ISSN No. 0976-5697

Volume 2, No. 2, Mar-Apr 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

A Combination of GA and PSO for Automatic Test Data Generation using Data Flow Coverage

Sanjay Singla*
Ph D Scholar
Suresh Gyan Vihar University
Jaipur,India
san_jay23@yahoo.com

H M Rai
Electronics & Communication,
NC College of Engineering
Panipat,India
hmrai1943@gmail.com

Priti Singla
Ph D Scholar
Suresh Gyan Vihar University
Jaipur,India
pritisingla04@gmail.com

Abstract:. Software testing plays an important role for software's quality and reducing the cost. In this paper we introduce a new algorithm that combine the power of Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) called Genetic-Particle Swarm Combined Algorithm (GPSCA) which is used to generate automatic test data that satisfy data-flow coverage criteria. Finally, the paper presents the results of the experiments that have been carried out to evaluate the effectiveness of the proposed GPSCA with new fitness function compared to the Genetic Algorithm and PSO algorithms.

Keywords: Genetic Algorithms, Automatic test data generation, data flow testing, Particle Swarm Optimization.

I. INTRODUCTION

In software testing, Automatic test data generation plays a very vital role important role as it significantly reduce the time and cost of software. Software Testing is the process of exercising the software product in pre-defined ways to check if the behavior is the same as expected behavior [2]. The main objectives of testing are to provide quality products to customers. There are many methods propose by many researchers from time to time [4]-[7], [9] -[12].

Recently, the use of genetic algorithms (GAs) in test data generation became the focus of several research studies. Girgis[13] has proposed a technique that uses GA which is guided by the data flow dependencies in the program, to search for test data to fulfill data flow path selection criteria namely the all-uses criterion.

However, GA has started getting competition from other heuristic search techniques, such as the particle swarm optimization. Various works [16]-[20] show that particle swarm optimization is equally well suited or even better than genetic algorithms for solving a number of test problems [21].

This paper presents a new algorithm that combine the power of Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) called Genetic-Particle Swarm Combined Algorithm (GPSCA) which is used to generate automatic test data that satisfy data-flow flow dependencies in the program, to search for test data to fulfill one of the most demanding in the family of data flow path selection criteria, developed by Rapps and Weyuker [14], namely the all-uses criterion In this paper. Further, this paper show the effectiveness of the proposed GPSCA with new fitness function compared to the Genetic Algorithm and PSO algorithms.

II. BACKGROUND

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

A. 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. figure. 2. shows the control flow graph G of the example program, which is shown in figure. 1.

B. Data flow analysis 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 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. However, empirical evidences show that the all-uses criterion is the most effective criterion compared to the other data flow criteria. All-uses criterion requires that test data be included which causes the traversal of at least one sub-path from each variable definition to every p-use and every c-use of that definition. The all-uses criterion requires a def-clear path from each def of a variable to each use (c-use and p-use) of that variable to be traversed. A def-clear path is a path from definition node u to use node v or edge (v, t) where variable is not redefined [27].

III. PROPOSED WORK

```
0    1    program test;
1    1    var I, j, k : integer ;
2    1         begin
3    1         i := 0 ;
4    1         j := 0 ;
5    1         read( k ) ;
6    2         while (k <> 0) do
7    2         begin
8    3         if ( k mod 2 ) = 0
9    4         then i := i + 1
10   5                   else
11   5         j := j +1 ;
12   6         read( k )
13   6         end ;
14   7         write( i , j );
      15   7          end .
```

Figure. 1 Program 1

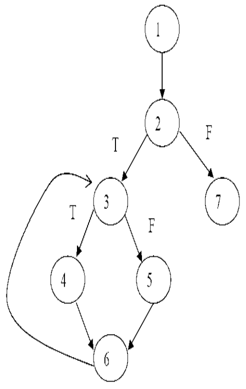

Figure. 2 Control Flow Graph

Table 1. List of the dcu-paths of the example program given in figure 1

DCU-Path No.	Variable	Def Node	C-use node
1	I	1	4
2	J	1	5
3	I	1	7
4	I	4	7
5	J	5	7
6	J	1	7

A. GPSCA

The combination of GA and PSO always performs better than GA or PSO alone [29-30]. The proposed GPSCA

we introduce a new algorithm that combine the power of Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) called Genetic-Particle Swarm Combined Algorithm (GPSCA) which is used to generate automatic test data that satisfy data-flow coverage criteria.

consists of three major operators: enhancement, crossover, and mutation.

Table 2. List of the dpu-paths of the example program given in figure 1

DPU-Path No.	Variable	Def Node	P-use node
1	K	1	2-3
2	K	1	2-7
3	K	1	3-4
4	K	1	3-5
5	K	6	2-7
6	K	6	2-3
7	K	6	3-4
8	K	6	3-5

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.

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 Pm = 0.1 is used by us.

B. Fitness function

We have a new evaluation function to find optimum set of test cases. The fitness function takes into account not only the number of the covered DU-Paths, but also how effective is a chromosome compared to the rest of the chromosomes in the current population and with respect to the set of paths at hand [22]. The fitness function used is mathematically expressed as

$$Eval(vi) = \frac{\text{no. of def} - \text{use paths covered by vi}}{\text{total no of def} - \text{use paths}} + \frac{\text{No. of newly covered path}}{\text{Path not yet covered}} \quad (1)$$

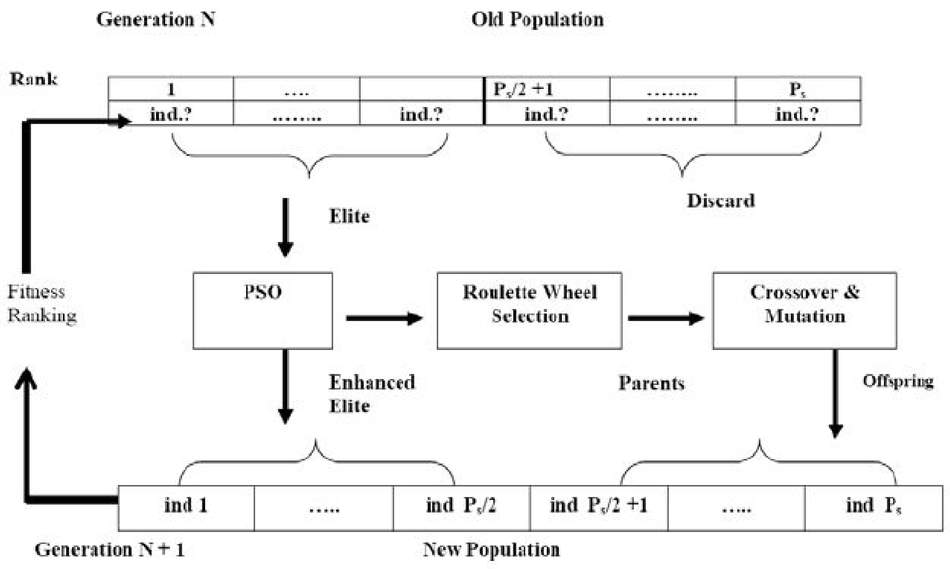

Figure. 3 GPSCA

The number returned from the first part of eq.(1) is the most common in literature and shows the percentage of coverage for the gene considering the total number of paths that have to be covered according to the selected criterion. Therefore, we resort to using additional parts in eq.(1). The second parts of the fitness function may be considered a kind of reward to the chromosome. These are introduced based on the observation that a chromosome which only covers already covered paths (in a former or the current generation) is not really a step towards optimization.

The second part of the fitness function gives the true contribution of the chromosome in terms of covering the targeted paths. The percentage of the paths covered for the first time by the chromosome under investigation is more important than the total number of paths covered.

IV. Experimental Results

MATLAB programming is used for implementing algorithms for experimentation purpose. The main goal of research is to combine the power of two algorithms (GA and PSO) and prove its power and effectiveness towards solving the testing problems. 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.

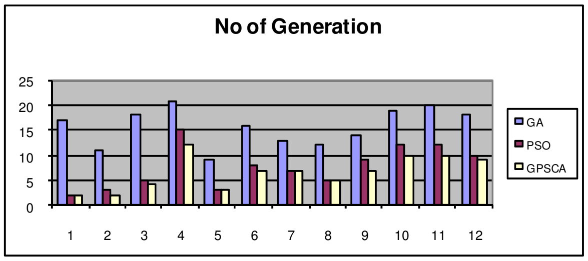

Figure.4 Comparison of number of Generations by GPSCA, GA and PSO

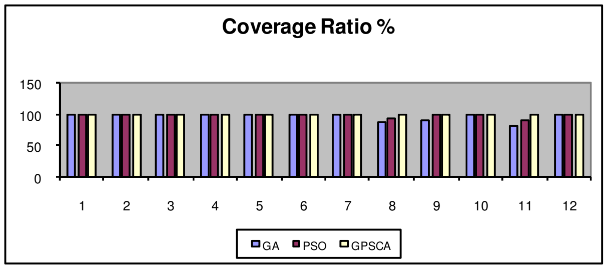

Figure .5 Comparison of number of coverage ratio % by GPSCA, GA and PSO

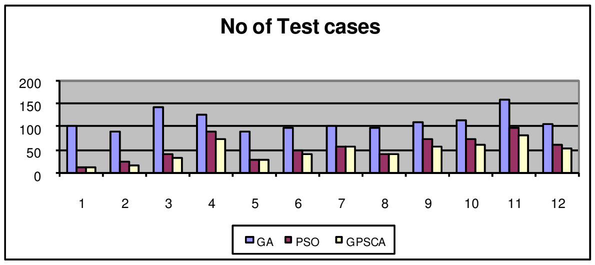

Figure.7 Comparison of number of test cases generated by GPSCA, GA and PSO

V. CONCLUSION AND FUTURE WORK

The results of our new approach GPSCA is better than GA and PSO as in some cases it has higher coverage ratio % than the PSO and GA. Also test case requirement by GPSCA is less than both two techniques (GA and PSO). Our experiment also demonstrates the effectiveness of our proposed approach in case of number of generations, as GPSCA require less generation than PSO and GA.

Our future work will be to study the test case generation using hybrid GA and ACO (Ant colony optimization) and compare its effectiveness with our GPSCA approach for data flow testing using dominance concept.

VI. REFERENCES

[1] B. Beizer, “Software Testing Techniques”, Second

Edition, Van Nostrand Reinhold, NewYork, 1990.

[2] 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.

[3] S. Desikan, G. Ramesh, “Software testing principles & practices”, Pearson Education.

[4] R. Boyer, B. Elspas, and K. Levitt, “Select-a formal system for testing and debugging programs by symbolic execution”, SIGPLAN Otices, Vol. 10, No. 6, pp. 234-245, June 1975.

[5] 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.

[6] 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.

[7] W. Howden, “Symbolic testing and the DISSECT symbolic evaluation system”, IEEE Trans. Software Eng., Vol. SE-4, No. 4, pp. 266- 278, 1977.

[8] J. H. Holland, “Adaptation in natural and artificial system, Ann Arbor”, The University of Michigan Press, 1975.

[9] D. Ince, “The automatic generation of test data”, Computer Journal, Vol. 30, No. 1, pp. 63-69, 1987.

[10] 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.

[11] J. Offutt, Z. Jin, and J. Pan, “The Dynamic domain reduction procedure for test data generation”, Software Practice and Experience, Vol. 29, No. 2, pp. 167–193, January 1997.

[12] N. Gupta, A. P. Mathur, and M. L. Soffa, “Automated test data generation using an iterative relaxation method”, In ACM SIGSOFT Sixth International Symposium on Foundations of Software Engineering (FSE-6) Orlando, Florida, pp 231–244, November 1998.

[13] M. R. Girgis, “Automatic test data generation for data flow testing using genetic algorithm”, Journal of Universal Computer Science, Vol. 11, No. 6, pp. 898–915, 2005.

[14] S. Rapps and E. J. Weyuker, “Selecting software test data using data flow information”, IEEE Transactions on Software Engineering, Vol. 11, No. 4, pp. 367-375, 1985.

[15] F. E. Allen and J. Cocke, “A program data flow analysis procedure”, Communication of the ACM, Vol. 19, No. 3, pp. 137–147, 1976.

[16] J. Kennedy and R. Eberhart, ”Particle swarm optimization”, IEEE International Conference on Neural Networks, IEEE Press, pp. 1942–1948, 1995.

[17] A. Windisch, S. Wappler and J. Wegener, “Applying paricle swarm optimization to software testing”, ACM, GECCO, London, England, United Kingdom, New York, pp. 1121-1128, 2007.

[18] K. Agrawal and G. Srivastava, “Towards software test data generation using discrete quantum particle swarm optimization”, ISEC, Mysore, India, pp. 65-68, February 2010.

[19] A. Li and Y. Zhang, “Automatic generating all-path test data of a program based on pso”, World Congress on Software Engineering. IEEE, Los Alamitos, pp. 189-193, 2009.

[20] R. C. Eberhart and J. Kennedy, “A new optimizer using particle swarm theory”, 6th International Symposium on Micromachine Human Science, pp. 39–43, 1995.

[21] N. Narmada and D. P. Mohapatra, “Automatic Test Data Generation for data flow testing using Particle Swarm Optimization”, Communications in Computer and Information Science , Vol. 95, No. 1, pp. 1-12, 2010.

[22] A. S. Andreou, K. A. Economides and A. A. Sofokleous, “An automatic software test-data generation scheme based on data flow criteria and genetic algorithms”, 7th IEEE International Conference on Computer and Information Technology, pp. 867-872, 2007.

[23] R. P. Pargas, M. J. Harrold and R. R. Peck, “Test Data Generation using Genetic Algorithms”, Software Testing Verification and Reliability, Vol 9, pp. 263-282, 1999.

[24] C. C. Michael, G. E. McGraw and M. A. Schatz, ”Generating software test data by evolution”, IEEE Transactions on Software Engineering, vol. 27, no.12, pp. 1085-1110, 2001.

[25] A. S. Ghiduk, M. J. Harrold and M. R. Girgis, “Using Genetic Algorithms to Aid Test-Data Generation for Data-Flow Coverage”, 14th Asia-Pacific Software Engineering Conference, 2007.

[26] J. T. Alander, T. Mantere, and P Turunen, “Genetic Algorithm Based Software Testing”, In Proceedings of International Conference (ICANNGA97), Wien, Austria, pp. 325-328, April 1998.

[27] A. Khamis, R. Bahgat and R Abdelaziz, “Automatic test data generation using data flow information”, Dogus University Journal, 2, pp. 140-153, 2000.

[28] A. S. Ghiduk and M. R. Girgis, “Using Genetic Algorithms and dominance concepts for generating reduced test data”, informatics, 34, pp. 377-385, 2010.

[29] K. H. Chang, J. H. Cross, W. H. Carlisle and D. B. Brown “ A framework for intelligent test data generation”, journal of intelligent and robotic systems- theory and application, Vo. 5, No. 2, pp. 147-165, 1992.