = \prod_{i=1}^{n} [x_{i,\min},\ x_{i,\max}]$, $x_{i,\min} = -10$, $x_{i,\max} = 10$, the max number of generations is 200, FEsmax =20 000, population size $N_{pop} = 100$, $\lambda = 1.123$, $c = 0.1$, $p_0 = 0.35$, $\alpha_1 = 0.000\,11$, $\alpha_2 = 0.000\,67$. The initial values of μ_{CR} and μ_F are 0.5. In this section, three OTP examples are chosen to test IVDE in Matlab.

4.1 Apollo prelaunch checkout[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 $FIR_{target} = 100\%$. The system is assumed to be in one of the failure states, i.e., the probability of no-fault condition $p(s_0) = 0$. This situation is very common in field maintenance, where the objective 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 are shown in Table 2. The optimal solution is shown in Table 3, where IVDE has independently run 15 times to solve the OTP in Matlab.

Table 2 Test parameters for APOLLO prelaunch checkout [1]

S	Test T															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
C	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

The results shown in Table 3 compared with huffman code-based heuristic eualuation functions (HEF1)[1] are the same. The average test cost value is $3.400\,0 \pm 1.81 \times 10^{-15}$ in 15 run times. Because in this OTP shown in Table 2, the different failure states have the same priori probability and the different test costs also have same value, its optimal solution will be more than one in general. And IVDE can almost get 15 different optimal test sequences but almost the same test cost and fitness in 15 runs.

Table 3 Optimal solution on APOLLO prelaunch checkout

Algorithm	Optimal test sequencing set Ts	Test cost
IVDE	Ts_1={t_{15},t_7, t_8, t_9}	$3.4000 \pm 1.81 \times 10^{-15}$
	Ts_2={ t_{11} , t_3, t_{13}, t_{12}}	
	Ts_3={ t_5 , t_8, t_6, t_1}	
	Ts_4={ t_{15}, t_{12}, t_{11}, t_{10}}	
	Ts_5={ t_8, t_9, t_4, t_{15}}	
	Ts_6={ t_9, t_{11}, t_3, t_2}	
	Ts_7={ t_{12}, t_{13}, t_9, t_{15}}	
	Ts_8={ t_5, t_8, t_{10}, t_{11}}	
	...	

4.2 Anti-tank system[15]

Anti-tank system is a guided missile system primarily designed to hit and destroy heavily armored military vehicle. In this system, there are 13 failure states and 12 tests. The parameters of failure state probability and test cost are shown in Table 4[15]. In this OTP, the different failure state has a different probability, and different tests have different test costs. The optimal solutions compared with the self-adaptive test optimizing DPSO algorithm (SADPSO)[15] are shown in Table 5, where IVDE has independently run 15 times to solve the OTP in Matlab. And all the comparisons are carried out under the same FIR constraints, i.e., 95%, 90%, 80%. The first line of Table 5 shows the FIR requirements, the test sequence generated by all algorithms should have a FIR no less than the required. The other lines shows all algorithm′s performance including the actual FIR (FIR result), different number of test sequence set and test cost.

Table 4 NMI scores of different algorithms

S	Test T												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}	
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	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	1.0	1.0	2.2	1.3	1.5	10	1.0	2.0	3.9	2.8	0.8	2.3	

As shown in Table 5, the performance of the proposed method is compared with SADPSO. SADPSO is regarded as the best algorithm discussed in [15]. Compared to SADPSO, the IVDE consumes less test cost in the same FIR_{target}. For example,under the $FIR_{\text{target}} = 95\%$ constraint, both algorithms need 11 tests to get the FIR, 100% bigger than FIR_{target}, but IVDE consumes test cost of 4.75 and SADPSO needs test cost of 4.86.

4.3 Super-heterodyne receiver complicated system

The test sequence design of the super-heterodyne receiver of the radar system[1] is always used to evaluate a new algorithm performance. The system consists of 36 different tests and 22 failure states. Its failure states and test relationship matrix D, failure priori probability P and test costs C are illustrated in Table 4 in [1]. IVDE is verified further by this example and used the same parameters as mentioned above except $n = 36, FEs\max = 0.25n \times 10^4$.

Table 5 Algorithm comparison in anti-tank system[15]

Algorithm	FIR	$\geq 95\%$	$\geq 90\%$	$\geq 85\%$	$\geq 80\%$	$\geq 75\%$	$\geq 70\%$
SADPSO-C	FIR	100%	94%	89%	83%	79%	73%
	Tests	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%
	Tests	11	10	10	9	9	8
	Cost	4.75	4.52	4.44	4.21	4.02	3.9

In Table 6, the performance of IVDE is compared with the SADPSO[15]. All the comparisons are carried out under the same FIR constraints, i.e., 90 %, 80 %, 70 % and 60 %. The test sequence generated by all algorithms should have a FIR no less than the required. For example, IVDE generates a test sequence with test cost of 3.25 and $FIR = 93.53\,\%$ when the required FIR index is no less than 90%, and it only needs 6 different tests but SADPSO needs 9 different tests. The solution of only 6 different tests to achieve the $FIR > 90\,\%$ by IVDE is shown in Fig. 4. To compare the solution of the problem in [8, 9, 10, 11], solutions with $FIR_{\text{target}} = 100\%$ solved by IVDE are shown in Table 7. To achieve 100 % FIR, IVDE needs 15 tests to isolate all the failure states and get the test cost of 3.347. This is better than the test cost of 3.952 6 in [8] and cost of 3.52 in [11]. That means IVDE can get a better solution. Although IVDE has also not gotten a solution with test cost less than 3.02 in [9], we think this result is unreasonable with the optimal test sequencing set because of t_8 used twice.

Table 6 Algorithm comparison in super-heterodyne receiver[15]

Algorithm	FIR	$\geq 90\%$	$\geq 80\%$	$\geq 70\%$	$\geq 60\%$
SADPSO-C	FIR	94.65%	80.26%	74.69%	60.26%
	Tests	9	13	10	10
	Cost	3.26	3.13	3.04	2.68
IVDE	FIR	93.53%	80.02%	74.85%	60.04%
	Tests	6	13	4	9
	Cost	3.25	3.16	2.84	2.68

In Table 7, IVDE has achieved different optimal test sequence sets of Ts_1, Ts_2 and Ts_3, which have the same test cost and fitness. If the optimal test sequence is {34, 8, 19, 30, 28, 26, 29, 32, 31, 7, 5, 21, 22, 10, 14}, its diagnostic decision tree is shown as Fig. 5. From the tree in Fig. 5, a failure state only needs part of the test set to form the decision tree path to isolate from the others. For example, failure state s_{19} only needs test sequence $\{t_{34}, t_8, t_{19}, t_{26}, t_{21}\}$ to isolate from the others. It is established from the tree, the longer is the path from the root, the failure state priori probability is smaller.

Table 7 Optimal solution on the super heterodyne receiver

Algorithm	Optimal test sequencing Ts	Cost J
DPSO-AO*[9]	$t_{30}, t_{26}, t_{31}, t_{34}, t_8, t_{11}, t_4, t_{21}, t_{22}, t_{12}, t_{28}, t_8, t_{19}$	3.02
GA[8]	$t_{34}, t_{28}, t_8, t_{26}, t_{31}, t_{32}, t_{19}, t_{21}, t_{27}, t_5, t_{14}, t_{22}, t_{12}, t_{13}$	3.952 6
DBPSO-AO*[11]	$t_{34}, t_7, t_8, t_{26}, t_{32}, t_{14}, t_{21}, t_{19}, t_{31}, t_{28}, t_{13}, t_{35}, t_{29}, t_{22}, t_{36}$	3.52
QPSO[10]	$t_{32}, t_{34}, t_{28}, t_{31}, t_8, t_{26}, t_{13}, t_{14}, t_5, t_{12}, t_{21}, t_{27}, t_{19}, t_{22}$	3.390 4
IVDE	$Ts_1 = \{t_{34}, t_8, t_{19}, t_{30}, t_{28}, t_{26}, t_{29}, t_{32}, t_{31}, t_7, t_5, t_{21}, t_{22}, t_{10}, t_{14}\}$ $Ts_2 = \{t_{34}, t_8, t_{19}, t_{30}, t_{28}, t_{29}, t_{32}, t_{31}, t_{26}, t_{11}, t_{21}, t_6, t_{23}, t_{15}, t_{22}\}$ $Ts_3 = \{t_{34}, t_8, t_{19}, t_{30}, t_{28}, t_{26}, t_{29}, t_{32}, t_{31}, t_{11}, t_{21}, t_6, t_{22}, t_7, t_{35}\}$	3.347 35

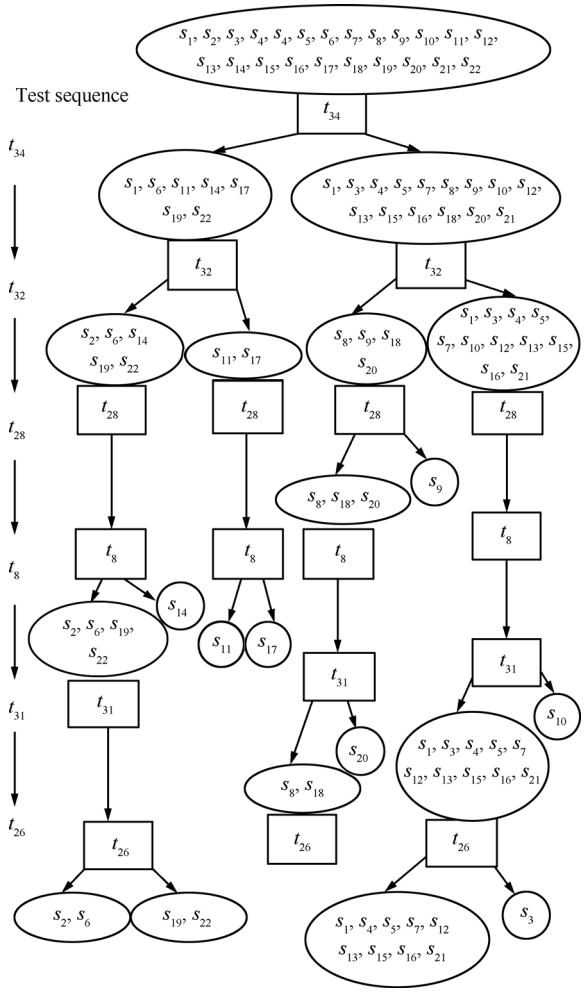


Fig. 4 Solution of 6 tests to achieve FIR >90% by IVDE

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)[19] , JADE [16] and CoDE[17]. The Matlab code of CLPSO, JADE and CoDE algorithms and their parameters are taken from [17]. Their fitness functions are the same as (18). Using a PC with Intel dual-core CPU, and Win7 operating system, IVDE, CLPSO, CoDE and JADE are independently run 15 times to solve the OTP in Matlab R2006 environment. The test cost 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 Fig. 6. The convergence of the three differential evolution algorithms (CoDE, JADE and IVDE) is better than the CLPSO. The average optimal test cost of JADE is not better than CLPSO. But IVDE has gotten better solution and convergence speed than the other′s after 2 000 FEs. That means IVDE has better features than the other algorithms.

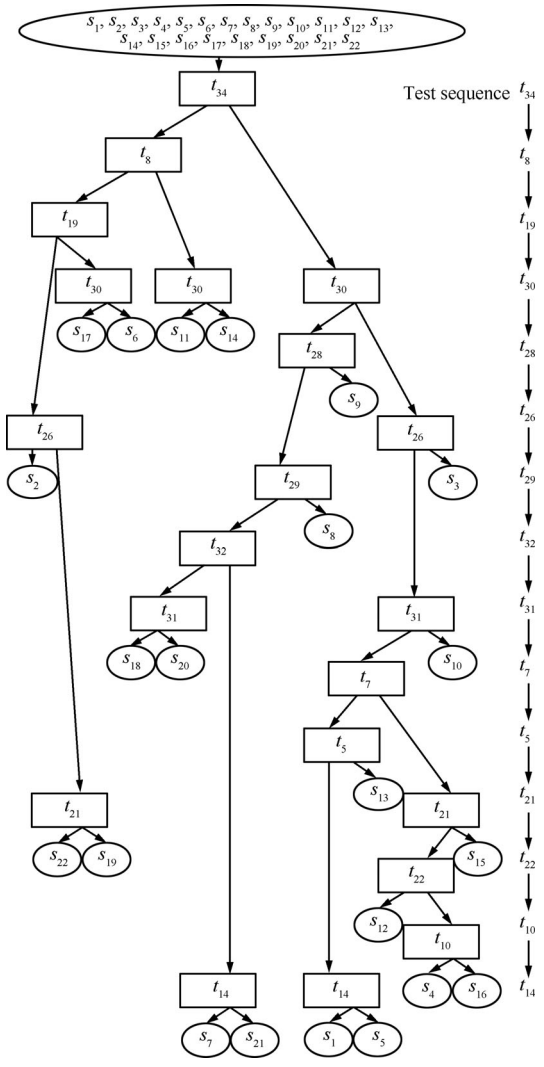


Fig. 5 Diagnostic decision tree by optimal test sequence of IVDE

Table 8 Average test cost on the super heterodyne receiver over 15 independent runs

Algorithm	Optimal average test cost
CLPSO	$3.353\,11\times10^{0}\pm1.80\times10^{-3}$
CoDE	$3.351\,29\times10^{0}\pm9.06\times10^{-16}$
JADE	$3.354\,52\times10^{0}\pm4.98\times10^{-3}$
IVDE	$3.349\,02\times10^{0}\pm7.56\times10^{-4}$

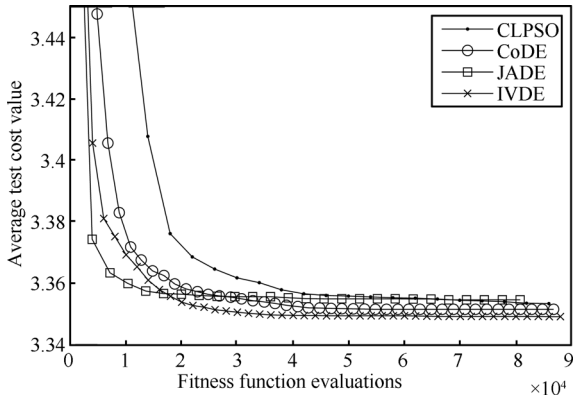


Fig. 6 Convergence of IVDE , CLPSO, CoDE and JAD

5 Conclusions

We have developed a new differential evolution algorithm with additional inertial velocity factor 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 algorithms in the paper [8, 10, 11]. And compared with SADPSO[15], the IVDE algorithm can get better solution to reduce the test cost and number of tests under the condition of the specified FIR during the system design.

The IVDE algorithm proposed in this paper introduces an 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 by National Natural Science Foundation of Jiangxi Province, China (No. 20132BAB201044) and Jiangxi Higher Technology Landing Project, China (No. KJLD12071).

References

[1] K. R. Pattipati, M. Alexandridis. Application of heuristic search and information theory to sequential fault diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, no. 4, pp. 872–887, 1990.

[2] V. Raghavan, M. Shakeri, K. R. Pattipati. Optimal and near-optimal test sequencing algorithms with realistic test models. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 29, no. 1, pp. 11–27, 1999.

[3] M. Shakeri, V. Raghavan, K. R. Pattipati, A. Patterson-Hine. Sequential testing algorithms for multiple fault diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 30, no. 1, pp. 1–14, 2000.

[4] Z. Hu, S. G. Zhang, Y. M. Yang, L. J. Song, Y. Liu. Test sequencing problem considering life cycle cost based on the tests with non-independent cost. *Chemical Engineering Transactions*, vol. 33, no. 3, pp. 253–258, 2013.

[5] F. Tu, K. R. Pattipati, S. Deb, V. N. Malepati. Computationally efficient algorithms for multiple fault diagnosis in large graph-based systems. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 33, no. 1, pp. 73–85, 2003.

[6] O. E. Kundakcioglu, T. Ünlüyurt. Bottom-up construction of minimum-cost and/or trees for sequential fault diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 37, no. 5, pp. 621–629, 2007.

[7] C. L. Dong, Q. Zhang, S. C. Geng. A modeling and probabilistic reasoning method of dynamic uncertain causality graph for industrial fault diagnosis. *International Journal of Automation and Computing*, vol. 11, no. 3, pp. 288–298, 2014.

[8] J. S. Yu, B. Xu, X. S. Li. Generation of test strategy for sequential fault diagnosis based on genetic algorithms. *Journal of System Simulation*, vol. 16, no. 4, pp. 833–836, 2004. (in Chinese)

[9] R. H. Jiang, H. J. Wang, B. Long. Applying improved AO* based on DPSO algorithm in the optimal test sequencing problem of large scale complicated electronic system. *Chinese Journal of Computers*, vol. 31, no. 10, pp. 1835–1840, 2008. (in Chinese)

[10] G. Y. Lian, K. L. Huang, J. H. Chen, F. Q. Gao. Optimization method for diagnostic sequence based on improved particle swarm optimization algorithm. *Journal of Systems Engineering and Electronics*, vol. 20, no. 4, pp. 899–995, 2009. (in Chinese)

[11] X. H. Qiu, J. Liu, X. H. Qiu. Random DBPSO algorithm application in the optimal test-sequencing problem of complicated electronic system. In *Proceedings of the 2nd International Conference on Computer and Automation Engineering*, IEEE, Singapore, vol. 1, pp. 107–111, 2010.

[12] P. R. Srivatsava, B. Mallikarjun, X. S. Yang. Optimal test sequence generation using firefly algorithm. *Swarm and Evolutionary Computation*, vol. 8, pp. 44–53, 2013.

[13] J. L. Pan, X. H. Ye, Q. Xue. A new method for sequential fault diagnosis based on Ant Algorithm. In *Proceedings of the 2nd International Symposium on Computational Intelligence and Design*, IEEE, Changsha, China, vol. 1, pp. 44–48, 2009.

[14] Z. L. Pan, L. Chen, G. Z. Zhang. Cultural algorithm for minimization of binary decision diagram and its application in crosstalk fault detection. *International Journal of Automation and Computing*, vol. 7, no. 1, pp. 70–77, 2010.

[15] C. L. Yang, J. H. Yan, B. Long, Z. Liu. A novel test optimizing algorithm for sequential fault diagnosis. *Microelectronics Journal*, vol. 45, no. 6, pp. 719–727, 2014.

[16] J. Zhang, A. C. Sanderson. JADE: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

[17] Y. Wang, Z. X. Cai, Q. F. Zhang. Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 55–66, 2011.

[18] J. Kennedy, R. C. Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, IEEE, Piscataway, USA, vol. 4, pp. 1942–1948, 1995.

[19] J. J. Liang, A. K. Qin, P. N. Suganthan, S. Baskar. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

Xiao-Hong Qiu received the B. Sc. and M. Sc. degrees in automatic control from Beijing University of Aeronautics and Astronautics, China in 1989, 1992 respectively, and the Ph. D. degree in flight control, guidance and simulation from Beijing University of Aeronautics and Astronautics, China in 1995. From 1995 to 2002, he was a senior engineer and vice general manager in the Institute of Unmanned Vehicle, Beijing University of Aeronautics and Astronautics. Since 2002, he has been a professor of Jiangxi Agricultural University, Jiangxi Normal University, China. Currently, he is a professor in Software School at Jiangxi University of Science and Technology, China. He is the author of three books, and more than 70 articles. He was a recipient of Defense Science and Technology Progress Second Award in 2001.

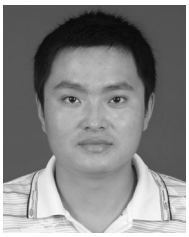
His research interests include intelligent control and intelligent computing.

E-mail: jxauqiu@163.com (Corresponding author)
ORCID iD: 0000-0003-1007-812X

Yu-Ting Hu received the B. Sc. degree in electrical engineering and automation from Jiangxi University of Science and Technology, China in 2013. She is currently a master student in the Jiangxi University of Science and Technology, China.

Her research interests include the development of software and intelligent computing.

E-mail: 1016361898@qq.com
ORCID iD: 0000-0002-5539-5340



Bo Li received the B. Sc. degree in electrical engineering and automation from Jiangxi University of Science and Technology, China in 2002, and M. Sc. degree from School of Science, Jiangxi University of Science and Technology, China in 2005. Since 2005, he is a lecturer at Software School, Jiangxi University of Science and Technology, China.

His research interests include the development of software and intelligent algorithm.

E-mail: libo.jx@163.com
ORCID iD: 0000-0002-0406-4584