The Journal of Systems and Software 85 (2012) 2531–2540

Contents lists available at SciVerse ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Grouping target paths for evolutionary generation of test data in parallel

Dunwei Gong[a], Tian Tian[a,*], Xiangjuan Yao[b]

[a] School of Information and Electrical Engineering, China University of Mining and Technology, Xuzhou, Jiangsu 221116, PR China
[b] School of Science, China University of Mining and Technology, Xuzhou, Jiangsu 221116, PR China

ARTICLE INFO

Article history:
Received 31 August 2011
Received in revised form 18 May 2012
Accepted 23 May 2012
Available online 31 May 2012

Keywords:
Software testing
Test data
Multiple paths coverage

ABSTRACT

Generating test data covering multiple paths using multi-population parallel genetic algorithms is a considerable important method. The premise on which the method above is efficient is appropriately grouping target paths. Effective methods of grouping target paths, however, have been absent up to date. The problem of grouping target paths for generation of test data covering multiple paths is investigated, and a novel method of grouping target paths is presented. In this method, target paths are divided into several groups according to calculation resources available and similarities among target paths, making a small difference in the number of target paths belonging to different groups, and a great similarity among target paths in the same group. After grouping these target paths, a mathematical model is built for parallel generation of test data covering multiple paths, and a multi-population genetic algorithm is adopted to solve the model above. The proposed method is applied to several benchmark or industrial programs, and compared with a previous method. The experimental results show that the proposed method can make full use of calculation resources on the premise of meeting the requirement of path coverage, improving the efficiency of generating test data.

1. Introduction

Software testing is an important mean of guaranteeing software quality. The process is referred to as white-box testing if the source code of software under test is required during testing (Beizer, 1990). To check whether software meets the given requirements or not, some test adequacy criteria are needed for white-box testing. Among various criteria, the most commonly used one is path coverage, which requires test data generated can traverse the target path(s) of the program under test (Shan et al., 2005).

There are various methods of generating test data meeting path coverage. These methods are divided into static and dynamic ones according to whether software under test is executed, and dynamic methods need to execute software under test when generating test data (Shan et al., 2004). Generating test data covering target paths using genetic algorithms is one of dynamic methods which has brought to wide-ranging attention and achieved fruitful achievements (Xu, 2007). Genetic algorithm is a random search method inspired from biology evolution and heredity mechanism (Holland, 1975). For generating test data using genetic algorithms, the problem of generating test data is converted into an optimization one which is solved by genetic algorithms (Xanthakis et al., 1992).

Real-world software often contains multiple paths. It will undoubtedly improve the efficiency of generating test data if test data covering multiple target paths are generated in one run of a genetic algorithm. To this end, the problem of generating test data covering multiple paths can be converted into a multi-objective optimization one which is solved by existing evolutionary multi-objective optimization methods. By this way, the desired test data can be generated (Ahmed and Hermadi, 2008; Cao et al., 2010; Gong and Zhang, 2010).

But when there are many target paths, an optimization problem with many objectives will be formed which is very difficult to solve. If existing evolutionary multi-objective optimization methods are used to solve the problem, it will be very difficult, or even impossible to get test data meeting the requirement of path coverage. Therefore, a complicated optimization problem can be converted into several simpler sub-optimization problems after target paths have been divided into several groups according to appropriate methods, among which each group contains relatively fewer target paths. Each sub-optimization problem is solved by evolving a subpopulation to generate test data covering target paths belonging to the corresponding group. Consequently, test data covering all target paths are generated (Gong et al., 2011). Unfortunately, existing grouping methods often make a great difference in the number of target paths belonging to different groups, and the efficiency of generating test data using genetic algorithms needs to be further improved.

The problem of grouping target paths when generating test data covering multiple paths using genetic algorithms is investigated in this study. An appropriate method of grouping target paths is expected which allows a small difference in the number of target

* Corresponding author. Tel.: +86 516 83995312; fax: +86 516 83995312.
E-mail address: tian_tiantian@126.com (T. Tian).

0164-1212/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jss.2012.05.071

paths belonging to different groups and a great similarity among target paths in the same group. To this end, the number of groups is uniquely determined by calculation resources available, and the base path is selected according to the sum of path similarities. And target paths belonging to a group are selected according to similarities among base paths and others. After grouping target paths, a mathematical model is built for the problem of generating test data, and a strategy of generating test data using a multi-population parallel genetic algorithm is adopted.

The rest of the paper is organized as follows. The proposed method of grouping target paths is given in Section 2, which includes a strategy of grouping target paths, a mathematical model for the problem of generating test data after grouping target paths, a strategy of evolutionary generation of test data, and the steps of the proposed algorithm. In Section 3, the proposed method is applied to some benchmark and industrial programs, and the experimental results and analysis are also given. Section 4 reviews related work. Finally, Section 5 concludes this paper, and provides possible opportunities for future research.

2. Proposed method

For convenience of description, several related concepts are introduced before the method of grouping target paths. More detailed contents about these concepts can be found in Gong et al. (2011).

Node: Denote a program under test as P. A node is a statement block of P, and all statements contained in the node are either executed or not at all when P is run. A node may be the condition of a loop, the condition of a select statement, or one or more successive statements.

Path: It is a serial of nodes traversed by the control flow when P is run under a given input, and denoted as p. The number of nodes contained in a path is called the path length.

Similarity of paths: Consider two target paths p_i and p_j of P, and their nodes are $p_{i1}, p_{i2}, \ldots, p_{i|p_i|}$ and $p_{j1}, p_{j2}, \ldots, p_{j|p_j|}$, respectively, where $|p_i|$ and $|p_j|$ are the path length of p_i and p_j, respectively. Compare p_j and p_i from p_{j1} to check whether they have the same nodes, and denote the number of successively same nodes as $|p_i \cap p_j|$. The similarity between p_i and p_j is defined as the ratio of the number of successively same nodes to the larger path length, and denoted as $s(p_i, p_j)$, whose expression is as follows:

$$s(p_i, p_j) = \frac{|p_i \cap p_j|}{\max\{|p_i|, |p_j|\}} \tag{1}$$

Denote P's input as x, where x may be either a scalar or a vector. If x is a vector, the types of different components may be different. Denote the set of inputs as D and m target paths to be covered as $p_1, p_2, \ldots, p_m$. According to Ahmed and Hermadi (2008), the problem of generating test data covering these m target paths can be converted into a multi-objective optimization one, and its mathematical model is as follows:

$$\begin{aligned} &\min(f_1(x), f_2(x), \ldots, f_m(x)) \\ &\text{st.} x \in D \end{aligned} \tag{2}$$

where $f_i(x)$ represents the objective related to the ith target paths. $f_i(x)$ will reach the minimum if and only if the traversed path is just the ith target one when P is run under the generated test data.

2.1. Grouping target paths

There are two factors to be considered when grouping target paths: the number of target paths and the composition of each group. The idea of the proposed method of grouping target paths in this study can be described as follows. The number of target paths in each group is determined according to calculation resources available to narrow the difference of the number of target paths in different groups. Target paths belonging to a group are determined according to similarities of paths to improve the similarity between the base path and the others in the group.

Suppose that there are n_r calculation resources available, and they are homogeneous. For example, n_r computer nodes with the same configuration can be used to simultaneously generate test data covering target paths. To reduce the time consumption spent in generating test data, the load of these calculation resources should be balanced by and large. In other words, we expect to generate test data using different calculation resources, and the number of target paths to be covered has a small difference in different calculation resources.

Owing to only n_r calculation resources available, m target paths should be divided into n_r groups, denoted as $g_1, g_2, \ldots, g_{n_r}$, and test data covering target paths of each group are generated only by one calculation resource, so that the number of test data generated by each calculation resource is the smallest. The detailed strategy is as follows:

Denote the number of target paths contained in g_i as $|g_i|$. For the first group, g_1,

$$|g_1| = \left\lfloor \frac{m}{n_r} + 0.5 \right\rfloor \tag{3}$$

For the ith group, g_i, $i = 2, 3, \ldots, n_r$,

$$|g_i| = \left\lfloor \frac{m - \sum_{j=1}^{i-1} |g_j|}{n_r - i + 1} + 0.5 \right\rfloor \tag{4}$$

Now, we illustrate that the groups obtained by the method above have a small difference in the number of target paths.

The number of target paths belonging to the first group, g_1, can be expressed as:

$$|g_1| = \left\lfloor \frac{m}{n_r} + 0.5 \right\rfloor = \begin{cases} \left\lfloor \frac{m}{n_r} \right\rfloor \\ \left\lfloor \frac{m}{n_r} \right\rfloor + 1 \end{cases} \tag{5}$$

When $|g_1|$ is equal to $\left\lfloor \frac{m}{n_r} \right\rfloor$, we have

$$\frac{m}{n_r} - 0.5 \le \left\lfloor \frac{m}{n_r} \right\rfloor \le \frac{m}{n_r} \tag{6}$$

Combining formulas (4)–(6), we have:

$$\begin{aligned} \left\lfloor \frac{m}{n_r} \right\rfloor &= \left\lfloor \frac{m}{n_r} + 0.5 \right\rfloor = \left\lfloor \frac{m - (m/n_r)}{n_r - 1} + 0.5 \right\rfloor \le |g_2| \\ &= \left\lfloor \frac{m - \lfloor m/n_r \rfloor}{n_r - 1} + 0.5 \right\rfloor \le \left\lfloor \frac{m - (m/n_r) + 0.5}{n_r - 1} + 0.5 \right\rfloor \\ &= \left\lfloor \frac{m}{n_r} + 0.5 + \frac{0.5}{n_r - 1} \right\rfloor \le \left\lfloor \frac{m}{n_r} + 1 \right\rfloor \le \left\lfloor \frac{m}{n_r} \right\rfloor + 1 \end{aligned} \tag{7}$$

According to formulas (5) and (7), we have:

$$\left\lfloor \frac{m}{n_r} \right\rfloor \le |g_2| \le \left\lfloor \frac{m}{n_r} \right\rfloor + 1 \tag{8}$$

When $|g_1|$ is equal to $\left\lfloor \frac{m}{n_r} \right\rfloor + 1$, we have:

$$\frac{m}{n_r} - 1 \le \left\lfloor \frac{m}{n_r} \right\rfloor \le \left\lfloor \frac{m}{n_r} \right\rfloor - 0.5 \tag{9}$$

Combining formulas (4), (5) and (8), we have: