Journal of Basic and Applied Engineering Research
Print ISSN: 2350-0077; Online ISSN: 2350-0255; Volume 2, Number 10; April-June, 2015 pp. 850-853
© Krishi Sanskriti Publications
http://www.krishisanskriti.org/jbaer.html

An Approach for Data Flow Testing in Object-Oriented Programming through Particle Swarm Optimization

Sonam Bhati[1] **and Pradeep Tomar**[2]

[1,2]*Dept. Computer Science and Engg. School of ICT Gautam Buddha University, Greater Noida, Uttar Pradesh, India*
E-mail: [1]*sonam2709@gmail.com,* [2]*parry.tomar@gmail.com*

Abstract—*Software Testing plays a crucial role in the software development lifecycle as it is used to improve the quality and increase the reliability of software. Software testing successfulness is always determined on the basis of generated test cases and their prioritization. So, it consumes more effort, time and cost. Today, a numerous soft computing based approaches are available for better accuracy in testing. The aim of this paper is to provide review of some of the recent work that has been done in the area of software testing by using computational techniques. Based on those work, This paper proposes an approach for data flow testing using PSO. This paper discusses how PSO algorithm is used for optimizing in the issue of data flow testing. It will use a simple algorithm of PSO (Particle Swarm Optimization) inspired by social metaphors of behaviour which uses the concepts for optimizing the nonlinear function of particle swarm theory for data flow testing which guarantees full path coverage.*

Keywords: *Data Flow Testing, Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Control Flow Graph (CFG), Dominance Tree.*

1. INTRODUCTION

Software quality can be measured through product reliability and customer's satisfaction. An exhaustive testing ensures the quality of the software. Software testing is generally divided into white box testing and black box testing. White box testing is also known as structural testing as the structure or logic is considered in this testing. Testing mainly comprises of static testing and dynamic testing. Static testing does not need any tool as the testing of program is done without executing the program and it is done through verification and dynamic testing is performed with the execution of code and it is a part of validation. In this testing, test data is given to the system in the form of input and results are checked against the expected output while executing the software as the structure or logic is not considered in this testing. On the other hand, white box testing is another essential technique in dynamic testing. It is also known as structural testing as whole structure design and code is tested and its aim is to test the internal parts and uncover bugs as many as possible in the logic of the program.

In Software Development Process, testing is costly and time consuming phase. There are some of essential task in software testing to create the test data to satisfy a given test-coverage criterion. This process is known as test-data generation. Also, there are many crucial activities associated with software testing such as:

- To cover a certain testing criterion, find the path cover.
- To satisfy the path cover, generate the test data.
- Test execution by using the test data and the software under test.
- Evaluation of test results.

Today, for a large project there are large numbers of test cases required. So it becomes difficult for tester to test large and complex programs. Therefore, there is need to reduce the testing set to generate optimal test data which further reduces the time and cost involved in testing. This work focuses on the computational technique which is guided by data flow dependencies in the program to search for test data to fulfil data flow selection criterion. This paper presents an algorithm of PSO technique to generate test data which gives a strong level of software coverage. In this paper the emphasis is given on PSO algorithm which will be used for data flow testing.

The rest of the paper is organized as follows. Section 2 gives some basic concepts and definitions and survey of various research papers related to dynamic testing Section 3 describes about data-flow analysis technique. Section 4 shows a PSO algorithm which is used for optimizing data-flow testing in the proposed approach. Section 5 introduces conclusion and future work.

2. BACKGROUND

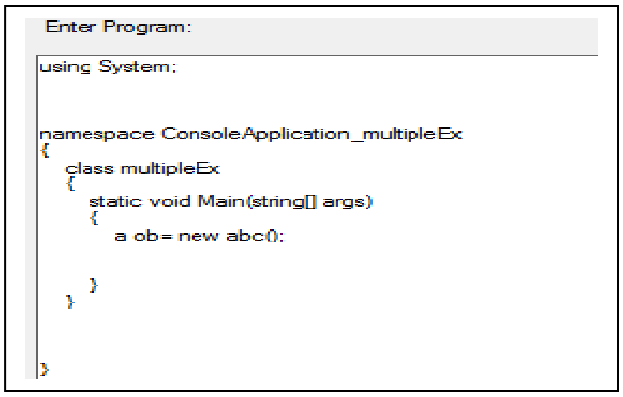
Here, this paper discusses about some basic concepts that are used throughout this work.

2.1. Control Flow Graph

The *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.

2.2. Dominance Tree

For *G* = (*V*, *E*), a directed graph with two distinguished nodes *n0* and *nk*, A node n dominates a node m, if every path P from the entry node n0 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*o 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]. Fig. . 2 gives the dominator tree of program 1.



```
Enter Program:

using System;


namespace ConsoleApplication_multipleEx
{
  class multipleEx
   {
     static void Main(string[] args)
      {
        a ob= new abc();


      }
   }


}
```

Fig. 1: Program 1

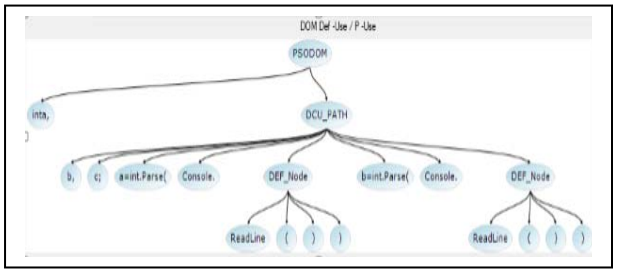


Fig. 2: Dominance Tree

As the present market is highly competitive, it is a pressing need of software organizations to provide good quality software products to the customer within the estimated budget, and hence a strong level of testing coverage technique is essential. Recently, there are some novel search-based optimizations techniques have been developed such as Ant Colony Optimization (*ACO*), Particle Swarm Optimization (*PSO*), Bees Colony Optimization, and Artificial Immune System (*AIS*) which are used in optimizing the testing. An ACO algorithm is a probabilistic technique for solving computational problems which can be used to find “good" paths through the graph. It depends on the behaviour of ants in finding paths from their colony to food. Although, this algorithm has been already proposed in the issue of software data-flow testing [1].

Ahmed S. Ghiduk et.al. [1] has proposed the ACO algorithms in the issue of software data-flow testing and an ant colony optimization based approach for generating set of optimal paths to cover all definition-use associations (du-pairs) in the program under test. The ant colony algorithms has been adopted to search the CFG and a model built on the program input domain in order to get the path cover and the test data that satisfies the selected path. Fig 1[1] provides a block diagram of the technique used in this paper. They have also used two ant colony algorithms for using with data flow testing .One algorithm generates set of paths for covering all def-use pairs in the software under test (*SUT*) and the other algorithm finds set of test data to satisfy this set of paths.

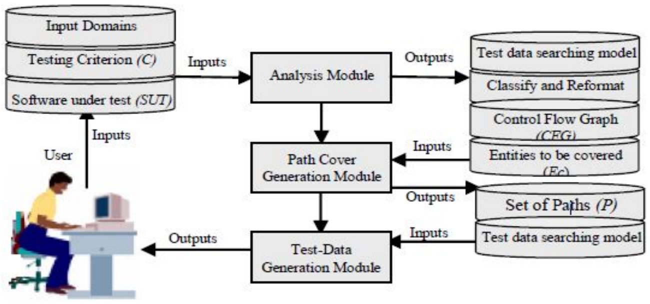


Fig. 3: Block Diagram of Data Flow Testing Technique

D. Jeya Mala et.al. [2] has proposed a new non-pheromone-based test suite optimization approach inspired by the behavior of biological bees based on ABC which is motivated by the intelligent behavior of honey bees. In their approach, the sites are the nodes in the Software under Test (SUT), the artificial bees modify the test cases with time and the bee’s aim is to discover the places of nodes with higher coverage and finally the one with the highest usage by the given test case. Also, they focused on investigating whether this new approach outperforms existing test optimization approach based on Genetic Algorithms (GA) in the task of software test optimization.

B. Holldobler and E. O. Wilson [3] proposed an ant colony optimization technique which is a set of instructions based on search algorithms of artificial intelligence for optimal solutions; here the iconic member is ANT System. Ants are blind and small in size and still are able to find the shortest route to their food source with the use of antennas and pheromone liquid to be in touch with each other. ACO inspired from the behavior of live ants, are capable of synchronization with searching solutions for local problem by maintaining array list to maintaining previous information gathered by each ant.

P.R Srivastava et.al. [2] presents an algorithm by applying an ant colony optimization technique, for generation of optimal and minimal test sequences for behavior specification of software and they have also used approach to generate test sequence in order to obtain the complete software coverage. There have also been discussed the comparison between two meta- heuristic techniques (Genetic Algorithm and Ant Colony optimization) for transition based testing. In their paper, they demonstrate the generation of the optimal test sequence for the state -transition based software testing. It also describes the standard method of state transition based testing and its coverage level but the existing method using GA does not provide full coverage for software.

R.Lefticaru and F.Ipate [5] devised a Genetic Algorithm based on test data generation technique. They have used UML state diagram for test data generation in order to generate test data before coding, In their work, three major steps were involved in testing such as generating set of test inputs, execution of those inputs on the program under tests, and then checking whether the test executions reveal faults.

3. DATA FLOW ANALYSIS TECHNIQUE

In this technique each variable is classified as either a definition occurrence or a use occurrence. A definition occurrence of a variable is where a value is associated with the variable. A use occurrence of a variable is where the value of the variable is referred. Each use occurrence is further classified as a computational use (c-use) or a predicate use (p-use). If the value of the variable is used to decide whether a predicate is true for selecting execution paths, the occurrence is called a predicate use. Otherwise, the occurrence is called a computational use. Their criteria require that test data to be included which cause the traversal of sub-paths from a variable definition to either some or all of the p-uses, c-uses, or their combination.

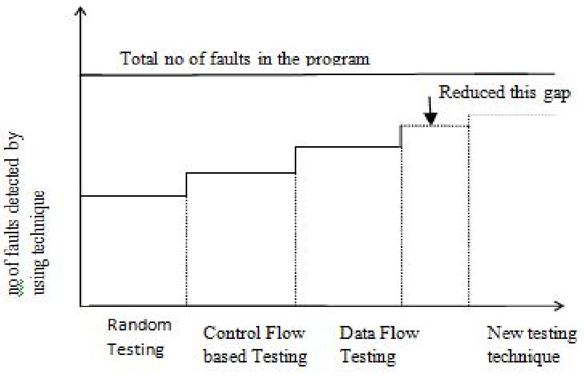


Fig. 4: Comparison of Testing Techniques

However, data-flow testing is important because it expanded the concept of control-flow testing criteria and focused on how a variable is defined and used in the program, which could lead to more efficient and targeted test suites. The results of using ant colony optimization algorithms in software testing which is obtained so far are preliminary and none of the reported results directly addresses the problem of test-data generation or path-cover finding for data-flow based software testing but the testing with PSO can be made more effective. This paper aims at proposing an approach in which PSO algorithm is applied in software data-flow testing. This technique is based on generating set of optimal paths to cover all definition-use associations (def-use or *du*-pairs) in the program under test. Fig. [4] shows the comparison of various testing techniques.

4. PROPOSED APPROACH

In this paper, it shows a simple algorithm of PSO (Particle Swarm Optimization) developed by Kennedy and Eberhart inspired by social metaphors of behavior which uses the concepts for optimizing the nonlinear function of particle swarm theory. This approach is based on generating set of optimal paths to cover all definition-use associations (*du*-pairs) in the program under test. PSO is initialized with a group of random particles and each individual in PSO is assigned with a randomized velocity according to its own and its companions "flying experiences" and the individuals called particles, are then flown through hyperspace. It has memory, so knowledge of good solutions is retained by all particles. Each particle is updating by two following best values at each iterations. The first one is the best solution it has achieved so far, this value is called pbest. Another best value that is tracked by particle swarm optimizer is a best value obtained so far by any particle in the population. PSO can perform much better in achieving more def-use coverage as compared to other existing search based optimization techniques like ACO, GA etc as it has the advantage of memory so it can keep information of good solutions of all particles. The algorithm of PSO consists of these steps given below:

```
1. Initiate Swarm.
2. Repeat
3. For p=1 to number of particles do
4. Evaluate (p)
5. Update past experience (P)
6. Update neighbourhood best (p, k)
7. For d=1 to number of dimensions do
8. Move (p, d)
9. End for
10. End for
11. until criterion.
```

The continuous PSO version uses a real-valued multidimensional space as belief space, and evolves the position of each particle in that space using the following equations:

$$v_{id}^{t+1} = w \cdot v_{id}^{t} + c_1 \cdot \psi_1 \cdot (p_{id}^{t} - x_{id}^{t}) + c_2 \cdot \psi_2 \cdot (p_{gd}^{t} - x_{id}^{t})$$

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1}$$

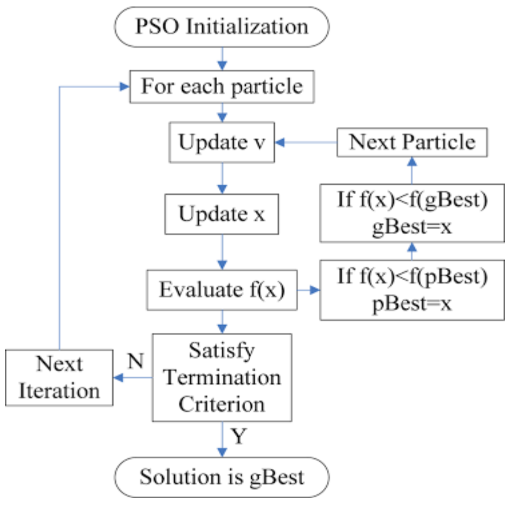


Fig. 5: Flow Graph

5. CONCLUSION AND FUTURE WORK

This paper has reviewed various research papers based on dynamic testing and has found that most of the works concentrates on the coverage but none of them told about which technique is better suited for full coverage. Evolutionary structural testing is an approach used to generate test cases that uses GA, ACO or other search based optimization which is guided by data flow dependencies in the program to cover the def-use association. Since cost and coverage are two important factors in case of testing. This paper has proposed a process for data flow testing using PSO as it is found that by using PSO algorithm, performance of testing can be improved as well as testing can be done more effectively. In Future work, the PSO algorithm will be implemented in performing data flow testing to provide an efficient path with maximum code coverage and minimum cost. Furthermore, the results can be compared with other meta-heuristic techniques such as PSO and ACO etc.

6. ACKNOWLEDGEMENT

This research paper is made possible through the help and support from everyone, including: parents, teachers, family and friends. Especially, please allow me to dedicate my acknowledgment of gratitude toward the following significant advisors and contributors:

First and foremost, I would like to thank Dr. Pradeep Tomar for his most support and encouragement. He kindly read my paper and offered invaluable detailed advices on grammar, organization and the theme of the paper. Finally, I sincerely thank to my parents, family, and friends, who provide the advice and financial support. The product of this research paper would not be possible without all of them.

REFERENCES

[1] Ahmed S. Ghiduk" A New Software Data-Flow Testing Approach via Ant Colony Algorithms" *Universal Journal of Computer Science and Engineering Technology* 1 (1), 64-72, Oct. 2010. © 2010 Uni CSE, ISSN: 2219-2158.

[2] Praveen Ranjan Srivastava and KmBaby " Automated Software Testing Using Meta-heuristic Technique Based on An Ant Colony Optimization" *IEEE Transactions on Software Engineering*, vol. 3, no. 4, 266-278, 1977.

[3] D .Jeya Mala, V. Mohan " ABC Tester -Artificial Bee Colony Based Software Test Suite Optimization Approach.*"Proc. of 7th International Conference on Hybrid Intelligent Systems* (HIS'07), Sept. 2007, pp. 84-89. IEEE Press.

[4] K. Li, Z. Zhang and W. Liu "Automatic Test Data Generation Based On Ant Colony Optimization" *Proc. of Fifth International Conference on Natural Computation* 2009, pp. 216-219. IEEE Press .

[5] P. R. Srivastava"An Approach of Optimal Path Generation using Ant Colony Optimization" *Proc. of TENCON* 2009, pp.1-6. IEEE Press.

[6] M. Dorigo and C. Blum "Ant colony optimization theory: A survey*" Theoretical Computer Science*, 344(2-3), pp. 243-278, 2005.

[7] Anu Sharma, Arpita Jadhav, Praveen Ranjan Srivastava and Renu Goyal "Test Cost Optimization Using Tabu Search", *Software Engineering & Applications*, 2010, 3: 477-486.

[8] Liu X. B., Cai Z. X., "Artificial Bee Colony Programming Made Faster", *Fifth International Conference on Natural Computation (ICNC)*, August, 2009, Vol. 4, pp. 154-158, 14-16.

[9] B. Holldobler and E. O. Wilson, "The Ants, Berlin: "*Springer-Verlag*, 1990.

[10] R. Lefticaru and F. Ipate, "Automatic State-based test generation Using genetic algorithms," in *Proc. Ninth International Symposium on Symbolic and Numeric Algorithms 2007)*, 2007, pp.188-195

[11] R. C. Eberhart and J. Kennedy, "A New Optimizer using Particle Swarm Theory", *6th International Symposium on Micro machine Human Science*, pp. 39-43, 1995.