A Hybrid PSO Approach to Automate Test Data Generation for Data Flow Coverage with Dominance Concepts

Sanjay Singla[1], Dharminder Kumar[2], H M Rai[3] and Priti Singla[1]

[1] Department of CSE, Suresh Gyan Vihar University, Jaipur, Rajasthan, India
[2] Department of Computer Science & Engineering, GJUST, Hisar, Haryana, India
[3] Department of ECE, N C College of Engineering, Israna, Panipat, Haryana, India
san_jay23@yahoo.com

Abstract

This paper presents a technique that based on a combination of genetic algorithm (GA) and particle swarm optimization (PSO), and is thus called GPSCA (Genetic-Particle Swarm Combined Algorithm) which is used to generate automatic test data for data flow coverage with using dominance concept between two nodes. 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.

Keywords: *software testing, genetic algorithms, automatic test data generation, test adequacy criteria, data flow testing, particle swarm optimization, dominance concept*

1. Introduction

Software test is the main approach to find errors and defects assuring the quality of software [29]. Software testing is an expensive component of software development and maintenance [23]. Testing is a complex, labor-intensive, and time consuming task that accounts for approximately 50% of the cost of a software system development [1].

Absolutely, an automated software testing can significantly reduce the cost and time of developing software. There are many methods propose by many researchers from time to time [4]-[7], [9]-[12], [31] for automatic generation of test cases. Random test data generators select random inputs from test data from some distribution [31]. Path-oriented or structured based test data generators typically use the program's control flow graph, select a particular path and use a technique such as symbolic evaluation to generate test data for that path [2], [4-6]. Goal oriented test data generator select input to execute the selected goal, such as a statement irrespective of path taken [32]. Intelligent test data generators often rely on sophisticated analyses of the code to guide the search for new test data [33]. Recently, the use of genetic algorithms (GAs) in test data generation became the focus of several research studies. For example Pargas et al. [23] presented a genetic algorithm directed by the control-dependence graph of the program under test to search for test data to satisfy all-nodes and all-branches criteria. Michael et al. [24] discussed the use of GAs for automatic test-data generation to satisfy condition-decision test-coverage criterion. 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.

Data-flow testing is important because it augments control-flow testing criteria and concentrates on how a variable is defined and used, which could lead to more efficient and targeted test suites. Girgis [13] used the ratio between the numbers of the covered def-use paths covered by a test case to the total number of def-use paths. This technique cannot find the closeness of the test cases because the fitness function gives the same value for all test cases that cover the same number of def-use paths and ‘0’ for all test cases that do not cover any def-use path. This technique will result in a loss of valuable information (test data that contains good genes) when it ignores test cases that cover only the use node [25].

However, GA has started getting competition from other heuristic search techniques, such as the particle swarm optimization. Like GA, PSO is initialized with a population of random solutions. Its development was based on observations of the social behavior of animals such as bird flocking, fish schooling, and swarm theory. 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. Compared with GA, PSO has some attractive characteristics. It has memory, so knowledge of good solutions is retained by all particles; whereas in GA, previous knowledge of the problem is destroyed once the population changes. It has constructive cooperation between particles, particles in the swarm share information between them [30]. 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].

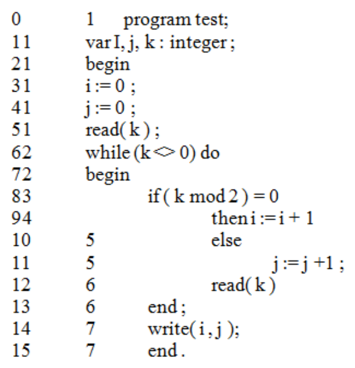
This paper presents a new technique GPSCA- combination of genetic and particle swarm algorithm to automatic test-data generation for searching test data to satisfy the data-flow coverage criteria with using fitness function that evaluates the fitness of test data based on its relation, through dominance, to the definition and use in the data-flow requirement. The paper also presents a set of empirical studies that show the effective of our test-data generation technique in achieving coverage (generating test data that cover the test requirements), reducing the number of test cases.

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



```
0       1    program test;
11      var I, j, k : integer;
21      begin
31      i := 0 ;
41      j := 0 ;
51      read( k ) ;
62      while (k <> 0) do
72      begin
83              if ( k mod 2 ) = 0
94                      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. Example program (The 1st column represents statement numbers, and the 2nd one represents block numbers)

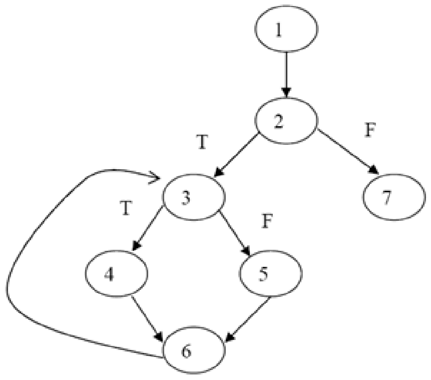


Figure 2. Flow graph for the example program given in figure 1

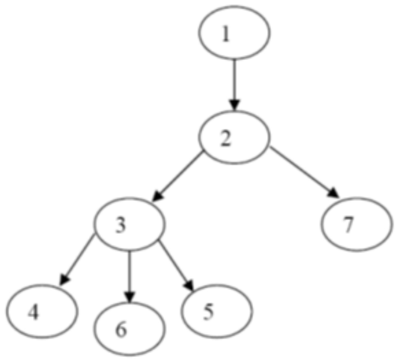


Figure 3. Flow graph for the example program given in figure 1

2.2. Dominance Tree

For G = (V, E), a directed graph with two distinguished nodes n_0 and n_k, 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 no 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. 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 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. 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].

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

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

3. GA and PSO

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]. In this section, basic concepts of GAs and PSO are introduced, followed by a detailed introduction of GPSCA.

3.1. Genetic Algorithms

In GA, a candidate solution for a specific problem is called an individual or a chromosome and consists of a linear list of genes. Each individual represents a point in the search space, and hence a possible solution to the problem. A population consists of a finite number of individuals. Each individual is decided by an evaluating mechanism to obtain its fitness value. Based on this fitness value and undergoing genetic operators, a new population is generated iteratively with each successive population referred to as a generation. The GAs use three basic operators (reproduction, crossover, and mutation) to manipulate the genetic composition of a population. Reproduction is a process by which the most highly rated individuals in the current generation are reproduced in the new generation. The crossover operator produces two off-springs (new candidate solutions) by recombining the information from two parents. There are two processing steps in this operation. In the first step, a given number of crossing sites are selected uniformly, along with the parent individual at random. In the second step, two new individuals are formed by exchanging alternate pairs of selection between the selected sites. Mutation is a random alteration of some gene values in an individual. The allele of each gene is a candidate for mutation, and its function is determined by the mutation probability [30].

A basic pseudo algorithm for a GA is as follows:

```
Initialize (population)
Evaluate (population)
While (stopping condition not satisfied) do
{
Selection (population)
Crossover (population)
Mutate (population)
Evaluate (population)
}
```

3.2. Particle Swarm Optimization (PSO)

Particle Swarm Optimization is an algorithm developed by Kennedy and Eberhart [16] 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. In the basic PSO technique, suppose that the search space is d-dimensional.

1. Each member is called particle, and each particle (i-th particle) is represented by d-dimensional possitional vector and described as $X_i = [x_{i1}, x_{i2}, ..., x_{id}]$

2. The set of n particles in the swarm is called population and is described as

3. The best previous position for each particle (the position giving the best fitness value) is called particle best and described as. $PB_i = [pb_{i1}, pb_{i2}, ..., pb_{id}]$

4. The best position among all of the particle best position achieved so far is called global best and described as. $GB_i = [gb_{i1}, gb_{i2}, ..., gb_{id}]$

5. The rate of position change for each particle is called the particle velocity and described as. $V_i = [v_{i1}, v_{i2}, ..., v_{id}]$

At iteration k the velocity for d-dimension of i-th particle is updated by:

$$v_{id}\left(k+1\right) = wv_{id}(k) + c_1 r_1 (pb_{id}(k) - x_{id}(k)) + c_2 r_2 (gb_{id}(k) - x_{id}(k)) \quad (1)$$

Where i= 1,2,..,n and n is the size of population, w is the inertia weight, c_1 and c_2 are the acceleration constants, and r_1 and r_2 are two random values in range [0,1].

6. The i-particle position is updated by:

$$x_{id}(k+1) = x_{id}(k) + v_{id}(k+1) \quad (2)$$

Based on (1) and (2), the population of particles tends to cluster together with each particle moving in a random direction. The computation of PSO is easy and adds only a slight computation load when it is incorporated into GA.

3.3. GPSCA

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 offsprings 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) and (2) may be applied to the elites.

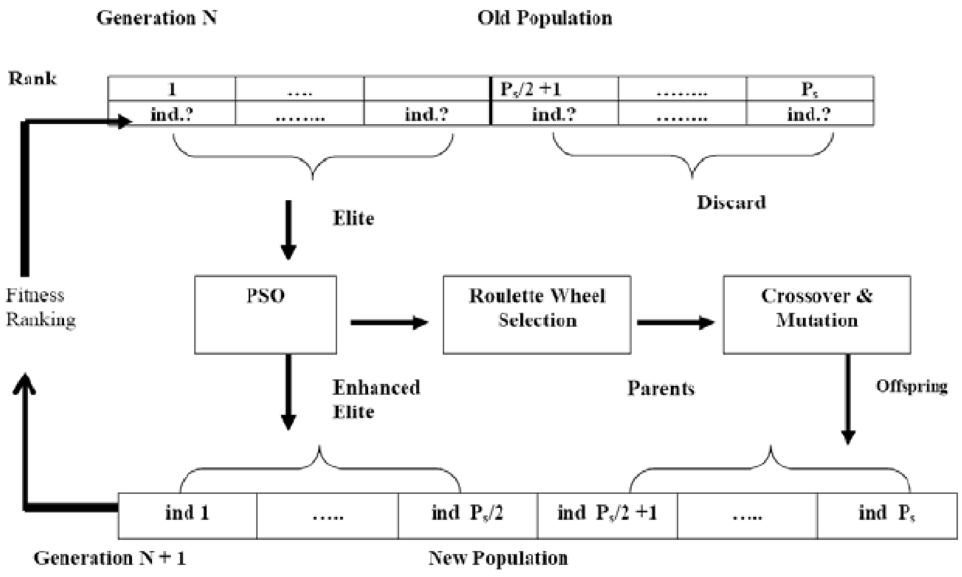


Figure 4. GPSCA

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 offsprings 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.

4. Fitness Function

This function considers the du-pair as two objectives. For the definition-computational use (dcu) such as (d,u,v) the first objective is the def node d and the second objective is the use node u of variable v. For the definition predicate use (dpu) such as (d, (t, h), v) the first objective is the def node d and the second objective is the head node h, where there is a predicate use of variable v at edge (t, h).

To find the closeness, our technique (NTDG – New Test Data Generation based on PSO/ GA) adds a value called closeness level (CL) to the fitness values of the test cases that missed any node of the dominance paths of the two objectives. CL depends on two hypotheses: the first hypothesis is that test cases that cover the definition node is closer than test cases that do not cover any one of the target nodes or cover the use node only; the second hypothesis is that test cases that miss one of the target nodes and try again to cover this target are closer than test cases that miss the target node and do not try again to cover it. Proposed technique NTDG uses equation (4) to calculate the closeness level.

To evaluate a test case NTDG executes the program under test with it as input, and records the traversed nodes that are covered by this test case and belong to the dominance paths of the two objectives. Then, the fitness value ft(d, u, v_i) for a dcu-pair (d, u, var), where d and u are the def and c-use nodes of variable var, respectively or dpu pair (d, (t, u), var), where d is the def and (t,u) is p-use edge of variable var and v_i (i = 1, . . ., PopSize) is an individual of the current population, is calculated as follows:

1. Find dom(d) and dom(u): the dominance paths of the def node and the use node.
2. Add the def node d to dom(u) to give high priority to cover def before use.
3. Execute the program under test using the generated test data and determine cdom(d) and cdom(u) the covered nodes of dom(d) and dom(u), respectively.
4. Compute udom(d V u): the not covered nodes of dom(d) or dom(u).
5. Find fdom(d V u): the nodes of dom(d) or dom(u) that are covered more than one time. Then, we calculates the fitness value using the following

$$\mathrm{ft(d,u,v_i)}=\frac{1}{2}\times\left(\frac{|\mathrm{cdom(d)}|}{|\mathrm{dom(d)}|}+\frac{|\mathrm{cdom(u)}|}{|\mathrm{dom(u)}|}\right) \qquad (3)$$

Where |A| gives the cardinality of A.

The CL is calculated by the following formula:

$$CL=\frac{1}{2}\times\min\left(o.9,\frac{|\,fdom(dVu)\,|-|\,udom(dVu)\,|}{|\,dom(d)\,|\times|\,dom(u)\,|}\right) \qquad (4)$$

A test case is optimal (i.e., covers the target) if its fitness value ft(v_i) = 1. For example, suppose the target du-pair is the dcu (1, 4, I) and tc_1= 1, 2, 3, 5, 6, 2, 7 and tc_2= 1, 2, 3, 4, 6, 2, 3, 5, 6, 2, 7.

dom(d)=dom(1)= 1

dom(u)=dom(4)= 1, 2, 3, 4

1- $$ft(1,4,tc_1) = \frac{1}{2} \times \left(\frac{1}{1} + \frac{3}{4} \right) = 0.875$$

$$CL = \frac{1}{2} \times \min\left(0.9, \frac{1-1}{1 \times 4} \right) = 0.0$$

$$ft(1,4,tc_1) = 0.875 + 0.0 = 0.875$$

2 - $$ft(1,4,tc_2) = \frac{1}{2} \times \left(\frac{1}{1} + \frac{4}{4} \right) = 1.00$$

$$CL = \frac{1}{2} \times \min\left(0.9, \frac{3-1}{1 \times 4} \right) = 0.25$$

$$ft(1,4,tc_2) = 1.0 + 0.25 = 1.25 \sim 1$$

As $ft(1,4,tc_1) < ft(1,4,tc_2)$. Thus tc_2 is more effective than tc_1.

5. Experimental Results

Our technique takes the instrumented version of the program under test P', the dominator tree (DT) and set of test requirements (TestReq) requird for data flow as inputs and returns the total number of genration, 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 main goal of research is to combine the power of two algorithms (GA and PSO) and prove its power and effectiveness towars 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 demostrate its effectiveness in achiving the test cover ration in less number of generations.

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.

Table 3. Comparison between GPSCA, GA and PSO Techniques

Prog. No.	No. of Variables	Population Size	Coverage Ratio %			No of Test cases			No of Generation		
			GA	PSO	GPSCA	GA	PSO	GPSCA	GA	PSO	GPSCA
1	3	7	100	100	100	154	126	98	22	18	14
2	3	10	93	100	100	80	40	40	8	4	4
3	1	9	100	100	100	126	90	72	14	10	8
4	2	10	100	100	100	30	30	30	3	3	3
5	2	8	100	100	100	112	96	80	14	12	10
6	2	10	81	92	100	230	180	150	23	18	15
7	6	9	71	89	97	99	63	54	11	7	6

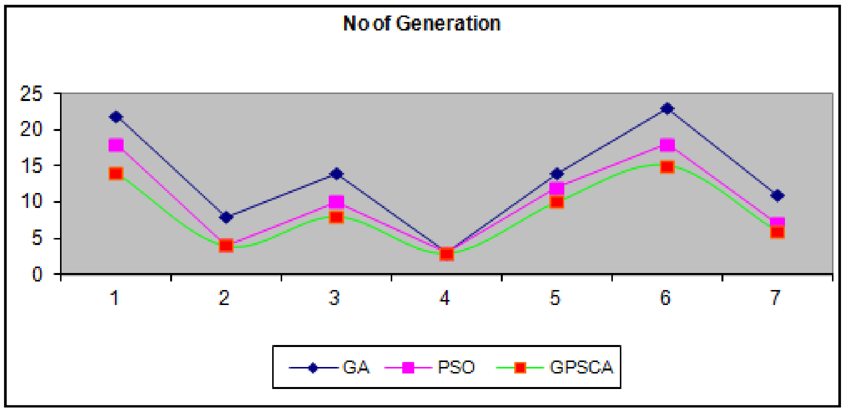


Figure 5. Comparison of number of Generations by GPSCA, GA and PSO

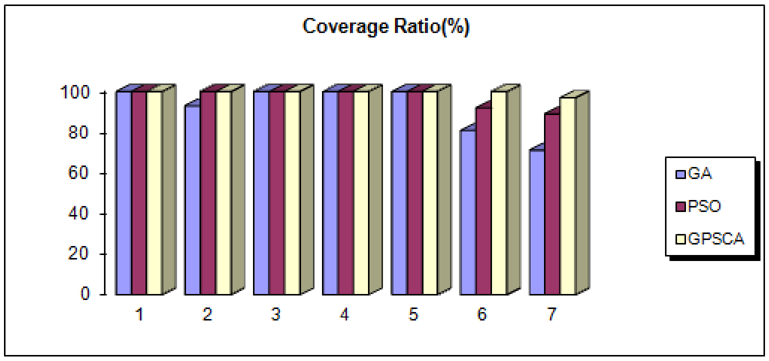


Figure 6. Comparison of number of coverage ratio % by GPSCA, GA and PSO

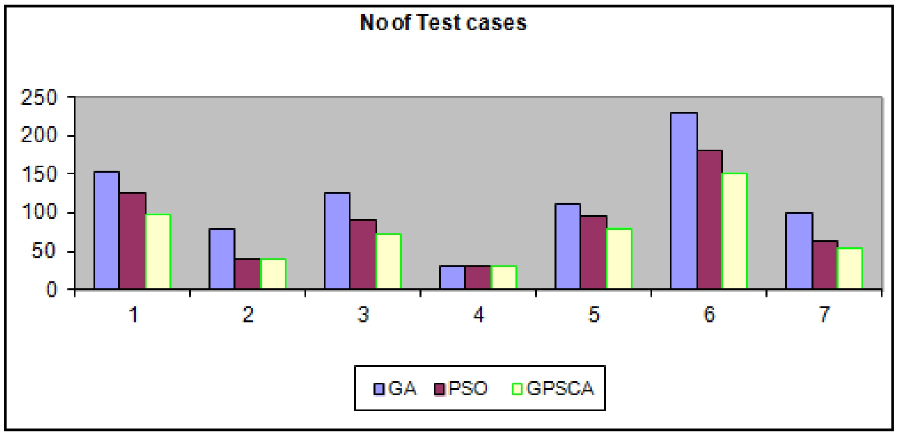


Figure 7. Comparison of number of test cases generated by GPSCA, GA and PSO

6. Conclusion and Future Work

We have developed an algorithm for generating test cases using combining the power of GA and PSO, GPSCA – Genetic Particle Swarm Combined Algorithm with a new multiobjective fitness function. Experiments have been carried out to show the effectiveness of the proposed GPSCA compared to the PSO and GA techniques. 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 demostarte the effectiveness of our proposed aproach in case of number of generations, as GPSCA require less genertaion 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 fectiveness with our GPSCA approach for data flow testing using dominance concept.

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 and G. Ramesh, “Software testing principles & practices”, Pearson Education.

[4] 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.

[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 particle 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. 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.

[30] C. Juang " A hybrid of genetic algorithm and particle swarm optimization for recurrent network design", IEEE transactions on systems management and cybernetics, Vol. 34, No. 2, pp. 997-1006, 2004.

[31] H. D. Mills, M. Dyer and R. C. Linger "cleanroom software engineering", IEEE software, Vol. 4, No. 5, pp. 19-25, 1987.

[32] B. Korel "automated software test data generation", IEEE transactions on software engineering, Vol. 11, No. 5, pp. 299-306, 1990.

[33] 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.

Authors

Sanjay Singla was born on 24th February 1978. He is M.Tech (CSE), Ph.D. (CSE) pursuing in 'Automatic generation of software test cases using soft computing'. His area of interest is Software Engineering, DBMS, Operating Systems, and Programming Languages etc. Presently he is Associate Professor and Head CSE Dept., Om Institute of Technology & Management , Hisar- 125001.

Prof. Dharminder Kumar received his Ph.D in the area of Computer Science in Computer Networks. He is recipient of Gold Medal at his Master's degree. Currently he is heading the Faculty of Engineering and Technology as Dean and also the department of Computer Science & Engineering as Chairman, GJUST, Hisar, India. He has been engaged in teaching & research since last 22 years.

Prof. H.M.Rai was born on 1st August, 1943. He received the B.Sc. Engg. (Electrical), from Punjab University in 1963. M.E. from University of Roorkee in 1966 and PhD from Regional Engineering College, Kurukshetra University in 1992. He joined as lecturer in REC, Kurukshetra in 1966. From there he retired as Professor in 2003. He has published a number of technical papers and text books. His area of interest is Electrical Drives, Management Technology, Energy Systems, Instrumentation and control. At present he is Professor at NCCE Israna, Panipat.

Priti Singla was born on 4th October, 1977. She has done M.Sc. Applied Mathematics (Gold Medalist) from Guru Jambheshwar University of Science and Technology, Hisar in 1999 and M.Tech (CSE) in 2005. She is pursuing Ph.D. under title 'Soft computing approach for meteorological weather forecasting' under the able guidance of Dr. H.M. Rai. Her area of interest is Operation Research, Statistics, Numerical Methods, Soft Comuting, Discrete Mathematics etc.