Intelligent Algorithm of Optimal Allocation of Test Resource Based on Imperfect Debugging and Non-homogeneous Poisson Process

Xiong Wei 1,2, Guo Bing 3*, Shen Yan 3, Wenli Zhang 1,4

1 Computer Science College, Sichuan University, Chengdu 610064, China
2 Leshan Vocational & Technical College, Leshan 614000, China
3 School of Control Engineering, Chengdu University of Information Technology, Chengdu 610225, China
4 Chengdu Technological University, Chengdu 611730, China

Abstract

Based on non-homogeneous Poisson Process (NHPP), by a combination of imperfect debugging in the testing stage and introducing wrong model, the optimal distribution expression and constraint conditions of test resources are derived in this paper, and due to the NP-hard nature of this problem, so we introduce the intelligent algorithm. By a combination of basic intelligent algorithms of GA, SA and CS, the algorithms of GA-CSA and GA-SA are put forward in this paper, which are also compared with each other concerning the efficiency of solving the problem. In terms of optimal degree and convergence rate of solutions, the effect of GA-CSA is the best and the effect of GA-SA is the second, the effect of GA is equivalent to that of SA, the effect of the latter two algorithms is lower than that of the former two algorithms. The experiment result has proved that our optimal mathematical deduction of test resources is correct and the intelligent algorithm is effective.

Key words: NHPP, TCM, GA-CSA, GA-SA

1. INTRODUCTION

As an important safeguarding measure of reliability of software, the software test has received universal attention. How to carry out reasonable software test under limited conditions and realize the balance of software test cost and reliability is an important subject of software development. The allocation problem of software test resources refers to the fact that under the premise of limited test resources, how to allocate the test resource in a reasonable and optimal way in order to reduce the cost of software testing while the reliability of the software system is up to the standards at the same time.

Literature (Fiondella and Gokhale, 2012) applies software architecture theory to optimize allocation of test resources, in order to reduce the cost of test; literature (Wang et al., 2010) discusses multi-objective evolutionary algorithm to solve optimal allocation of test resource; literature (Huang and Lo, 2006) discusses reliability of modularized software and allocation of optimal test resource of cost; literature (Lyu et al., 2002) considers the optimal allocation of test resource under the growth model of software reliability; literature (Aggarwal et al., 2012; Berman and Culter, 2004; Dai et al., 2013; Morali and Soyer, 2003; Pietrantuono et al., 2010; Zhang and Pham, 1998) discusses the optimal resource allocation of type of software test from multiple angles and methods.

The relation between the reliability of software and the test resources can be indicated by Reliability Growth Model (RGM). The relation between cost of software test and resources can be indicated by Test Cost Model (TCM). In the following, RGM and TCM will be introduced respectively.

Over the past thirty years, various software models have been put forward (Goela and Okumoto, 1979; Lyu, 1996; Musa et al., 2010), in which G-O model and NHPP model are the most commonly used and the most well-known.
2. MATHEMATICAL MODEL

We used Non-Homogeneous Poisson Process for modeling of software reliability. According to the theories of NHPP, suppose the number of defects emerging during the time of $(0, t)$ is $N(t)$, for any $t \geq 0, x \geq 0$, we can get:

$$P\{N(t + x) - N(t) = k\} = \frac{[m(t + x) - m(t)]^k}{k!} e^{-[m(t + x) - m(t)]}$$

(1)

$m(t)$ is the mean function of NHPP. As for $N(t)$, the distribution function is:

$$P\{N(t) = n\} = \frac{[m(t)]^n}{n!} e^{-m(t)}$$

(2)

In order to build up RGM, we have made the following hypothesis:

1) For any model $i$, the wrong removing process can be indicated by NHPP with the mean value of $m_i(t)$

2) In the wrong removing phase, the wrong content would be removed with probability of $p$;

3) In the wrong removing phase, new mistakes may emerge;

Wrong remove rate is a $S$-type non-decreasing logistic function.

4) Based on the hypotheses above, for the testing of some module, the process of removing mistakes can be described with the following expression:

$$\frac{\partial m(t)}{\partial t} = p * b(S(t)) * (a(t) - m(t)) * \frac{\partial S(t)}{\partial t}$$

(3)

$S(t)$ indicates the test resources of cumulative consumption during the time of $(0, t]$, $a(t)$ and $b(t)$ can be indicated as:

$$b(t) = \frac{b}{1 + \beta e^{-\beta t}}$$

(4)

$$a(t) = a + \alpha * m(t)$$

(5)

$b$ is the wrong detection rate, $a$ is the number of mistakes existing in the system at the start of the testing of the software, $\alpha$ is the ratio of production of new mistakes, $\beta$ is a constant. (4) and (5) are put into (3), the value of $m(t)$ is:

$$m(t) = \frac{a}{1 - \alpha} \left[ 1 - \frac{(1 + \beta) e^{\beta S(t)}}{1 + \beta e^{-\beta S(t)}} \right]^{1 - \alpha}$$

(6)

Prove :

(4) and (5) are put into (3), which can be simplified into:

$$\frac{dm(t)}{dt} + \frac{(1 - \alpha) pb}{1 + \beta e^{-\beta S(t)}} * \frac{dS(t)}{dt} * m = \frac{apb}{1 + \beta e^{-\beta S(t)}} * \frac{dS(t)}{dt}$$

(7)

Let:
The following first order linear differential equation can be obtained:

\[
\frac{dm(t)}{dt} + P(t)m(t) = \frac{a}{1-\alpha} \dot{P}(t)
\]  \hspace{1cm} (8)

The general solution of the first order linear differential equation is:

\[
m(t) = e^{-\int P(t)dt} \left[ \int \frac{a}{1-\alpha} \dot{P}(t) e^{\int P(t)dt} dt + C \right] \]
\[
= a \int (1-\alpha) pb \dot{S}(t) dt + \frac{a}{1-\alpha} \int \dot{P}(t) dt + C
\]  \hspace{1cm} (9)

which can be simplified into

\[
m(t) = \frac{a}{1-\alpha} + C \cdot e^{-\int P(t)dt}
\]  \hspace{1cm} (10)

\[
\int P(t) dt = \left[ (1-\alpha) pb \dot{S}(t) + \frac{a}{1-\alpha} \int \dot{P}(t) dt \right]
\]
\[
= (1-\alpha) pb \dot{S}(t) + \frac{a}{1-\alpha} \int \dot{P}(t) dt + C
\]
\[
= (1-\alpha) pb \dot{S}(t) + \frac{a}{1-\alpha} \log \left( 1 + \beta e^{-\dot{S}(t)} \right) + C
\]
\[
\]  \hspace{1cm} (11)

is put into (10) to get:

\[
m(t) = \frac{a}{1-\alpha} + C \cdot e^{-\int [1-\alpha] pb \dot{S}(t) + \frac{a}{1-\alpha} \int \dot{P}(t) dt + C}
\]
\[
= \frac{a}{1-\alpha} + C \cdot e^{-[1-\alpha] \int pb \dot{S}(t) dt} \cdot e^{-\frac{a}{1-\alpha} \int \dot{P}(t) dt} \cdot C
\]
\[
= \frac{a}{1-\alpha} + C \cdot e^{-[1-\alpha] \int pb \dot{S}(t) dt} \cdot e^{-\frac{a}{1-\alpha} \int \dot{P}(t) dt} \cdot C
\]
\[
= \frac{a}{1-\alpha} + C \cdot \left( \frac{e^{-\dot{S}(t)}}{1 + \beta e^{-\dot{S}(t)}} \right)^{\frac{a}{1-\alpha} \int \dot{P}(t) dt}
\]
\[
\]  \hspace{1cm} (12)

As \( m(0) = S(0) = 0 \), (12) can be put into the formula to obtain the value of \( C_2 \):

\[
C_2 = -\frac{a}{1-\alpha} \log (1 + \beta)
\]  \hspace{1cm} (13)

is put into (12), finally the value of \( m(t) \) can be obtained:

\[
m(t) = \frac{a}{1-\alpha} + C_2 \cdot \left( \frac{e^{-\dot{S}(t)}}{1 + \beta e^{-\dot{S}(t)}} \right)^{\frac{a}{1-\alpha} \int \dot{P}(t) dt}
\]
\[
= \frac{a}{1-\alpha} - \frac{a}{1-\alpha} \log (1 + \beta) \left( \frac{e^{-\dot{S}(t)}}{1 + \beta e^{-\dot{S}(t)}} \right)^{\frac{a}{1-\alpha} \int \dot{P}(t) dt}
\]
\[
= \frac{a}{1-\alpha} \left[ 1 - \left( \frac{1 + \beta e^{-\dot{S}(t)}}{1 + \beta e^{-\dot{S}(t)}} \right)^{\frac{a}{1-\alpha} \int \dot{P}(t) dt} \right]
\]
The whole process of demonstration is complete.

Suppose there are N modules in the system, the total testing time is T, during the time of T, the test resources consumed by any module is $S_i$, then the formula (6) can be rewritten into the form based on the test resources, indicated as followed:

$$m_i(S_i) = \frac{a_i}{1 - a_i} \left[ 1 - \left( \frac{(1 + \beta_i) e^{\beta_i S_i}}{1 + \beta_i e^{\beta_i S_i}} \right)^{n(1 - \alpha_i)} \right]$$

(14)

In the formula, $b_i$ is the wrong detection rate of the i module, $a_i$ is the number of mistakes for the i module when the software started, $\alpha_i$ is the ratio of new mistakes in the i module, $\beta_i$ is the constant, $p_i$ is the probability of removing a mistake in the i module.

Kapur et al., (1999) put forward general TCM, which can be described in the following expression:

$$R(t) = \frac{m(t)}{a / (1 - a)}$$

(15)

With a combination of the formula (14) and (15), the reliability of any module i during the testing time T can be defined as:

$$R_i = 1 - \left( \frac{(1 + \beta_i) e^{-\beta_i S_i}}{1 + \beta_i e^{-\beta_i S_i}} \right)^{n(1 - \alpha_i)}$$

(16)

3. COST MODEL OF SOFTWARE TEST

Kapur et al., (1999) put forward general TCM, which can be described in the following expression:

$$C(T) = C_1 * m(T) + C_2 * (m(\infty) - m(T)) + C_3 * T$$

(17)

The formula (17) is changed into the model between test cost and test resources, which is indicated in the following expression:

$$C(S) = C_1 * m(S) + C_2 * (m(\infty) - m(S)) + C_3 * S$$

(18)

In the expression, S indicates the resources consumed by software test, $C(S)$ indicates the cost of software test, the first part indicates the cost of looking for and revising mistakes in the software test, the second part indicates the cost of looking for and revising mistakes after the software is published, the last part indicates the other cost before the software is published, such as the payment of the testing personnel. $C_1$ indicates the cost of looking for and revising a software mistake during the testing period, $C_2$ indicates the cost of looking for and revising a software mistake during the operation of the software, $C_3$ indicates other cost of resources consumed in unit testing.

In the software testing, every module is tested independently, so the total cost of software system is the sum of test cost of every module, which can be indicated as followed:

$$C = \sum_{i=1}^{N} C_i(S_i)$$

(19)

During the period of software test, every module of the software is tested independently, and the importance of every module is different, suppose $w_i$ is the weight factor of the i model, the formula (14) and (18) are put into the formula (19), we can get the total cost of the test of the system:
In the expression, $C_{1i}$ and $C_{2i}$ represent $C_1$ and $C_2$ in the corresponding formula (1-11) of the i module.

4. OPTIMIZING THE OBJECT

In the optimal allocation of the test resources of the software, our optimal objective is to minimize the test cost, independent variable is the test resource consumed by every model. The optimal objective can be indicated as:

Minimize:

$$C = \sum_{i=1}^{N} \left( C_{i1} - C_{i2} \right) \frac{w_i}{1-\alpha_i} \left[ 1 - \left( \frac{(1+\beta_i e^{-\lambda_i S_i})}{1+\beta_i e^{-\lambda_i S_i}} \right)^{\gamma_i(\alpha_i)} \right] + \sum_{i=1}^{N} C_{i2} \frac{\alpha_i}{1-\alpha_i} + C_3 \sum_{i=1}^{N} S_i$$ (20)

5. CONSTRAINT PROGRAMMING

Suppose the cost sum of test resource of every model is $S_{ce}$, the sum of allocated test resource of every module does not exceed some fixed value $U_{Sce}$, so the constraint programming of the test resources can be indicated as:

$$S_{ce} = \sum_{i=1}^{N} S_i \leq U_{Sce}$$ (22)

The reliability of every module is not less than some fixed value $L_{Res}$, then the reliable constraint can be indicated as:

$$R_i = 1 - \left( \frac{(1+\beta_i e^{-\lambda_i S_i})}{1+\beta_i e^{-\lambda_i S_i}} \right)^{\gamma_i(\alpha_i)} \geq L_{Res}$$ (23)

6. SYSTEM OPTIMIZATION ALGORITHM

The whole optimization issue is a combination optimization issue, which is also a Non-deterministic Polynomial issue of difficulty. Therefore, we need a heuristic searching method to achieve the optimal answer of issues.

6.1 The system optimization algorithm based on GA and CSA

In the heuristic algorithm, Genetic Algorithm (GA) has possessed strong and global searching ability, which is often used to solve NP issue of difficulty, but the local searching ability is relatively poor. On the contrary, CUCKOO Search Algorithm (CSA) (Yang and Deb, 2009; Yang and Deb, 2010; Yang and Deb, 2011) has possessed good local searching ability, but its global search ability is relatively poor. Therefore, we considered combining GA with CSA, we put Levy flight of CSA into the use of GA to improve its local search ability. We named the improved optimized algorithm as GA-CSA, we used $S=(S_1, S_2, \ldots, S_N)$ to encode individuals of a group. Suppose $p_i$ indicates the i individual in the generation of t, the corresponding $p_{ij}$ indicates the j component, then the method of updating individuals by Levy flight (Pavlyukevich, 2007) is as followed:

$$p_{ij}^{t+1} = p_{ij}^t + s \cdot \left( p_{ij}^t - gbest \right)$$ (24)

In the expression, gbest indicates the global optimized status of the current population, s indicates the step length of Levy flight: $s=\alpha \cdot \mu / |v|^{1/\beta}$, the parameter $\mu \sim N(0, \sigma^2)$, $v \sim N(0, I)$, among which:
\[
\tilde{\sigma}_{\mu}^2 = \left( \frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\Gamma(1+\beta/2)\left[2^{\beta-1}\beta\right]} \right)^{1/\beta} 
\]

(25)

Generally we set \( \alpha=0.01, \beta=1.5 \). Detailed procedure of GA-CSA algorithm is as followed:

Step 1 Initialized population \( P_0 \), the number of iterations maxgen, the scale of population size pop;

Step 2 The historical optimal of initialized individuals pbest, the globally optimal of the population gbest;

Step 3 Initialization \( t=0 \);

Step 4 If \( t=\text{maxgen} \), stop the implementation of procedures;

Step 5 Carry out the selection, cross-operation and mutation of, producing the offspring;

Step 6 As for \( \forall p \in P_{t+1} \), update individual \( p \) according to Levy flight;

Step 7 Judge whether \( p \) is more optimal than pbest, if it is yes, then update pbest;

Step 8 Judge whether pbest is more optimal than gbest, if it is yes, then update gbest;

Step 9 Update current number of iterations: \( t=t+1 \).

6.2 System optimization algorithm based on GA and SA

Besides the mentioned fact that CSA has very good local search ability, in the heuristic algorithm, Simulated Annealing (SA) also has very good local search ability, but global search ability is poor, and the convergence rate is slow. In this section, we considered combining GA with SA to form new optimization algorithm. In the following sections, we will make a comparison with the former GA-CSA through the experiment simulation.

We introduced Boltzmann mechanism of SA into GA to improve the local search ability of GA and named it as GA-SA. We used \( S=(S_1, S_2, ..., S_N) \) to encode the individuals of population, \( C(S) \) indicates the corresponding objective function, detailed procedure of GA-SA is as followed:

Step 1 Initialized starting temperature \( T=T_0 \), the termination temperature \( T_{\text{end}} \), the cooling coefficient \( \mu \), the number of internal recycle Loop;

Step 2 Initialized parent population \( P \), the historical optimal of individuals \( p_{\text{best}} \), the globally optimal of population \( g_{\text{best}} \);

Step 3 If \( T<T_{\text{end}} \), the algorithm is terminated;

Step 4 Initialization \( i=1 \);

Step 5 If \( i>\text{Loop} \), then \( T=\mu*T \), turn to Step 3;

Step 6 Initialized offspring population \( Q=\Phi \);

Step 7 Carry out the selection, overlapping and mutation of the population \( P \), producing new population \( P_{\text{new}} \);

Step 8 For each individual \( P_{\text{new}} \in P_{\text{new}} \), \( P \in P \) in \( P_{\text{new}} \) and \( P \), adopt Boltzmann mechanism of SA, judge the individuals needed to be stored \( P_{\text{save}}=\text{Boltzmann}(P_{\text{new}}, P) \), which should be stored into the offspring population \( Q \);

Step 9 Judge whether \( P_{\text{save}} \) is more optimal than pbest, if it is yes, then update pbest;
Step 10 Judge whether pbest is more optimal than gbest, if it is yes, then update gbest;

Step 11 Carry out i = i + 1, turn to the procedure Step 5.

In Step 8 of GA-SA, judge whether we should store $P_{new}$ or $P$, Boltzmann mechanism is introduced, the detailed realization method is indicated as followed:

$$P_{save} = \text{Boltzmann}(P_{new}, P)$$

1. If $P_{new}$ meets the constraint conditions, and does not satisfy the constraints then

2. $P_{save} = P_{new}$

3. else if $P_{new}$ does not meet the conditions of constraint then

4. $P_{save} = P$

5. else

6. $\Delta \text{Obj} = \text{Obj}(P_{new}) - \text{Obj}(P)$

7. if $\Delta \text{Obj} \leq 0$ then

8. $P_{save} = P_{new}$

9. else

10. $Pr = \exp(-\Delta \text{Obj}/T)$, $r = \text{Random}[0, 1]$

11. if $Pr > r$ then $P_{save} = P_{new}$ end if

12. end if

13. end if

14. }

7. EXPERIMENT SIMULATION

Through the effectiveness and robustness of verification algorithm GA-CSA and GA-SA in the experiment simulation system, we selected GA and SA as comparative algorithms. The comparison of algorithm performance is mainly made by the advantage and disadvantage of test cost of the software.

In the experiment, 6 different test examples are selected in total, their names are EX1, EX2, EX3, EX4, EX5 and EX6 respectively; the corresponding number of test model is 10, 12, 14, 16, 18 and 20. The parameter settings of EX1, EX2, EX3, EX4, EX5 and EX6 are as indicated in the table 1.

<table>
<thead>
<tr>
<th>Test case</th>
<th>EX1</th>
<th>EX2</th>
<th>EX3</th>
<th>EX4</th>
<th>EX5</th>
<th>EX6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of modules</td>
<td>10</td>
<td>12</td>
<td>14</td>
<td>16</td>
<td>18</td>
<td>20</td>
</tr>
<tr>
<td>Test resource constraints ($U_{Sce}$)</td>
<td>5000</td>
<td>6000</td>
<td>7000</td>
<td>8000</td>
<td>9000</td>
<td>10000</td>
</tr>
<tr>
<td>Reliability constraint ($L_{Rea}$)</td>
<td>0.93</td>
<td>0.93</td>
<td>0.93</td>
<td>0.93</td>
<td>0.93</td>
<td>0.93</td>
</tr>
<tr>
<td>Test phase correction cost ($C_1$)</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Runtime modification cost ($C_2$)</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>
7.1 Algorithm parameter setting

The parameter setting of GA-CSA algorithm is as followed in the table 2.

<table>
<thead>
<tr>
<th>Parameters Of GA-CSA</th>
<th>0.5</th>
<th>0.5</th>
<th>0.5</th>
<th>0.5</th>
<th>0.5</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size (n)</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Iteration number (G)</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Crossover-probability (PC)</td>
<td>0.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mutation-probability (PM)</td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>alpha (α)</td>
<td>0.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>beta (β)</td>
<td>1.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The parameter setting of an algorithm is indicated as followed in the table 3:

<table>
<thead>
<tr>
<th>Parameters Of GA-SA</th>
<th>0.5</th>
<th>0.5</th>
<th>0.5</th>
<th>0.5</th>
<th>0.5</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size (n)</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initial temperature (T0)</td>
<td>500</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Termination temperature (Tend)</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cooling coefficient (MU)</td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internal-cycle-times (Loop)</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Crossover-probability (PC)</td>
<td>0.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mutation-probability (PM)</td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Population size (n)</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The parameter setting of GA, SA algorithm is as followed in the table 4, table 5:

<table>
<thead>
<tr>
<th>Parameters Of GA</th>
<th>0.5</th>
<th>0.5</th>
<th>0.5</th>
<th>0.5</th>
<th>0.5</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crossover-probability (PC)</td>
<td>0.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mutation-probability (PM)</td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Population size (n)</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Iteration number (G)</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameters Of SA</th>
<th>0.5</th>
<th>0.5</th>
<th>0.5</th>
<th>0.5</th>
<th>0.5</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial temperature (T0)</td>
<td>1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Termination temperature (Tend)</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cooling coefficient (μ)</td>
<td>0.95</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internal-cycle-times (Loop)</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7.2 Performance comparison

The experiment is fulfilled in the encoding environment of AMD Athlon(tm) II P340 Dual-Core 2.20GHZ CPU, 4GB RAM, Matlab R2008a. In order to reduce the effect of the random data over the experiment results, the algorithm is run independently for ten times in 6 test examples, then the average value of the test cost C is calculated, the mean value replacement algorithm is used in every test example to obtain the minimal test cost.
Figure 1 is the minimal test cost solved through the algorithm in 6 test examples. From the figure, it can be seen that the minimal test cost solved by GA-SA is smaller than that by GA-CSA, GA and SA, which indicates that the test cost solved by GA-SA is optimal, while the test cost solved by GA-CSA is smaller than that by GA and SA, which indicates that the test cost solved by GA-CSA is more optimal than that by GA and SA, the test costs of GA and SA are very close.

![Figure 1. Algorithm for the minimum test cost](image)

Table 6 show the solutions in detail, the average test cost solved by GA-CSA, GA-SA, GA and SA is 12946, 12268, 14915 and 14718 respectively, the solution of GA-SA is the minimal, in terms of the optimization result: GA-SA > GA-CSA > GA >SA.

<table>
<thead>
<tr>
<th>algorithm</th>
<th>EX1</th>
<th>EX2</th>
<th>EX3</th>
<th>EX4</th>
<th>EX5</th>
<th>EX6</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA-CSA</td>
<td>7861</td>
<td>9529</td>
<td>11786</td>
<td>14058</td>
<td>16331</td>
<td>18112</td>
</tr>
<tr>
<td>GA-SA</td>
<td>7402</td>
<td>8832</td>
<td>10759</td>
<td>12798</td>
<td>15993</td>
<td>17825</td>
</tr>
<tr>
<td>GA</td>
<td>9507</td>
<td>11382</td>
<td>13745</td>
<td>16182</td>
<td>18409</td>
<td>20265</td>
</tr>
<tr>
<td>SA</td>
<td>9257</td>
<td>11178</td>
<td>13571</td>
<td>16009</td>
<td>18213</td>
<td>20078</td>
</tr>
</tbody>
</table>

8. CONCLUSION

Based on NHPP, this paper derives the optimal allocation program under limitation of test resources from the combination of imperfect debugging, introducing mistakes in debugging and the hypothesis that the wrong remove rate is S-type non-decreasing logic function; on this basis, apply GA-CSA, GA-SA, GA and SA algorithms for solutions, and make comparison of four intelligent algorithms. Among the four algorithms, the solution of GA-SA is the best, the solution of GA-CSA is the second best, the solution of GA and the solution of SA are equivalent. The experiment result has shown that the assumptions, mathematical derivation and intelligent algorithm are reasonable.

ACKNOWLEDGMENTS

This work was supported in part by the State Key Program of National Natural Science Foundation of China under Grant No.61332001; The National Natural Science Foundation of China under Grant No. 61272104 and 61472050; the Science and Technology Planning Project of Sichuan Province under Grant No. 2014JY0257, 2015GZ0103 and 2014-HM01-00326-SF.

REFERENCES