Automatic Test Data Generation Approach using Combination of GA and PSO with Dominance Concepts

Sanjay Singla¹, H. M. Rai², and Priti Singla³

¹Research Scholar, Department of Computer Science & Engg, Suresh Gyan Vihar University, Jaipur, Rajasthan, INDIA
²Department of Electronics & Communication N C College of Engineering, Israna, Panipat, Haryana, INDIA
³Research Scholar, Department of Computer Science & Engg, Suresh Gyan Vihar University, Jaipur, Rajasthan, INDIA
¹E-mail: san_jay23@yahoo.com

Abstract: This paper presents an automatic test data generation technique that uses a new a new Algorithm called CGPSA (Combined Genetic-Particle Swarm Algorithm) that based on a combination of genetic algorithm (GA) and particle swarm optimization (PSO). It uses dominance relation between nodes. Finally, the paper presents the results of the experiments using this technique with new proposed algorithm and show the effectiveness of this approach as compare to other algorithms i.e., GA and PSO.

Keywords: Genetic algorithms, data flow testing, particle swarm optimization, dominance concept.

1. INTRODUCTION

Testing is a costly and time consuming phase in software development process. One critical task in software testing is the creation of test data to satisfy a given test-coverage criterion. This process is known as test-data generation [25]. A number of test data generation techniques have been proposed by researchers from time to time [4]-[7], [9]-[12], [29]. Recently, the genetic algorithms (GAs) are playing an important role in the area of software testing [23]-[24]. Test data generation method based on the GA uses fitness function that is based on control dependencies or branch distance in its calculations. The fitness function that consider branch distance face problem with nested branch predicated and by using control dependencies faces a problem to find an input to traverse a target node within loops [28]. To overcome this problem and reduce the test cases thus testing cost, we proposed a new approach CGPSA with new fitness function that uses the concept of dominance relationship between nodes.

2. BACKGROUND

We introduce here some basic concepts that are used throughout this work.

2.1 The Control Flow Graph

The control flow graph (CFG) of a program can be represented by a directed graph \( G = (V, E) \) with a set of nodes \( (V) \) and a set of edges \( (E) \). Each node represents a group of consecutive statements, which together constitute a basic block. The edges of the graph are then possible transfers of control flow between the nodes. Fig. 2 shows the control flow graph \( G \) of the example program, which is shown in Figure. 1.

Fig. 1: Program 1

Fig. 2: Control Flow Graph

2.2 Dominance Tree

For \( G = (V, E) \), a directed graph with two distinguished nodes \( n_0 \) and \( n_e \), a node \( n \) dominates a node \( m \), if every path \( P \) from the entry node \( n_0 \) to \( m \) contains \( n \). A dominator tree \( DT(G) = (V, E) \) is a directed graph in which one distinguished node \( n_0 \) called the root, is the head of \( n_e \) edge; every node \( n \) except the root \( n_0 \) is a head of just one edge and there exists a (unique) path (dominance path dom \( (n) \)) from the root \( n_0 \) to each node \( n \) [25]. Figure. 3 gives the dominator tree of program 1. The dominance path of node 5 is dom (5) = 1, 2, 3, 5.
2.3 Reduce the Test Cases

Our proposed approach is based on the concept of dominance relationship between nodes of the program’s control flow graph. The main objective is to cover a subset of statement that guarantees the coverage of all statements of the tested program [28]. The set leaves nodes of the dominance tree are very important as every set of path that covers it also covers all nodes in tree. The set of leaves node of dominance tree shown in Fig. 3 is \( A = \{4, 5, 6, 7\} \). The dominance path of each element of set \( A \) is as follow:

- \( \text{dom}(4) = 1, 2, 3, 4 \)
- \( \text{dom}(5) = 1, 2, 3, 5 \)
- \( \text{dom}(6) = 1, 2, 3, 6 \)
- \( \text{dom}(7) = 1, 2, 7 \)

Covering an element from the set \( A \) means covering its dominance path. Covering all elements of set \( A \) means covering all dominance paths which results in covering all nodes of CFG that results in the coverage of all statements.

3. GPSCA

The proposed GPSCA combines the power of both GA with PSO to form a hybrid algorithm. The combination of GA and PSO always performs better than GA or PSO alone [29-30]. The proposed GPSCA consists of three major operators: enhancement, crossover, and mutation [30].

**Enhancement**

In each generation, after the fitness values of all the individuals in the same population are calculated, the top-half best-performing ones are marked. These individuals are regarded as elites. Instead of reproducing the elites directly to the next generation as elite GAs do, we first enhance the elites. The enhancement operation tries to mimic the maturing phenomenon in nature, where individuals will become more suitable to the environment after acquiring knowledge from the society. Furthermore, by using these enhanced elites as parents, the generated off-springs will achieve better performance than those bred by original elites. PSO is used to enhance individuals of the same generation. Here, the group constituted by the elites may be regarded as a swarm, and each elite corresponds to a particle in it. In PSO, individuals of the same generation enhance themselves based on their own private cognition and social interactions with each other. In GPSCA, we adopt and regard this technique as the maturing phenomenon. Based on PSO, (1) may be applied to the elites.

**Crossover**

To produce well-performing individuals, in the crossover operation parents are selected from the enhanced elites only. To select parents for the crossover operation, the roulette wheel selection scheme is used. Two off-springs are created by performing crossover on the selected parents. In this study, we used single point crossover.

**Mutation**

Mutation is an operator whereby the allele of a gene is altered randomly so that new genetic materials can be introduced into the population. Mutation probability \( P_m = 0.1 \) is used by us.

4. FITNESS FUNCTION

The algorithm uses a new evaluation (fitness) function to evaluate the generated test data based on the concepts of the dominance relations between nodes of the program’s control flow graph. The fitness function used to evaluate each test case by executing the program with it as input, and recording the traversed nodes in the program that are covered by this test case. We denote to the set of traversed nodes by \( \text{exePath} \). Also, it finds the dominance path \( \text{dom}(n) \) of the target node \( n \). The fitness function is the ratio of the number of covered nodes of the dominance path of the target node to the total number of nodes of the dominance path of the target node. The fitness value \( f_t(v_i) \) for each chromosome/particle \( (i = 1, ..., \text{pop_size}) \) is calculated as follows:

1. Find \( \text{exePath} \): the set of the traversed nodes in the program that are covered by a test case.
2. Find dom(n): dominance path of the target node n (the set of dominator nodes from the entry of the dominator tree to n).

3. Determine (dom(n) - exePath): uncovered nodes of the dominance path (the difference between the dominance path and the traversed nodes).

4. Determine (dom(n) - exePath'): covered nodes of the dominance path (the complement set of the difference set between the dominance path and the traversed nodes).

5. Calculate |(dom(n) - exePath')|: number of covered nodes of the dominance path (cardinality of the complement set).

6. Calculate dom(n): number of nodes of the dominance path of the target node n (cardinality of the dominance set).

Then, \[ f_{ij} = \frac{|(dom(n) - exePath')|}{|dom(n)|} \] (1)

The fitness value is the only feedback from the problem for the GA and PSO. A test case that is represented by the chromosome \( v_i \) is optimal if its fitness value \( f_{ij} = 1 \) [28].

5. EXPERIMENTAL RESULTS

Our technique takes the instrumented version of the program under test \( P' \) and the dominator tree (DT) as inputs and returns the total number of generation, number of test cases and cover ratio percentage. Initial population for GA, PSO and GPSCA program is generated randomly.

For implementation/programming purpose, we used MATLAB. The effectiveness of the proposed GPSCAO is compared with GA and PSO. We perform our new technique GPSCA on set of programs and compare it with the GA and PSO on the same set of programs to demonstrate its effectiveness in achieving the test cover ration in less number of generations.

6. CONCLUSION

As shown in figure 5, our GPSCA performs better than GA and PSO in number of generation and in some cases GPSCA is successful in achieving more def-use coverage percentage as shown in figure 6. From figure 7 it can be analyzed easily that number of test cases generated in case of GPSA is less than GA and PSO. Overall, it can be concluded that performance of GPSCA is better as compare to GA and PSO.

REFERENCES


