An Effective Test Data Generation Method based on Improved Particle Swarm Optimization Algorithm

Xiaojiang Chen, Lei Wang, Jijun Zeng, Daohuan Jiang and Dexin Huang

ABSTRACT

A suitable set of test data can improve test efficiency and automatic level in automated testing. This paper presents an automatic generation method of test data based on improved particle swarm optimization algorithm, which can significantly improve the efficiency of automated testing.

KEYWORDS

Automated Testing, Particle Swarm Optimization, Test Data Generation

INTRODUCTION

Statistics show that software testing in the entire software life cycle occupies an important position, accounting for more than 60% of the entire development costs. Thus, software testing is the need to spend huge manpower and resources and time, so it is very important to improve the degree of automation of software testing. Among them, to improve the degree of automation generated test data is to improve the degree of automation software testing the key.
In this paper, an automatic generation algorithm of test data based on improved particle algorithm is proposed, which improves the degree of automation and efficiency of software testing.

**PARTICLE SWARM ALGORITHM BASIC PRINCIPLE AND ITS IMPROVEMENT**

Particle swarm optimization algorithm, the most basic idea is through the group of individuals between the coordination and information sharing to find the optimal solution. Suppose you think of each search problem as a bird, called "particle". In a D-dimensional space inside the initialization of a group of m memory function of the particles, where the position of the i-th particle \( x_i = \{ x_{i1}, x_{i2}, ..., x_{iD} \} \), the velocity \( v_i = \{ v_{i1}, v_{i2}, ..., v_{iD} \} \), the individual has experienced the best position \( p_{bi} = \{ p_{i1}, p_{i2}, ..., p_{iD} \} \), the population has experienced the best position \( g_{bi} = \{ g_{i1}, g_{i2}, ..., g_{iD} \} \); all particles are determined by a fitness function to determine the fitness value its current position is good or bad; the direction of movement of each particle according to \( p_{bi} \) and \( g_{bi} \) to adjust the particles. Therefore, the particles in the group are affected by other excellent particles in the population. The particle velocity position changes as follows:

\[
\begin{align*}
    v_{id}^{(t+1)} &= v_{id}^{(t)} + \beta_1 (p_{bi} - x_{id}) + \beta_2 (g_{bi} - x_{id}) \quad (1) \\
    x_{id}^{(t+1)} &= x_{id}^{(t)} + v_{id}^{(t+1)} \quad (2)
\end{align*}
\]

\( \beta_1, \beta_2 \) that \( r_1c_1, r_2c_2, r_1, r_2 \) belongs to \([0,1]\) between the random number; \( c_1, c_2 \) is a non-negative constant, also known as learning factors, where the default is 2. \( t \) represents the number of iterations. The above particle optimization method, although the convergence speed is fast, but easy to fall into the local optimum. In order to solve this problem, two particles adjacent to the particle can be included in the factors that affect the particle. Assuming that the two adjacent particles in the particle field are k, l, the best positions of their individual are pbk, pbl, then:

\[
\begin{align*}
    v_{id}^{(t+1)} &= v_{id}^{(t)} + \beta_1 (p_{bi} - x_{id}) + \beta_2 (g_{bi} - x_{id}) + \beta_3 [ \gamma (p_{bkid} - x_{id}) + (1- \gamma)(p_{blid} - x_{id}) ] \quad (3) \\
    x_{id}^{(t+1)} &= x_{id}^{(t)} + v_{id}^{(t+1)} \quad (4)
\end{align*}
\]

\( \beta_3 \) is the same meaning of \( \beta_1, \beta_2 \). \( \gamma \) is the number between \([0,1]\). When \( pbk > pbl \), much larger, then the greater the value of \( \gamma \), otherwise the smaller the value of \( \gamma \). The improved particle algorithm makes the particle not only learn from the best particles of the global experience, but also learn from the experience of the two particles, so that the information between the particles get better sharing, to avoid
the particle swarm algorithm into the local optimal situation, while also to maintain a good convergence rate.

AUTOMATICALLY GENERATE TEST DATA WITH AN IMPROVED PARTICLE ALGORITHM

Construct the Fitness Function

Here is the "branch function superposition method" to construct the fitness function, the basic idea of branching the superposition function is to first determine the target path, and then insert the branch function \( f \) in the form of the pile before the branch point of the path, and finally the Branch function values are added to get the fitness function.

Assuming that there are \( m \) branch points and \( n \) input parameters on the existing path \( p \), the expression of the branch function is:

\[
\begin{align*}
f_1 &= f_1 (x_1, x_2 \ldots, x_n) \\
f_2 &= f_2 (x_1, x_2 \ldots, x_n) \\
& \quad \ldots \\
f_m &= f_m (x_1, x_2 \ldots, x_n)
\end{align*}
\]

Thus, the fitness function \( F \) is obtained as follows:

\[
F = f_1 + f_2 + \ldots + f_m
\]

among them, \( f(x) = \begin{cases} 0 & x \leq 0; \\ x & x > 0; \end{cases} \)

Test Data Generation Step

(1) initialize, analyze the measured program, insert the branch function in the test path, calculate the fitness value. The number of selected population particles \( m \), the maximum number of iterations allowed \( T_{max} \), the fitness threshold is \( \varepsilon \), the position \( X \) of the initialized particle and the velocity \( V \) are the random numbers between (0,100) and (0,2), respectively.

(2) the number of iterations \( t = 0, F_g = 0, P_g = (0,0 \ldots 0) \)

(3) While \( (F_g \leq \varepsilon \& \ t < T_{max}) \)

(4) For\((i=0; i < m; i++)\{
\begin{align*}
&\text{if}(F_i > F_g(i)) \{ F_g(i) = F_i; P_i = X_i; \} \\
&\text{if}(F_i < F_g) \{ F_g = F_i; P_g = X_i; \} \\
&\text{if}(F_i < F_k(i) \& \& F_i > F_l(i)) \{ F_i = F_k(i); X_i = P_k; F_l(i) = F_i; X_l = P_l; \gamma = 0.9; \} 
\end{align*}
\)
if(F_i>F_k(i) && F_i>F_l(i)) {F_k(i) = F_i; P_k = X_i; F_l(i) = F_i; P_l = X_i; γ = 0.5;}
if(F_i>F_k(i) && F_i<F_l(i)) {F_k(i) = F_l(i); P_k = P_l; F_l(i) = F_i; P_l = X_i; γ = 0.1;}
if(F_i<F_k(i) && F_i<F_l(i)) {if(F_k(i)>F_l(i)) {F_i = F_k(i); X_i = P_k; F_l(i) = F_k(i); P_l = P_k; γ = 0.9;}
else {F_i = F_l(i); X_i = P_l; F_k(i) = F_l(i); P_k = P_l; γ = 0.1}}
}
(5) for(i=0; i<m; i++){
V_{i}^{(t+1)}=V_{i}^{(t)}\beta 1(P_i-X_i)+\beta 2(F_k-X_i)+\beta 3[\gamma(P_k-X_i)+(1-\gamma)(P_l-X_i)];
X_{i}^{(t+1)}=X_{i}^{(t)}+V_{i}^{(t+1)};
}
(6) t=t+1;

When the number of iterations reaches the highest value or the fitness reaches a stable value, the algorithm stops, and the optimal solution of the fitness function is the optimal solution of the optimal search.

ANALYSIS OF EXPERIMENTAL RESULTS

Select the software used in the test to generate equilateral triangular data as a test, analysis of Genetic Algorithm (GA), Artificial Immune Algorithm (AIA) and Improved Particle Swarm Optimization (IPSO) to Find the Optimal Number and Time of the Optimal Solution. The initial particle group m = 200, the results shown in Figure 1 below:

![Experimental results](image)

The results show that the improved particle swarm algorithm has the least time and the number of iterations in finding the optimal solution.

CONCLUSIONS

Automatic generation of test data is a key problem in software testing, improving the generation of test data can improve the degree of software testing automation and test efficiency. Based on the particle swarm algorithm to generate the test data, this paper proposes an improvement of the particle swarm optimization
algorithm, which avoids the local optimization of the particle swarm algorithm and makes the test data more efficiently.

REFERENCES