www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 5 Issue 12 Dec. 2016, Page No. 19521-19628

New Soft Computing Techniques in the Software Testing

Suraj Indiver[1] , Mr. Vyom Kulshreshtha[2]
M.Tech Student, Computer Science and Engineering,
SIT, Mathura, Dr. A.P.J.AKTU Lucknow
(Asstt. Professor)
(Supervisor), Computer Science & Engineering
SIT, Mathura, Dr. A.P.J.AKTU Lucknow

Abstract: Testing is tedious, time consuming and costly. This dissertation explores self test data creation technique. It uses a new algorithm called Genetic-Particle Swarm Combined Algorithm (GPSCA) which is based on a combination of Genetic Algorithm (GA) and Particle Swarm Optimization (PSO). It uses dominance relationship between two nodes. The test data were derived from the program's structure with the aim to traverse every statement in the software. 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. The main reason of using GPSCA is its ability to handle input data which may be of complicated and complex and difficult to determine manually. Thus, the problem of test data generation is treated entirely as an optimization problem. The performance of the proposed approach is analyzed on a number of programs having different size and complexity. Finally, the performance of GPSCA is compared to both GA and PSO for generation of automatic test cases to demonstrate its superiority. The effectiveness of test data generation using GPSCA is better than GA and PSO as it requires fewer tests than later and achieves 100% coverage in less number of generations

Keywords: GAs, PSO, GPSCA, Ant Colony Optimization, Dominance Tree, Neural Networks.

1. INTRODUCTION

Software test is the main approach to find errors and defects assuring the quality of software [1]. Therefore much efforts and time is required in order to generate test cases to test the software for reliability and other functionalities purposes. Manually, generation of test cases consumes a lot of time and it also depends on the skill of person. Therefore chances of errors at the time of designing of test cases are immense which leads to the inclusion of bugs in the system after testing also. On

The other hand, some test cases are better than the others in terms of finding the errors. Therefore, a testing system is

required to differentiate good (suitable) test data from bad test (unsuitable) data, and so it should be able to detect good test data if they are generated. To overcome this, it is essential to automate test data generation. Software testing tools must ensure that the test cases generated by it is falling under the corresponding testing criteria and are of good quality. Testing tool must generate test cases with diversified nature and it should not fall in local optima. Tool must be robust, reliable, general and adaptive. Test data which is generated for one program may or may not be necessary good for another program. Therefore tools must be of adaptive in nature for generating test cases for the software under test consideration. The thesis presents the result of the research done in the area of software testing using the soft computing approach which gives adequate picture of the research project with special reference to software testing using soft computing approach. Different soft computing approaches such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO) and hybrid of GA and PSO are used. These are used to compare and find the minimum software test cases for testing the software.

2. OBJECTIVES OF THE RESEARCH

To achieve the overall purposes of the dissertation following are the objectives:

a) To identify, characterize and to automatic prioritization of test cases in software testing using techniques like control flow analysis etc.
b) To propose a new approach for software testing process, optimizing testing efforts, testing complexity, quality and reliability issues.
c) To assess the feasibility of proposed soft computing technique to automatically generate test data for software testing.
d) To compare the results obtained with existing methods such as GA and PSO.

3. HYPOTHESES

a) To achieve the goal and aim of research following hypotheses are taken into consideration which are justified in the experiment and result chapter.
b) Genetic Algorithms (GAs) are efficient in generating the test cases for the software/ program under test.
c) Particle Swarm Optimization (PSO) is more powerful and efficient as compare to Genetic Algorithm (GA) in order to achieve the goal or generating test cases.

d) Genetic-Particle Swarm Combined Algorithm (GPSCA) is more powerful and efficient as compare to PSO and GA in order to achieve the goal or generating test cases.
e) Change in size of population give best results in case of GA and GPSCA.
f) Change in crossover probability give best results in case of GA and GPSCA.
g) Change in mutation probability give best results in case of GA and GPSCA.
h) Change in number of agents give best results in case of PSO and GPSCA.

4. CLASSIFICATION OF TESTING

A. Black Box Testing

Black box testing is done without the knowledge of the internals of the system under test. Black box testing is done from customer's view point. It involves looking at the specifications and does not require examining the code of a program. The test engineers engaged in black box testing only knows the sets of input and expected output and is unaware of how those inputs are transformed into output by software. Black box testing requires functional knowledge of the product to be tested [1, 9].

B. White Box Testing

White box testing is a way of testing the external functionality of the code by examining and testing the program code that realize the external functionality. This is also known as clear box, or glass box or open box testing. It is also called program based testing [2]. White box or logic-driven testing permits you to examine the internal structure of the program. This strategy derives test data from an examination of the program's logic [3]. White box testing takes into account the program code, code structure, and internal design flow [3].

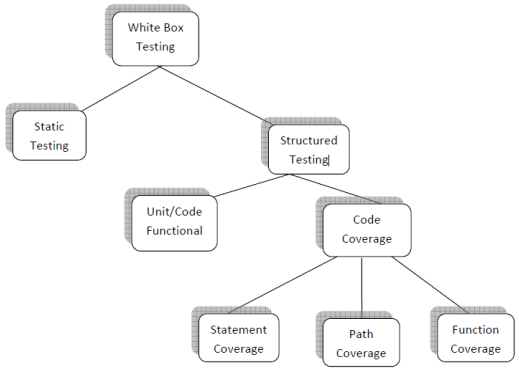

Figure 1: Classification of white box testing

5. TESTING PROBLEMS

To test the software, test cases are written. In order to find out how a test case is valid, one does not have a definite mechanism. One basically depends on the testers understanding of the requirement. In this process, one has lot of human error and his basic skill level taken into consideration. This leads to the inclusion of bugs in the system after testing also. To overcome this, it is essential to Automate Test Data Generation**.** Automated test data generation reduces an effort of software developers for creating test cases with a goal to minimize the amount of manual work involved in text execution and gain higher coverage with minimum cost and time.

6. ARCHITECTURE OF TEST AUTOMATION

Design and architecture is an important aspect of automation. The architecture of test automation is shown in figure 2. Architecture for test automation involves two major heads:

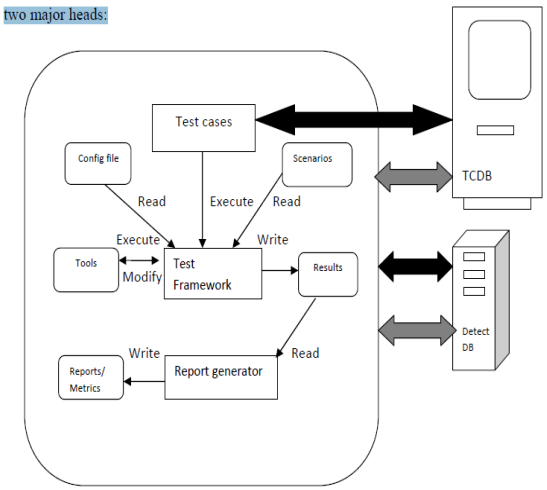

Figure 2: Test automation architecture

a) A test infrastructure that covers a test case database
b) A defect database or defect repository

There is no hard and fast rule on when automation should start and when it should end. The work on automation can go simultaneously with product development. It can overlap with multiple release of the product. Product and automation go parallel in the same direction with similar expectation.

7. SOFT COMPUTING TECHNIQUES

7.1 OPTIMIZATION

It is the process of making something better. Optimization is the process of adjusting the inputs to find the minimum or maximum output or result [4]. The Optimization Process is shown in figure 3. Optimization is the mechanism by which one finds the maximum or minimum value of a function or process. This mechanism is used in fields such as physics, chemistry, economics, and engineering where the goal is to maximize efficiency, production or some other measure.

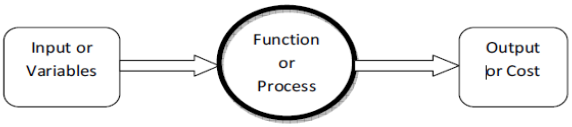

Figure 3: Optimization process

7.2 NEURAL NETWORKS (NNS)

There are millions of very simple processing elements or neurons in the brain, linked together in a massively parallel manner. This is believed to be responsible for the human intelligence and discriminating power [5]. Neural Networks are

developed to try to achieve biological system type performance using a dense interconnection of simple processing elements analogous to biological neurons. Neural Networks are information driven rather than data driven [6]. Typically, there are at least two layers, an input layer and an output layer. One of the most common networks is the Back Propagation Network (BPN) which consists of an input layer, and an output layer with one or more intermediate hidden layers [7].

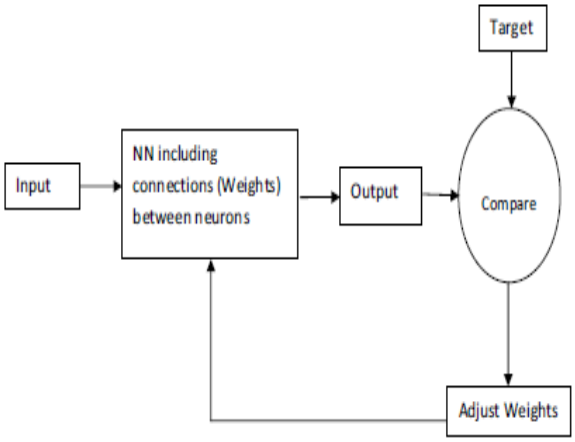

Figure 4: Neural networks

7.3 GENETIC ALGORITHMS (GAs)

GAs are general-purpose search algorithms, which use principles inspired by natural genetics to evolve solutions to problems [8]. As one can guess, genetic algorithms are inspired by Darwin's theory about evolution. They have been successfully applied to a large number of scientific and engineering problems, such as optimization, machine learning, automatic programming, transportation problems, adaptive control etc. GA starts off with population of randomly generated chromosomes, each representing a candidate solution to the concrete problem being solved and advances towards better chromosomes by applying genetic operators based on the genetic processes occurring in nature. So far, GAs had a great measure of success in search and optimization problems due to their robust ability to exploit the information accumulated about an initially unknown search space.

7.3.1 Representation of Chromosomes

The representation of chromosomes in GAs has very deep impact on the performance of GA-based function. There are different methods of representation of chromosomes like binary encoding, value encoding, permutation encoding, tree encoding etc. The most commonly used encoding is binary encoding proposed by Holland [9]. In this method, the value of individual is encoded as bit string consists of binary values either 0 or 1. Each chromosome of population consists of same length of binary string [10]. Suppose a program has two inputs X and Y having value 8 and 6 respectively and length of binary string is 5. Then, X and Y can be represented as shown in table 1.

Chromosome	Value	Binary Encoding
X	8	01000
Y	6	00110

Table 1: Binary Encoding

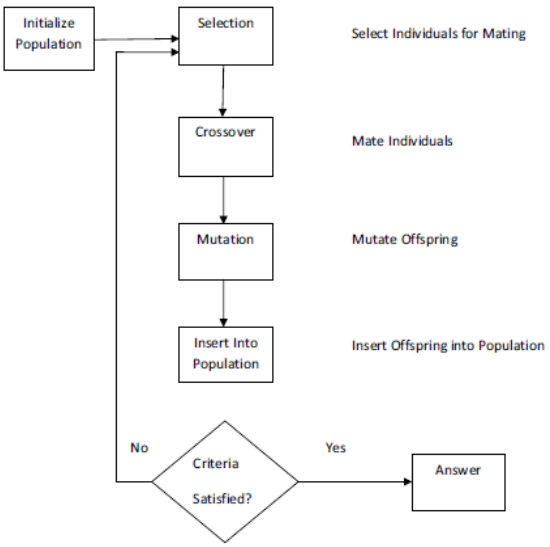

Figure 5: Fundamental mechanism of simple genetic algorithm

7.4 PARTICLE SWARM OPTIMIZATION (PSO)

Although GAs provides good solution but they not keep information about the best solution in the whole community. This strategy extends search by the introduction of memory. In this optimization, along with the local best solution, a global best solution is also stored somewhere in the memory, so that all particles not trapped into local optima but moves to global optima. PSO is an algorithm developed by Kennedy and Eberhart [11] that simulates the social behaviors of bird flocking or fish schooling and the methods by which they find roosting places, foods sources or other suitable habitat. The algorithm maintains a population potential where each particle represents a potential solution to an optimization problem. The PSO algorithm works by simultaneously maintaining several candidate solutions in the search space. During each iteration of the algorithm, each candidate solution is evaluated by the objective function being optimized, determining the fitness of that solution. Each candidate solution can be thought of as a particle "flying" through the fitness landscape finding the maximum or minimum of the objective function. Initially, the PSO algorithm chooses candidate solutions randomly within the search space. The flowchart of PSO algorithm is shown in figure 6(a).

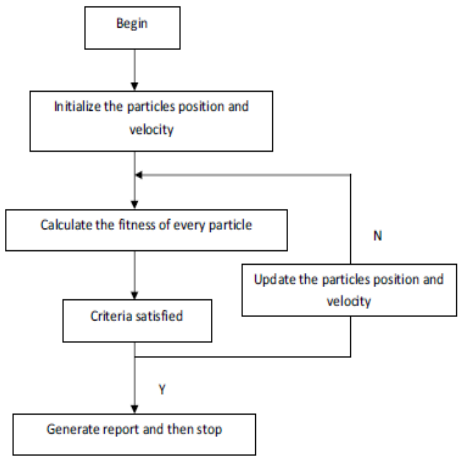

Figure 6 (a): Flow chart of PSO

The overall processing of PSO is shown in figure 6(b). PSO accepts list of parameters from the input file **pso1.run**. It produces the report in the output file **pso.out**.

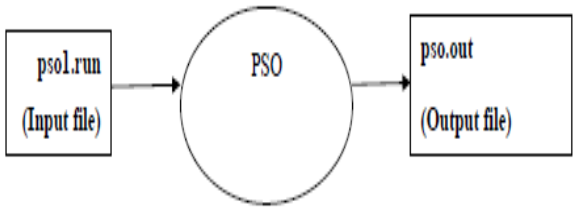

Figure 6 (b): PSO Process Diagram

7.5 ANT COLONY OPTIMIZATION (ACO)

The idea of ant colony optimization is as its name suggests, inspired from the ant colonies. Ant Colony Optimization (ACO) is a population-based, general search technique for the solution of difficult combinatorial problems, which is inspired by the pheromone trail laying behavior of real ant colonies [12]. Each ant moves along some unknown path in search of food and while it goes it leaves behind a trail of what is known as pheromone. The special feature of this pheromone is that it evaporates with time such that as time proceeds, the concentration of the pheromone decreases on any given path. Now it's obvious that the path with maximum pheromone is the one that has been traversed the most recently or in fact by most number of ants and hence the most desirable for following ant [13]. The first ACO technique is known as Ant System [14] and it was applied to the traveling salesman problem. This work was further carried by Dorigo, Di Caro, Blum etc [14-19]. Initial attempts at an ACO algorithm were not very satisfying until the ACO algorithm was coupled with a local optimizer.

One problem is premature convergence to a less than optimal solution this is because too much virtual pheromone was laid quickly. To avoid this stagnation, pheromone evaporation is implemented. In other words, the pheromone associated with a solution disappears after a period of time [20].

Step 1 : Initialization the pheromone trail
Step 2 : Iteration
- **For each ant repeat**
- **Solution construction using the current pheromone trail**
- **Evaluate the solution constructed**
- **Update the pheromone trail**
- **Until stopping criteria**

Figure 7: A Generic Ant Colony Algorithm

8. PROBLEM STATEMENT & PROPOSED APPROACH

8.1 The proposed approach composed of mainly two components:

a) Proposed Technique - Dominance Tree - Concept for reducing the test cases.

b) Proposed Metaheuristic - Genetic-Particle Swarm Combined Algorithm (GPSCA) which is used to generate automatic test data.

8.1 (a) Proposed Technique

This technique applies the concepts of dominance relations between nodes to reduce the cost of software testing.

- **Dominance Tree**

Before discussing the concept of Dominance Tree, the concepts about Control Flow Graph (CFG) is must.

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. There are two specially designated blocks, the entry block through which control enters into the flow graph and the exit block through which all control flow leaves.

Dominator 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 n0, called the root, is the head of no edge; every node n except the root n0 is a head of just one edge and there exists a (unique) path (dominance path dom (n)) from the root n0 to each node n. The root node dominates all nodes. The dominator tree is an ancillary data structure depicting the dominator relationship. This graph is a tree, since each node has a unique immediate dominator. Figure 8 show dominance tree.

DOI: 10.18535/ijecs/v5i12.64

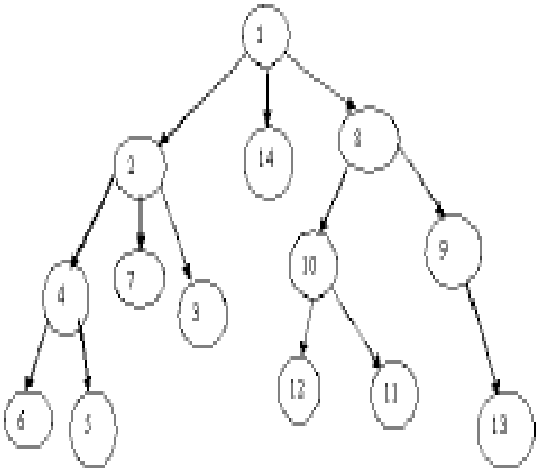

Figure 8: Dominance Tree

8.1 (b) Proposed Metaheuristics

It has been commonly accepted that finding optimality to NP hard problems is not a viable option since large amount of computational time is needed for judgment of such solutions. In reality, a good initial solution can be obtained by a heuristic/ metaheuristic in a reasonable computational time with less number of test cases.

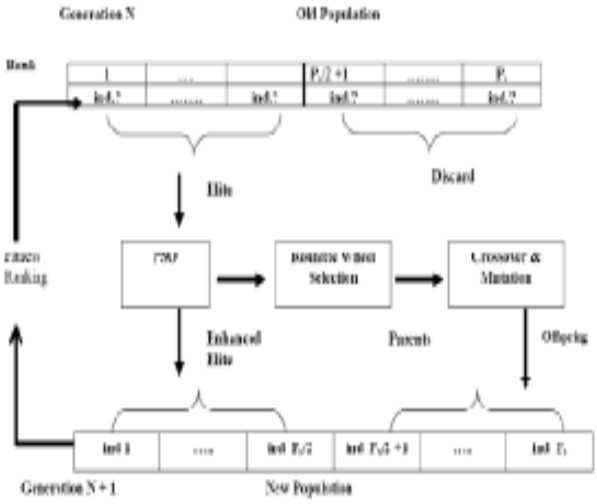

Figure 9(a): Proposed GPSCA Techniques

- **Genetic-Particle Swarm Combined Algorithm (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. The proposed GPSCA consists of three major operators: Enhancement, Crossover and Mutation..

The overall processing of GPSCA is shown in figure 9(b). GPSCA accepts list of parameters from the input file **gpsca.run** and produce the report in the output file **gpsca.out**.

Figure 9(b): GPSCA Process Diagram

9. RESEARCH METHODOLOGY

The proposed software is developed using C language. First test object source code is fed to program for instrumentation. The instrumented program provides the information for the covered node. Then, the instrumented program is fed to proposed search algorithms. The proposed search algorithm technique besides this instrumented program also accepts a file containing information about leaves of dominance tree along with dominance paths as inputs. It accepts other parameters like population size, length of chromosomes, maximum number of generations, and probabilities of the crossover and mutation operators in case of GA. The parameters in case of PSO are like number of agents, maximum iteration, dimensions etc. The algorithm produces a set of test cases, the set of nodes covered by these test cases and the list of uncovered nodes, if any, the list of distinct test cases, the list of test cases that covered dominance nodes, the coverage percentage etc. in the result file. The algorithm selects, one at a time, an uncovered node of the set of leaves nodes of the dominance tree and evolves the initial test data until the required test data are obtained or the maximum number of generations is achieved. Whenever a node is covered, the test case that caused this coverage is stored in an array list. The technique checks the coverage of remaining uncovered nodes by the generated test data that cover the current node. The whole process is shown in figure 10.

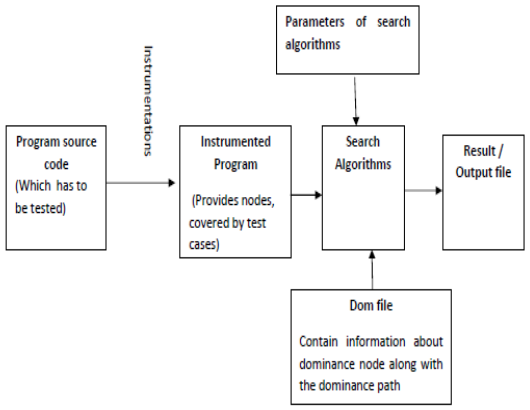

Figure 10: Building Blocks of Automatic Test Data Generator Program

Instrumentation: for the purpose of instrumentation, a program **instrument.h** is used which is called by **main1.c**; the **main1.c** demands the source code of the program which has to be instrumented and gives an instrumented program named as **prog.h.** The **prog.h** is used to provide the covered node information followed by test case when program is executed with the given test case. This information is used by the search algorithm to obtain the fitness function. The layout of instrumentation is given in figure 11.

DOI: 10.18535/ijecs/v5i12.64

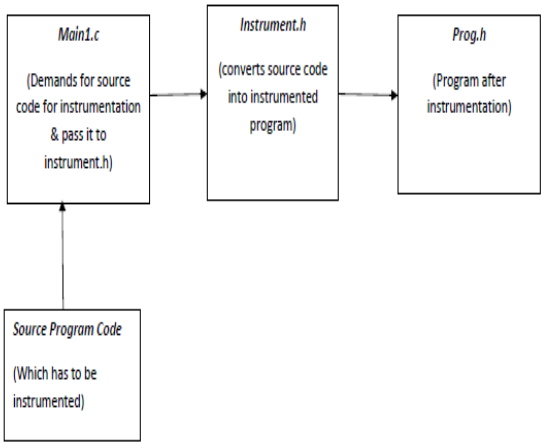

Figure 11: Steps of Instrumentation

The coding of the **ss4.c** (greatest of three numbers which is to be tested) is shown as in figure 12. The output of the instrumented program i.e. **prog.h** for the program greatest of three number stored in the file **ss4.c** is shown in figure 13

```
#include <stdio.h>
void greatest ( float x, float y, float z)
 {

 float mid;

 mid = z;
 if(y<z)
 {
     if(x<y)
     {
      mid = y;
     }
     else
     {
           if(x<z)
          {
           mid = x;
           }
     }
 }
else
 {
      if(x>=y)
      {
      mid = y;
      }
      else
     {
           if(x>z)
           {
           mid = x;
           }
      }
 }


// printf("Middle value= %d",mid);
 }
```

Figure 12: Source code greatest of three number program

Figure13.shows after executing the main program, it demands, the name of the program which has to be instrumented. After which the program shows the code of source program which is shown as first block of lines from 0 to 34 in the figure.

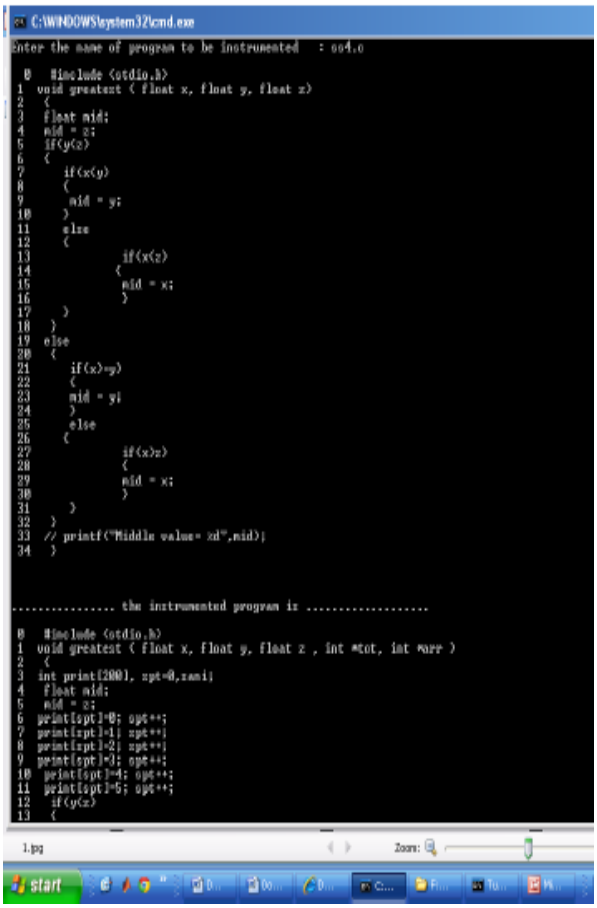

Figure 13: Program after instrumentation

10. RESULTS AND DISCUSSIONS

In the present work, Genetic Algorithm (GA), Particle Swarm Optimization (PSO) and Genetic-Particle Swarm Combined Algorithm (GPSCA) have been developed for fitness function which is based on dominance relation between two nodes. The fitness function based on the criteria of data flow coverage. The evaluation of all the algorithms implemented has been carried out using personal computer Pentium dual Core processor with 1 GB RAM. Computational analysis has been done among GA, PSO and GPSCA with the same stopping criteria i.e. either 100% coverage of all dominance nodes or maximum number of iterations/ generations achieved. This allows a fair comparison among different algorithm.

- **COMPARATIVE ANALYSIS OF GA, PSO and GPSCA**

The main aim of the research is to prove the usefulness and utility of these algorithms towards solving the ins and outs of testing objectives. The fitness function that is taken in this research is depending on the dominance relation between nodes of data flow graph. The main goal of research is to combine the power of two algorithms GA and PSO. It proves its power and effectiveness towards solving the testing problems. Our technique takes the instrumented version of the program under test, the dominator tree as inputs and returns the total number of generation, total number of test cases, total number of distinct test cases and cover ratio percentage. Initial population for GA, PSO and GPSCA program is generated

randomly. Experiments are conducted 10 times for averaging results. In each attempt, search functions are iterated for sufficient number of generations for each of ten runs. For experimental results are analyzed by changing the size of populations with different mutation and crossover probability in GA and GPSCA. In case of PSO and GPSCA also the results are determined by changing the number of individuals. A particle in case of PSO and a chromosome or individual in GA is treated as a test case.

A test case may be a solution of the problem that is under consideration or may not be. If a test case provides a perfect solution then it is called a valid test case otherwise it is called invalid test case. The solution of the problem is found when search algorithm achieves 100 percent coverage. If it not achieves the 100 percent coverage in defined number of generations / iteration then search algorithms fails. For experiment analysis seven programs for test data generation activity have been taken which are frequently used by researchers.

The brief discussions of these seven programs are as follow.

- **GTN** – This is program for finding greatest of three numbers, this is a simple program which accepts three inputs and return the greatest among three. But for research point of view program is very important as it contains nested if conditions and thus helps to check the effectiveness of search algorithms.
- **PRIME** – this is the simplest program check whether a given number is Prime number or not.
- **RM** – This program is loop based program which finds remainder of two integer numbers.
- **BS** - this program is based on Bubble Sorting algorithm which arranges the elements of array in ascending order. This program is unique in the sense that it is the only program in this set of programs which has nested loops.
- **QE** – this programs finds the roots of Quadratic Equation
- **MM** – this is array and loop based program that find minimum and maximum value from an array.
- **HCF** – this program accepts two inputs and finds Highest Common Factor between these two integer numbers. This program also contains loops.

The table 2 shows more characteristics about these programs like line of code (LOC), Cyclomatic Complexity (CC).

Table 2: Program Characteristics

Sr. No.	Program Name	LOC	CC
1	GTN	36	06
2	PRIME	16	03
3	RM	35	10
4	BS	21	04
5	QUAD	24	06
6	MINMAX	27	04
7	HCF	11	04

The proposed GPSCA is performed on these set of programs and compare it with the GA and PSO with the same set of programs to demonstrate its effectiveness in achieving the dominance tree leave nodes coverage in less number of generations. The results of the proposed GPSCA are compared with GA and PSO. As shown in figures 14 GPSCA performs better than GA and PSO in number of generation. From figure 15, it can be analyzed easily that number of test cases generated in case of GPSCA is less than GA and PSO because GPSCA achieves 100 percent coverage in less number of iterations.

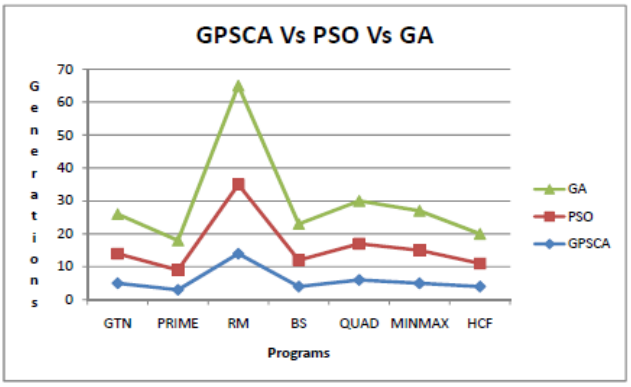

Figure 14: Comparisons of GPSCA, PSO and GA w. r. t. Number of Generations

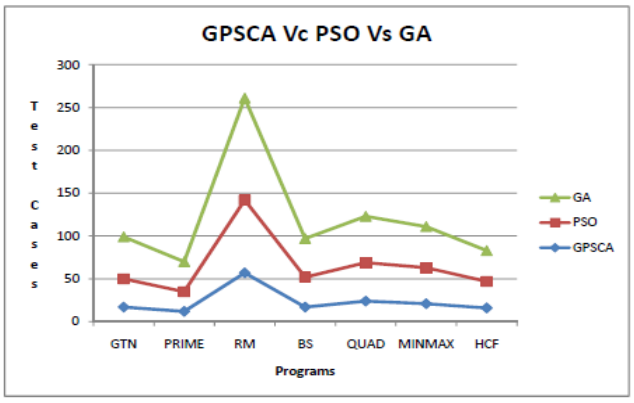

Figure 15: Comparisons of GPSCA, PSO and GA w. r. t. Number of Test Cases

The ratio of test cases and unique test cases generated by search algorithm is shown in figure 16. Ratio = Total Test cases / Unique Test Cases. The figure shows that this Ratio is worst in case of GA and the proposed GPSCA is more effective as compare to PSO in six out of seven programs.

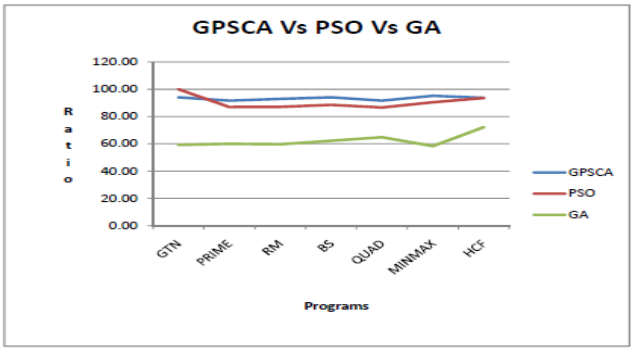

Figure 16: Comparisons of GPSCA, PSO and GA w. r. t. Ratio of Total Cases and Number of Unique Test Cases Generated

11. CONCLUSIONS

Some of the major findings from the present research work as follows:

1. The fitness function proposed for software testing confirms to be an effective. It effectively reduces the number of test cases required to achieve 100% coverage.
2. Performance of proposed GPSCA show better results as compared to GA and PSO alone.
3. The main advantage of GPSCA is that it provides a quick solution in less number of iterations.
4. The experiments were carried out on seven real world programs. The results confirm the effectiveness of proposed GPSCA as compared to GA and PSO.
5. The experiments show that proposed GPSCA performs better in generation of distinct test case as compared to GA and PSO search algorithms.
6. Proposed hybrid GPSCA has given excellent results for each of the program. Its performance in small as well as large domains shows that it has both types of search capabilities, local as well as global for fulfilling testing requirements. It is suitable for simple and complex problems. Therefore, proposed hybrid GPSCA is strongly recommended for the problem of test case generation.

12. FUTURE SCOPE OF RESEARCH WORK

Major findings and conclusions have been reported in the dissertation. However, it can be extended to give more efficient performance. One major aspect of future work is to develop new crossover and mutation processes. These should be efficient so that proposed search algorithm may be easier, efficient and faster. The instrumentation of the Software Under Test is done to a large extend but it can be further enhanced. So that it can produce control flow graph and dominance tree automatically and provides dominance tree leave nodes information with path details. A new approach with this proposed GPSCA may be possible to use this algorithm in parallel because of one of its component i.e. GA. The population can be split into several smaller sub populations. Depending on the method these sub populations can form a new population. The parallel GPSCA may be fruitful in searching of results because these sub populations can explore the search space more thoroughly.

REFRENCES

[1] K. Li, Z. Zhang and J. Kou, "Breading software test data with genetic-particle swarm mixed algorithms", Journal of Computers, Vol. 5, No. 2, pp. 258-265, 2010.
[2] B. Beizer, "Software testing techniques", Second Edition, Van Nostrand Reinhold, NewYork, 1990.
[3] R. A. DeMillo and A. J. Offlut, "Constraint-based automatic test data generation", IEEE Transactions on Software Engineering, Vol. 17, No. 9, pp. 900-910, 1991.
[4] S. Desikan and G. Ramesh, "Software testing principles & practices", Pearson Education, 2007.
[5] R. Boyer, B. Elspas and K. He Levitt, "Select-a formal system for testing and debugging programs by symbolic execution", SIGPLAN Otices, Vol. 10, No. 6, pp. 234-245, June 1975.
[6] L. Clarke, "A system to generate test data and symbolically execute programs", IEEE Transaction on Software Eng., Vol. SE-2, No. 3, pp. 215- 222, Sept. 1976.
[7] C. Ramamoorthy, S. Ho and W. Chen, "On the automated generation of program test data", IEEE Trans. Software Eng., Vol. SE-2, no. 4. pp. 293-300, Dec. 1976.
[8] W. Howden, "Symbolic testing and the DISSECT symbolic evaluation system", IEEE Trans. Software Eng., Vol. SE- 4, No. 4, pp. 266- 278, 1977.
[9] J. H. Holland, "Adaptation in natural and artificial system", MIT Press, The university of Michigan press, 1975.
[10] D. Ince, "The automatic generation of test data", Computer Journal, Vol. 30, No. 1, pp. 63 69, 1987.
[11] W. Miller and D. Spooner, "Automatic generation of floating-point test data", IEEE Trans. Software Eng., Vol. SE-2, No. 3, pp. 223-226, Sept. 1976.
[12] R. L. Haupt and S. E. Haupt, "Practical genetic algorithms", John Wiley & Sons publication.ll.
[13] S. Mitra and T. Acharya "Data mining multimedia, soft computing, and bioinformatics", John Wiley & Sons publication.
[14] L. C. Jain and N. M. Martin, "Fusion of neural networks, fuzzy systems and genetic algorithms: industrial applications", CRC Press, 1998.
[15] A. K. Dhingra, "Multi-objective flow shop scheduling using metaheuristics", Ph. D. thesis, NIT, Krukshetra, 2011.
[16] D. E. Goldberg, "Genetic algorithms in search, optimization & machine learning", Addison-Wesley, Reading, 1989.
[17] H. Singh, "Automatic generation of software test cases using genetic algorithms", M Tech. Thesis, TIET, Patiala, Punjab, 2004.
[18] J. Kennedy and R. Eberhart, "Particle swarm optimization", IEEE International Conference on Neural Networks,IEEE Press, pp. 1942–1948, 1995.
[19] A. S. Ghidk, "A new software data-flow testing approach via ant colony algorithms", Universal Journal of Computer Science and Engineering Technology, Vol. 1, No. 1, pp. 64-72, Oct. 2010.
[20] P. R. Srivastava, N. Jose, S. Barade and D. Ghose, "Optimized test sequence generation from usage models using ant colony optimization", IJSEA, Vol. 1, No. 2, 2010.
[21] M. Dorigo, V. Maniezzo and A. Colorni, "Ant system: optimization by a colony of cooperating agents", IEEE Transactions on Systems, Man, and Cybernetics, Vol. 26, No. 1, pp. 29-41, 1996.
[22] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey", Theoretical Computer Science, Vol. 344, No. 2, pp. 243-278, 2005.

AUTHOR PROFILE

Suraj Indiver
Master of Technology Student, Department of Computer Science and Engineering, Sachdeva Institute of Technology, Mathura, Dr. A.P.J. AKTU Lucknow. MCA from Maharishi Dayanand University, Rohtak in 2013.