Software Test Case Optimization Method based on Multi-Objective Particle Swarm Optimization

He Dandan
Dalian Institute of Science and Technology
Dalian Liaoning 116052, China
hedandan98@163.com

Abstract — As the computer technology improves rapidly, the scale of software has increased greatly, which makes it more and more difficult to find a bug in software. As a result, the enhancement of software quality and reliability has become an important task in the field of software engineering. Test is an important step that guarantees software quality and reliability. We put forward a novel multi-objective optimization scheme based on improved multi-objective particle swarm optimization. Then software test case optimization method is investigated based on proposed multi-objective particle swarm optimization. The experiment results show that the proposed software test case optimization method has high efficiency.

Keywords - Multi-objective optimization; Particle swarm optimization; Software test case

I. INTRODUCTION

Along with the rapid development of information technology in social production application, the role of computer software is becoming more and more important in information system, which also becomes complex and huge [1]. In some critical applications, how to guarantee the reliability of software has become a serious issue. Software testing is the important means to ensure software quality and reliability and test case generation is the core of the software testing. In traditional methods, generating test cases manually is an extremely complicated and tedious manual labor for software testers, which not only has high cost, but also is error-prone [2,3]. Automation testing technology drives the execution of the program through the automatic generation of test cases, which effectively guarantees the efficiency and effectiveness and controls the testing cost. Test generation technology based on searching technology is newly developed software testing technology emerging in recent years, which transforms the test cases into a search process, and has made great progress in software production.

Evolutionary algorithm is a kind of random search algorithm simulating biological natural selection and evolution, which is suitable for solving highly complex nonlinear problems and has got the very extensive application, at the same time it has good generality [4-6]. In target oriented evolution testing, since there is only a single target, advantages of evolution testing were fully shown. Optimization problem of the real world, however, often has multiple attributes, in general we need to optimize multiple targets at the same time. And in most cases, at the same time, optimization targets are conflicting with each other, such as in the production activities in enterprises, product quality and production cost are two conflicting goals. To achieve the goal of total optimization, we usually need to comprehensively consider conflicting sub-goals, namely the compromise on each target. Thus, in view of the multi-objective optimization problem, multi-objective evolutionary algorithm turned up [7, 8].

Software engineering needs to improve software productivity, so the software testing multi-objective optimization is put forward. It is difficult to solve optimization problems with the discontinuous, nonlinear and other complex characteristics by traditional optimization technology [9, 10]. Evolutionary algorithm provides a new way to solve multi-objective optimization problems and multi-objective evolutionary algorithm has been one of the focus areas in nearly 20 years [11, 12]. With the development of evolutionary algorithms, many classic algorithms have been proposed. Among them, particle swarm optimization (PSO) has always been one of the hot topics in recent year owing to its advantages, such as simple concept and easy combination, high efficiency, etc. And PSO has been expanded for solving multi-objective optimization problems.

Particle swarm optimization (PSO) has a few parameters and fast convergence and is easy to implement, so it has been used for test case generation technology. Multi-objective particle swarm algorithm based on Pareto optimization concept develops very fast. Deming Lei [13] proposed a kind of multi-objective PASO to solve multi-objective shop scheduling problem. N.C. Sahoo [14] proposed fuzzy Pareto driver model based on MOPSO, which corresponds to electric power distribution system planning. A novel particle swarm algorithm based on Pareto optimization was proposed [15], which was improved.
HE DANDAN: SOFTWARE TEST CASE OPTIMIZATION METHOD BASED ON IMPROVED MULTI-OBJECTIVE PARTICLE SWARM

II. MATHEMATICAL DESCRIPTION OF MULTI-OBJECTIVE PROBLEM

A single-objective optimization problem can be expressed as

\[
\min y = f_i(x), \quad i = 1, 2, \ldots, n
\]  

(1)

where \( x \in R^n \) represents decision variable of \( m \) dimension, \( f_i(x) \) represents \( n \) number of objective function and \( S = \{ x \in R^n | i = 1, 2, \ldots, n, x \geq 0 \} \) represents feasible domain. A multi-objective optimization problem composed of \( n \) number of optimization objective can be expressed as

\[
\min f(x) = \{ f_1(x), f_2(x), \ldots, f_n(x) \},
\]

\( x \in S \subset R^n \)  

(2)

\( S \subset R^n \) represents feasible domain of \( m \) number of decision variable. \( E = \{ f(x) | x \in R^n \} \) represents objective solution vector space. Any decision variable \( x \) meeting optimization objective is called feasible solution and \( x = (x_1, x_2, \ldots, x_m) \in S \) is called feasible solution set.

Definition 1, Pareto dominate. One vector \( u = (u_1, u_2, \ldots, u_m) \) dominates vector \( v = (v_1, v_2, \ldots, v_m) \), if and only if \( \forall i \in (1, 2, \ldots, m) \), \( u_i \leq v_i \land \exists i \in (1, 2, \ldots, m), u_i < v_i \) is set up.

Definition 2, Pareto optimum. \( x^* \in S \), there is no solution \( x \) superior than \( x^* \), \( x^* \) is called Pareto optimal solution of feasible solution set \( S \).

\[ P_f = \{ X = S | \exists X' \in S \forall F(X') \leq F(X) \} \]

is called Pareto optimal solution set, which is composed of all \( x^* \) meeting Pareto optimal solution. Objective function value domain corresponding to Pareto optimal solution is called Pareto front end.

\[ P_e = \{ F(X) = (f_1(x), f_2(x), \ldots, f_n(x)) | x \in P_f \} \]

When solving constrained multi-objective optimization problems, we should not only consider the fitness of individuals, but also consider the constraints. Because of the presence of constraints, the searching space of constrained optimization problem is composed of feasible domain and the infeasible domain, which causes discontinuity of the search space, especially when the proportion of the feasible region is small and the global optimal solution is at the feasible region boundary, increasing the difficulty to search the global optimal solution. A multi-objective constraint optimization problem that contains \( m \) number of decision variable, \( k \) number of objective function and \( n \) number of constraints can be expressed as

\[
\min : f(x) = (f_1(x), f_2(x), \ldots, f_n(x)), \quad x \in S
\]  

(3)

s.t. \( g(x) = (g_1(x), g_2(x), \ldots, g_m(x)) \leq 0 \)

III. SOFTWARE TEST CASE OPTIMIZATION BASED ON IMPROVED MULTI-OBJECTIVE PARTICLE SWARM

The key to deal with constraint optimization problem is to find optimal solution in the feasible domain. The distance between particle and feasible domain is taken as measurement of individual violating constraint degree, which is labelled as penalty function value. When the particle is not in feasible domain, closeness degree between particle and feasible domain boundary is used to describe constraint violating degree. Constraint conditions generally have two forms of equality and inequality as shown in (4).

\[
\begin{align*}
\text{s.t.} & \quad f_1(x) \leq 0, \quad i = 1, 2, \ldots, a \\
& \quad f_j(x) = 0, \quad j = 1, 2, \ldots, b
\end{align*}
\]

(4)

Constraint violating condition of particle \( x_i \) violating the k-th constraint condition is defined as

\[
G_k(x_i) = \begin{cases} 
\max \{ f_i(x), 0 \}, & i = 1, 2, \ldots, a \\
\max \{ |f_i(x)| - \beta, 0 \}, & j = 1, 2, \ldots, b
\end{cases}
\]

(5)

\( \beta \) represents tolerance degree of equality constraint. The total constraint violating degree is

\[
y(x_i) = \frac{1}{m} \sum_{j=1}^{m} G_k(x_i).
\]

(6)

For infeasible individual that violates constraint condition, we define the following penalty function.

\[
Y_i(x) = (1 - \alpha)y_i(x)
\]

(7)

\( \alpha \) represents the proportion of infeasible individual to swarm scale. The objective function is

\[
\phi_i(x) = f(x) + Y_i.
\]

(8)

Suppose \( X_i = (x_{i1}, x_{i2}, \ldots, x_{id}) \) represents the i-th particle in the swarm. Optimal position of this particle in the searching space is \( P_{i} = (p_{i1}, p_{i2}, \ldots, p_{id}) \) labeled as \( p_{best} \).

In this swarm, the optimal position is labeled as \( g_{best} \).

\[
V = (v_{i1}, v_{i2}, \ldots, v_{id})
\]

represents the speed of the i-th particle. For each generation of particle, particle value changes according to formula 9 and 10.

\[
v^{r+1}_{id} = w \cdot v^{r}_{id} + c_1 \cdot r_1 \cdot (p_{id} - x^{r}_{id}) + c_2 \cdot r_2 \cdot (p_{id} - x^{r}_{id})
\]

(9)

\[
x^{r+1}_{id} = x^{r}_{id} + v^{r+1}_{id}
\]

(10)

d = 1, 2, \ldots, D represents the dimension of searching space.
\( i = 1, 2, \ldots, N \) represents the number of particles. \( t \) represents the number of iteration. \( v_{id} \) represents speed value of the \( i \)-th particle of the \( d \)-th dimension. \( x_{id} \) represents position value of the \( i \)-th particle of the \( d \)-th dimension. \( p_{id} \) represents the optimal position of the \( i \)-th particle of the \( d \)-th dimension. \( p'_{id} \) represents the optimal position of the \( i \)-th particle of the \( d \)-th dimension. \( p_{id} \) is saved. If \( x_{id} \) dominates \( x_{id} \), fitness value of the swarm is saved. If \( x_{id} \) is compared with other individuals, individual serial number is labelled as local optimal individual. The individual and history optimal position is as follows. Judge optimal value and global optimal value of the individual. Chosen of individual optimal position is as follows.

**Step1.** Initialize swarm poison and speed, set evolution iteration, swarm scale.

**Step2.** According to fitness function \( \phi (x) \), fitness value of each particle is evaluated.

**Step3.** Call the following algorithm to order individual based on non-dominated sorting and density distance. For the individual of the same dominant level, density distance sorting is carried out. At the same time, individual in the optimal non-dominated solution set is ordered according to density distance, which makes Pareto frontier distribution evenly dispersed. The typical non-dominated classification theory is used to divide the whole particle swarm. Firstly, particle swarm is initialized and fitness value of the swarm is updated. Then the particle is ordered based on non-sorting relation to choose the first non-dominated solution set labeled as 1. Then the second non-dominated solution set is generated based on the first non-dominated solution set. The rest can be done in the same manner until each particle finds its own non-dominated solution set. Each non-dominated solution set corresponds to a dominance level. The non-dominated solution set with the highest dominance level is called Pareto frontier. For the particle of the same dominance level, the particle is ordered based on density distance. The particle owning the largest density distance is the best. The non-dominated solution set sorting strategy is as follows. \( \hat{S} p \) contains individual sequence number dominated by \( p \). \( np \) represents the number of individual that dominates \( p \). Individual \( p \) compares its level with other individual \( q \) in the non-dominated solution set \( p \). If \( p \) dominates \( q \), the serial number of other level individual dominated by \( p \) is saved. If \( q \) dominates \( p \), \( np = np + 1 \) and the level of \( p \) is saved. After comparing with other individuals, individual serial number dominated by \( p \) is saved.

**Step4.** Call the following algorithm to choose the local optimal value and global optimal value of the individual. Chosen of individual optimal position is as follows. Judge each individual \( i \), particle position of the first generation is selected as local optimum. The individual and history optimal position of the individual is placed together. If level of current and history optimal objective function is not the same, the higher level is saved as local optimal value and the distance is updated. If the level is the same, density distance between the individual is compared and the individual with the higher density distance is saved as local optimal individual.

The chosen of global optimal position is as follows. Firstly, level is compared and label level, objective function and maximum distance of the highest level is saved. Then the global optimal objective function and objective function with the highest level at present is placed together. Objective function level of global optimum and current optimal individual is deleted. Delete the individual, the level of which is not high. Find the individual with the maximum density distance from the rest frontier individuals and its serial number is labelled as global optimal individual serial number.

**Step5.** Update particle value, according to formula 9 and 10.

**Step6.** Evaluate current value of particle according to fitness function. If local optimal value of particle is not improved in \( M \) consecutive generation, the speed and position of the particle is randomly assigned again. Carry out non-dominated optimal sorting and adjust individual of non-dominated optimal set according to density distance. Then choose individual and global extreme value.

**Step7.** If it meets iteration condition, the algorithm stops and outputs Pareto optimal frontier. If it does not achieve optimum, turn to step 5. The distance operator is

\[
d_{i} = \sum_{j=1}^{n} (f_{j}(x_{i}) - f_{j}(x_{j}))^2
\]

Distance matrix between \( n \) number of particles is

\[
D = \begin{bmatrix}
d_{11} & d_{12} & \ldots & d_{1n} \\
d_{21} & d_{22} & \ldots & d_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
d_{n1} & d_{n2} & \ldots & d_{nn}
\end{bmatrix}
\]

Suppose set \( S \) is made up of a series of individual. In the objective space, calculate the Euclidean distance between individual \( i \) and other individuals and the results are ordered from small to large. If \( d_{i1} \) and \( d_{i2} \) represents the two smallest distance, density distance of individual \( i \) in set \( S \) can be expressed as

\[
C_{i} = (d_{i1}^2 + d_{i2}^2) / 2.
\]

**IV. EXPERIMENT AND ANALYSIS**

In order to test the performance of proposed algorithm, some standard testing function is selected. The maximum iteration number is 150, the swarm scale is 200, learning factor is 2 and inertia weight is 0.5. Each testing function is simulated 20 times. SCH function and its variable range is \( f_{1}(x) = x^2 \), \( f_{2}(x) = (x-2)^2 \), \(-10 \leq x \leq 10\).

DOI 10.5013/IJSSST.a.16.5A.12
Constraint of SCH is \( C(x) = x^2 - 2.5x + 1.5 \geq 0 \).

BNH function and its variable range is:
\[
f_1(x) = 4x_1^2 + 4x_2^2, \quad f_2(x) = (x_1 - 5)^2 + (x_2 - 5)^2, \quad 0 \leq x_1 \leq 5, \quad 0 \leq x_2 \leq 3.
\]

Constraint of BNH is \( 0 \leq x_1 \leq 5, \quad 0 \leq x_2 \leq 3 \). DBE function and its variable range is:
\[
f(x) = x_1^2 - x_2^2 - 105, \quad 105 \leq x \leq 203, \quad 203 \leq x \leq 214.
\]

Constraint of DBE function is:
\[
C_1(x) = 9x_1 + x_2 \geq 6, \quad C_2(x) = 9x_1 - x_2 \geq 1.
\]

TNK function and its variable range is:
\[
f(x) = x_1^2 - x_2^2 + 1 + 0.1 \cos(16 \arctan(x_1 / x_2)) \leq 0.
\]

Pareto frontier of four functions is shown in figure 1(a) to figure 1(d). It can be seen that the improved multi-objective particle swarm optimization algorithm can search optimal frontier solution set, distribution is uniform and continuous and it has non-convergent solutions. In this case, we consider coverage and the number of cases. Space, schedule and printtokens program is used to test the performance of proposed scheme. Test results of three cases are shown in figure 2. It can be seen that proposed scheme has good effect. Test statistical data is shown in table 1. \( n \) represents the average size of case set, \( \text{cov} \) represents average coverage degree and \( \text{best} \) represents the number of case with the maximum coverage degree and its coverage degree. \( t \) represents time, the unit of which is minute. It can be concluded that the proposed software test case optimization method has high efficiency.

![Figure 1](image_url) (a) Pareto frontier of SCH test function

![Figure 1](image_url) (b) Pareto frontier of BNH test function

![Figure 1](image_url) (c) Pareto frontier of DBE test function

![Figure 1](image_url) (d) Pareto frontier of TNK test function

<table>
<thead>
<tr>
<th>program</th>
<th>( n )</th>
<th>( \text{cov} )</th>
<th>\text{best}</th>
<th>( t )</th>
</tr>
</thead>
<tbody>
<tr>
<td>schedule</td>
<td>8.6</td>
<td>0.97</td>
<td>8/1</td>
<td>0.58</td>
</tr>
<tr>
<td>printtokens</td>
<td>17.8</td>
<td>0.97</td>
<td>14/0.98</td>
<td>1.26</td>
</tr>
<tr>
<td>space</td>
<td>171.6</td>
<td>0.93</td>
<td>178/0.96</td>
<td>2.15</td>
</tr>
</tbody>
</table>

Table 1 Test Statistical Data
The multi-objective optimization problems came from designing, planning and modelling of real complex system. There are many real life important decisions, which are relative with multi-objective optimization. This kind of multi-objective optimization algorithm provides a method and way to solve multi-objective optimization problems. The development of multi-objective optimization algorithm has experienced the conventional multi-objective optimization algorithm and multi-objective algorithm based on evolutionary algorithm has achieved good results. There are many common features between the particle swarm algorithm and the traditional evolutionary algorithms. But the particle swarm algorithm is simpler and has easier operation than evolutionary algorithm. Using multi-objective particle swarm algorithm to solve multi-objective optimization problem is a hot topic this year. To further improve the optimization performance, achieve better solution for some nonlinear, complicated multi-objective optimization problems, we has proposed an improved particle swarm algorithm available for multi-objective optimization problems. This improved version of particle swarm optimization algorithm can increase the efficiency of automatic test case generation.

V. CONCLUSIONS

The multi-objective optimization problems came from designing, planning and modelling of real complex system. There are many real life important decisions, which are relative with multi-objective optimization. This kind of multi-objective optimization algorithm provides a method and way to solve multi-objective optimization problems. The development of multi-objective optimization algorithm has experienced the conventional multi-objective optimization algorithm and multi-objective algorithm based on evolutionary algorithm has achieved good results. There are many common features between the particle swarm algorithm and the traditional evolutionary algorithms. But the particle swarm algorithm is simpler and has easier operation than evolutionary algorithm. Using multi-objective particle swarm algorithm to solve multi-objective optimization problem is a hot topic this year. To further improve the optimization performance, achieve better solution for some nonlinear, complicated multi-objective optimization problems, we has proposed an improved particle swarm algorithm available for multi-objective optimization problems. This improved version of particle swarm optimization algorithm can increase the efficiency of automatic test case generation.

REFERENCES

