MULTI-OBJECTIVE GENETIC ALGORITHM USING CLASS-BASED ELITIST APPROACH

P. Maragathavalli [1] and S. Kanmani [2]

[1]Assistant Professor, Department of Information Technology, Pondicherry Engineering College, Puducherry
marapriya@pec.edu
[2]Professor, Department of Information Technology, Pondicherry Engineering College, Puducherry
kanmani@pec.edu

ABSTRACT

An approach named Class-Based Elitist Genetic Algorithm is presented in this paper. The test data is being generated for object-oriented programs using evolutionary techniques. A class-based algorithm derived from class control-flow graph (CCFG) is used for testing object-oriented software.

Evolutionary techniques have been used for solving most of the software engineering problems. Evolutionary testing technique which is based on theory of evolution like reproduction, mutation, recombination, and selection is used to generate test cases in CBEGA. Evolutionary Algorithm applies these techniques repeatedly to a set of individuals called population to obtain optimal solution in a global search space with minimum search time.

Multi-objective optimization involves optimizing a number of objectives simultaneously like time, cost and fault detection capability. The objectives considered here for optimization are maximum path coverage, minimum test suite size, and minimum execution time. For experiments, the data are taken from Siemens' test suite which shows better path coverage results like 96% in CBEGA and are compared with simple GA which gives only 88% path coverage for a set of sample java classes.

KEYWORDS

Multi-objective Genetic Algorithm, Search Time, CBEGA, Test-suite Reduction, Path Coverage, Elitism

1. INTRODUCTION

Evolutionary algorithms (EAs) are search methods that take their inspiration from natural selection and survival of the fittest mechanisms [3]. EAs differ from traditional optimization techniques in that they involve a search from a population of solutions, not from a single point. In an EA, involves a competitive selection that filters the least favorable solutions. The solutions with high fitness are recombined with other solutions by swapping parts of a solution with another. Solutions are also mutated by making a small change to a single element of the solution. EAs are robust optimization methods used for test data generation.

Multi-objective optimization refers to the solution of problems with two or more objectives to be satisfied at the same time [1] [3], i.e., simultaneously. Most real world problems have multiple objectives to achieve in order to get an optimal solution. This situation creates a set of problems

DOI : 10.5121/cseij.2012.2504

in Multi-Objective Optimization Problems. A MOOP has a number of objective functions, which are to be minimized or maximized [10].

Often, traditional approaches for solving multi-objective optimization problem try to change the multiple objectives into a single objective problem in which only a global optimal point is desired. The MOOP produces a set of solutions which are superior to the rest of the solutions with respect to all objective criteria but are inferior to other solutions in one or more objectives [5]. These solutions are called Pareto Optimal solutions or non-dominated solutions. A Pareto optimal set is the mathematical solution to a multi-objectives problem [10] [4]. A solution is Pareto-optimal if no other solution can improve one object function without reducing at least one of the other objectives.

Genetic Algorithm is the most popular heuristic technique to solve Multi-Objective Design and Optimization problems. In this paper, CBEGA employs both evolutionary and multi-objective optimization techniques. The objectives are maximization of coverage and minimization of execution time. The immigration rate, one of the GA parameters, is an elitist operator that controls the migration of test cases from one era to next era. Elitism is the process of selecting best individuals from a population. Elitism is important since it allows the solutions to get better over time. Elitism can speed up the performance of the GA significantly; also it helps to prevent the loss of good solutions once they have been found.

The rest of this paper is organized as follows: Section 2 describes the concept of genetic algorithms. Section 3 describes the multi-objective optimization problem. Section 4 describes the new Class-Based Elitist Genetic Algorithm, section 5 consists of experiment and result analysis and section 6 consists of conclusion.

2. EVOLUTIONARY SEARCH USING GENETIC ALGORITHMS

Genetic Algorithms (GAs) are adaptive heuristic search techniques based on the evolutionary techniques of natural selection, recombination and mutation [7]. The principle behind GAs is that they create and maintain a population of individuals represented by chromosomes. The input to the algorithm is a set of potential solutions to that problem and a metric called a *fitness function* allows each candidate to be quantitatively evaluated.

Genetic Algorithm then evaluates each candidate according to the fitness function. Only certain individuals in the current population have been selected to the next generation while a bad one is being eliminated from the selection process. The fitness value of each element, which could be the objective of the solution, is used to distinct good and bad individuals from the population. Reproduction selects individuals with high fitness values in the population, and through crossover and mutation of such individuals, a new population is derived in which individuals may be even better fitted to their environment. Crossover involves two chromosomes swapping chunks of data to get new offspring where as mutation introduces slight changes into a small proportion of chromosome and is representative of an evolutionary step. The structure of a simple genetic algorithm is given below

```
Simple Genetic Algorithm()
{ initialize population;
evaluate population;
while termination criterion not reached
{ select solutions for next population;
perform crossover and mutation;
evaluate population;}
}
```

The above algorithm will iterate until the population has evolved to form a solution to the problem, or until a maximum number of iterations have occurred.

2.1. Multi-Objective Optimization Problems (MOOP)

A Multi-Objective Optimization Problem has a number of objective functions, which are to be minimized or maximized, i.e., the process of simultaneously optimizing two or more conflicting objectives subject to certain constraints [5].

The general form for Multi-Objective Optimization Problems (MOOP) can be expressed as,

Fitness function $f_m(x)$, where m = 1, 2, 3… M no. of objective functions &
x = single candidate solution.

Multi-Objective Optimization is sometimes referred as vector optimization, because a vector of objectives, instead of a single objective, is optimized. For example, try to achieve maximum path coverage with a minimum execution time. MOOP can be linear or non-linear type. There are two general approaches to multiple-objective optimization. One is to combine the individual objective functions into a single composite function or move all but one objective to the constraint set. In the former case, determination of a single objective is possible with methods such as utility theory, weighted sum method, etc., but the problem lies in the proper selection of the weights or utility functions to characterize the decision-makers preferences. In practice, it can be very difficult to precisely and accurately select these weights, even for someone familiar with the problem domain. The second general approach is to determine an entire Pareto optimal solution set or a representative subset [4]. A Pareto optimal set is a set of solutions that are non-dominated with respect to each other.

2.2. Single and Multi-Objective Optimization Problems

In single-objective optimization problem only single optimal solution is required whereas in multi-objective optimization problem produces a set of solutions which are superior to the rest of the solutions with respect to all objective criteria; but are inferior to other solutions in one or more objectives. These solutions are called Pareto optimal solutions or non-dominated solutions.

2. 2. 1. Ranking Approach

In multi-criteria decision making (MCDM) approach, the task of finding multiple Pareto-optimal solutions is achieved by executing many independent single-objective optimizations, each time finding a single Pareto-optimal solution. A parametric scalarizing approach (such as the weighted-sum approach, ε-constraint approach, and others) can be used to convert multiple objectives into a parametric single-objective objective function. By simply varying the parameters (weight vector or ε-vector) and optimizing the scalarized function, different Pareto-optimal solutions can be found.

MCDM methodology employs multiple, independent single objective optimizations to find different Pareto-optimal solutions. The Pareto-optimal front corresponds to global optimal solutions of several scalarized objectives. However, during the course of an optimization task, algorithms must overcome a number of difficulties, such as infeasible regions, local optimal solutions, at regions of objective functions, isolation of optimum, etc., to converge to the global optimal solution.

2. 2. 2. Diversity Approach

The NSGA-II procedure is one of the popularly used evolutionary multi-objective optimization (EMO) procedures which attempt to find multiple Pareto-optimal solutions in a multi-objective optimization problem. Instead of arbitrarily discarding some members from the last front, the points which will make the diversity of the selected points the highest are chosen. The crowded-sorting of the points of the last front which could not be accommodated fully is achieved in the descending order of their crowding distance values and points from the top of the ordered list are chosen.

2. 2. 3. Hybrid Approach

Combination of both the approaches is used in Non-dominated Ranking Genetic Algorithm NRGA [13], for solving multi-objective optimization problems. In stage-based, two stages are used for test case generation in which non-domination is used in test case generation and the ranking is used in fitness evaluation.

3. CLASS-BASED ELITIST GENETIC ALGORITHM

In this section, CBEGA is presented to generate tests for the object-oriented programs. Fig.1 shows the interaction of evolutionary testing with CBEGA. The CBEGA parameters are initial population size, crossover probability, mutation probability, maximum no. of generations per era, maximum no. of eras and immigration rate. For test case selection we use dominance based methodology and for calculating fitness, ranking technique. Both the techniques are used collectively for test data generation.

3.1. Control-flow and Data-flow Testing

The input representation is in the form of control flow graphs (CFGs) and the output will be the generated test cases. CFG will give the dependencies between variables associated with specific nodes [2].

Data-flow testing focuses on execution of all the interactions between variables i.e. all def-use-associations for any variable ‘v’ in the program. Criteria for generating test cases consists of a triple [d, u, v] where‘d’ def_node, ‘u’ use_node & ‘v’ associated variable.
The inputs of GA are:

1) An instrumented version of the class under test, and
2) Test requirements, which satisfy a given control or data-flow test criterion.

3.2. Test Data Generation

Test generation for structural programs is the process of identifying test data, which execute the program under test and satisfy a given test criterion [8] [9]. Test generation for the object-oriented programs involves generating: 1) test cases, which are sequences of methods issue on an object of the class under test and satisfy a given test criterion, and 2) test data, which is a set of values for the arguments of the called methods. Thus test data generation in improved GA takes place in two stages. First stage generates the method sequences and the second stage generates the corresponding argument values for the methods.

A typical test case for a given [d, u, v] contains the following:

- Invocation of a constructor 'c'
- Method m_d that causes the execution of 'd'
- Method m_u that causes the execution of 'u'
- Values to the parameters 'v'

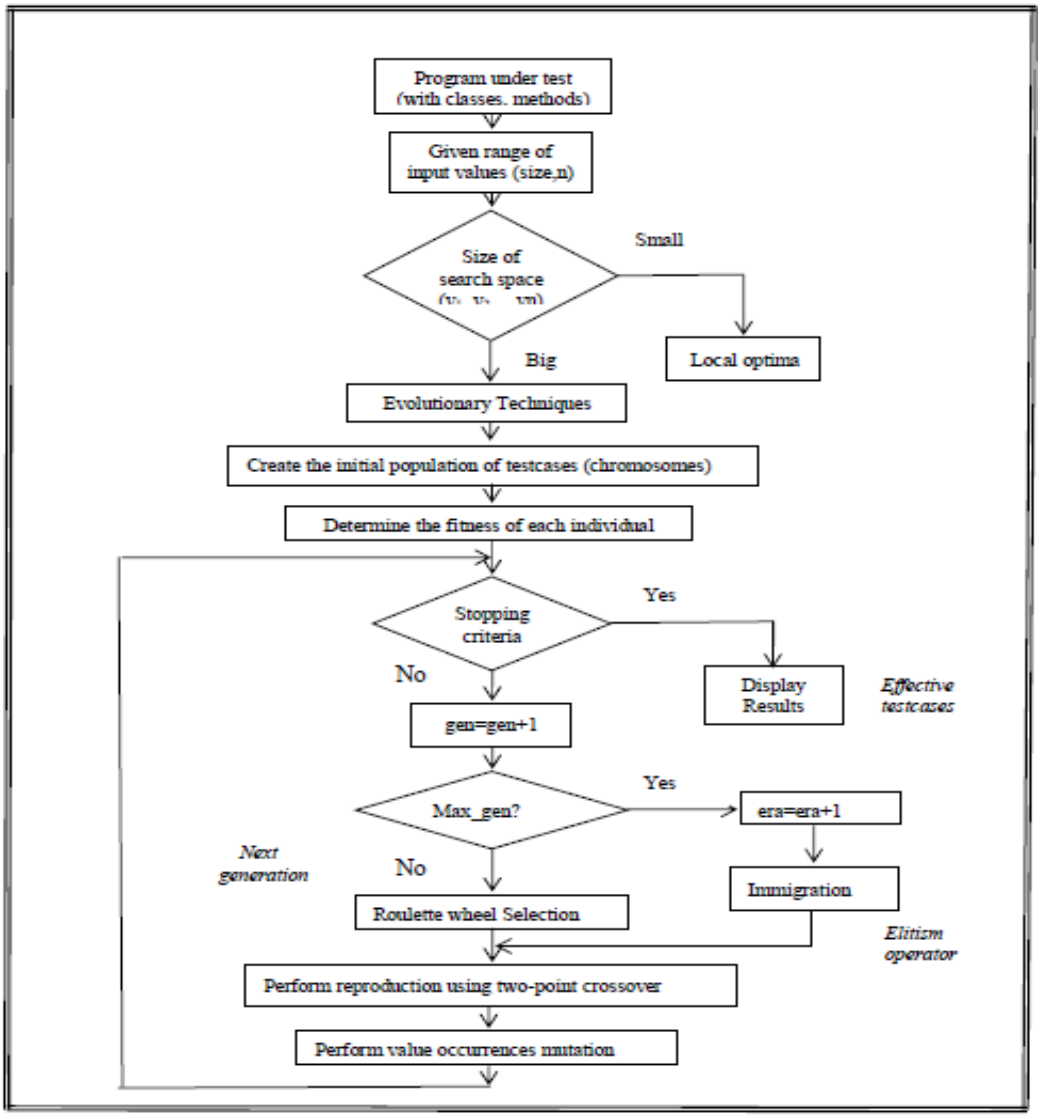


Figure 1. Interaction of evolutionary testing with Class-Based Elitist Genetic Algorithm

3.3. Genetic Operations

Genetic operators such as roulette wheel selection, two-point crossover, value occurrences mutation and immigration rate are experimented in CBEGA for test data generation.

3.3.1 Roulette wheel Selection

In this type of selection, the individual is selected on the basis of fitness. The probability of an individual to be selected is calculated based on the fitness value of the individual, and thus individuals with higher fitness values have better chances of being selected.

3.3.2 Two-point Crossover

The crossover allows us to combine individuals i.e. chromosomes that were selected for reproduction. Two-point crossover calls for two points to be selected on the parent chromosomes. Everything between the two points is swapped between the parent chromosomes, rendering two child chromosomes.

Testcase1: 1 3| 2 5 4| &7 8 3
Testcase2: 1 2| 4 3 5| &9 12 33
Offspring1: 1 3 4 3 5 &7 8 3
Offspring2: 1 2 2 5 4 &9 12 33

3.3.3 Value Occurrences Mutation

Mutation alters one or more gene values in a chromosome from its initial state. Value Occurrences mutation attempts to replace a duplicate value of an individual with a missing value to improve the individual's fitness.

Testcase1: 1 4 4&5 9
Offspring: 1 4 2&5 9

3.3.4 Immigration Rate

Elitism improves the performance and convergence of the GA. Immigration is an elitist operator that controls the migration of test cases from one era to next era.

$$Immigration\,rate(I) = \frac{no\ of\ test\ cases\ reduced\ to\ next\ generation}{\text{no of test cases in the current generation}}$$

3. 4. Fitness Function

The fitness value of the test case is calculated using two fitness functions f1 and f2 since we are considering two objectives for optimization namely maximization of coverage and minimization of execution time. The first fitness function is a maximization of coverage and the second one is a minimization function. The functions are given by

$$f1(x) = \frac{coverednodes}{totalnodes} \quad [1]$$

$$f2(x) = \frac{executiontimeofatestcase}{totalexecutiontime} \quad [2]$$

$$f(x) = \max\{f1(x)\} + \min\{f2(x)\} \quad [3]$$

The optimal test cases are chosen by their fitness values that are calculated based on the values from these two fitness functions. The best test case will have value equal to 1 for the function f1 and will have the value approximately equal to zero for the function f2.

4. EXPERIMENTS AND RESULT ANALYSIS

The CBEGA is used for test data generation in java with two stages. The CBEGA has been tried several times with different values of population size (50, 75, 100…), mutation probability and crossover probability (merely equals 1). The effectiveness of CBEGA is studied by applying the algorithm on simple java classes.

The implementation starts with the Class Control Flow Graph (CCFG) of the class which is drawn with the help of a set of def-use associations identified from the Class under Test (CUT). CBEGA initializes each sequence by a randomly selected constructor and the methods. Stage 1 generates or updates sequences of method calls. Stage 2 performs the traditional genetic algorithm to generate the required parameters.

The execution of genetic algorithm repeats till all the test requirements is satisfied or the maximum number of eras is reached or until getting the optimal solution with fitness value nearly equals 1. The final test cases are the minimal test set obtained from the resultant test cases of GA which satisfy all the target test requirements. Thus, the minimum no. of test cases is found for testing a given class.

The sample programs taken have the no. of conditions in the range 18 to 42 with Stack having 18 conditions and Bit Set containing 42 conditions. The immigration rate is set to 0.5 and no. of generations per era is 30.The execution time increases with increasing number of conditions as shown in the Figure 3. From Figure 2 it is inferred that coverage is greater in CBEGA than in simple GA. Thus when compared to simple GA, CBEGA performs better in terms of efficiency in generating test cases. The results obtained for sample java programs are shown in table 1.

Table 1 shows the subject programs studied in this paper. Each program and its test suite are taken from Software Infrastructure Repository (SIR) [2]. *printtokens* and *schedule* are part of Siemens suite. SIR contains multiple test suites for these programs; four test suites are chosen randomly. *space* is an Array Description Language (ADL) interpreter that was developed by European Space Agency. From SIR, four test suites are chosen randomly for each program. Coverage data and the execution time are noted for the subject programs with GA and CBEGA. Figure 4 shows the results of *printtokens, schedule* and *space* test suites.

Table 1: The results obtained for the sample java classes comparing GA and CBEGA

Sample programs	No. of eras	No. of generations	No. of conditions	Test suite	Coverage for GA%	Coverage for CBEGA%	Execution time (ms) CBEGA
LinkedList	3	80	20	5	85	95	2100
Stack	6	170	18	10	83	98	2082
TreeMap	5	150	25	10	92	96	2445
BinomialHeap	3	95	38	8	92	97	3340
Bitset	3	80	42	8	90	95	4025
Sample programs	**No. of eras**	**No. of generations**	**No. of conditions**	**Test suite**	**Coverage for GA%**	**Coverage for CBEGA%**	**Execution time (ms) CBEGA**
HashMap	3	100	32	8	90	96	2785
Binary SearchTree	4	120	24	8	83	91	2235
FibonocciHeap	3	90	36	8	88	97	2910
Disjset	3	95	40	8	90	95	3695
TreeSet	4	110	28	8	89	96	2612
Car crash crisis management system	7	200	54	5	86	96	5015
eHealth system	6	180	48	6	88	95	4923
Traffic collision avoidance system	3	95	47	6	87	97	4265
N-queens problem	4	110	52	8	85	97	4420
Chess playing	4	110	58	6	86	96	5150
Subject programs from Siemens' test suite							
printtokens	4	100	60	1	83.12	89.35	7215
				2	87.36	93.72	7373
				3	80.3	86.48	7388
				4	90.8	96.21	7380
schedule	4	110	42	1	85.46	89.7	6783
				2	90.81	93.65	6818
				3	89.83	91.94	6802
				4	94.5	98.31	6950
space	3	75	74	1	81.79	84.53	8020
				2	83.16	87.42	8280
				3	72.38	80.68	7998
				4	88.84	93.65	8352

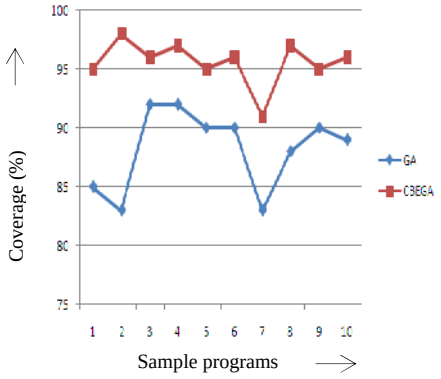


Figure 2. Graph showing coverage

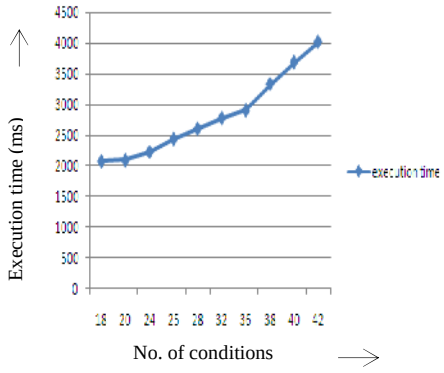
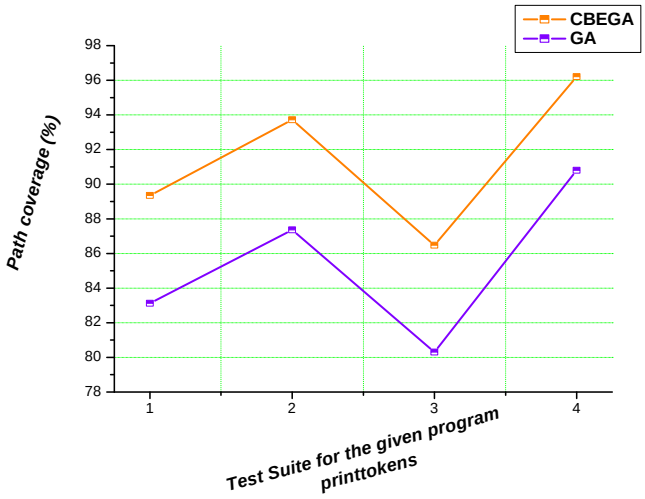
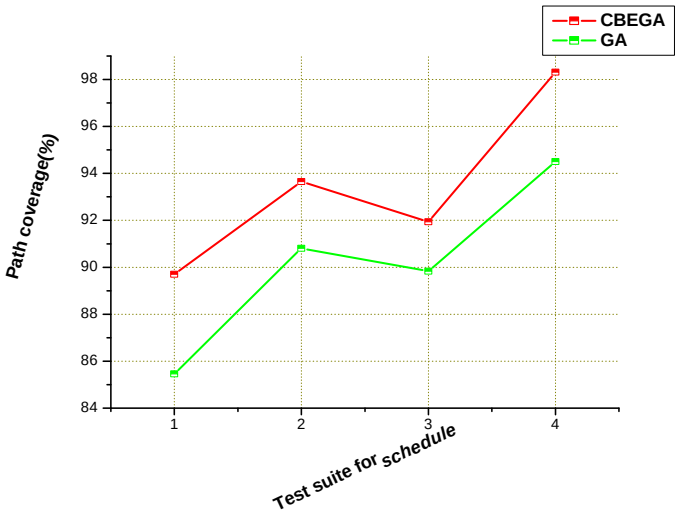


Figure 3. Graph showing execution obtained for GA and CBEGA time for CBEGA





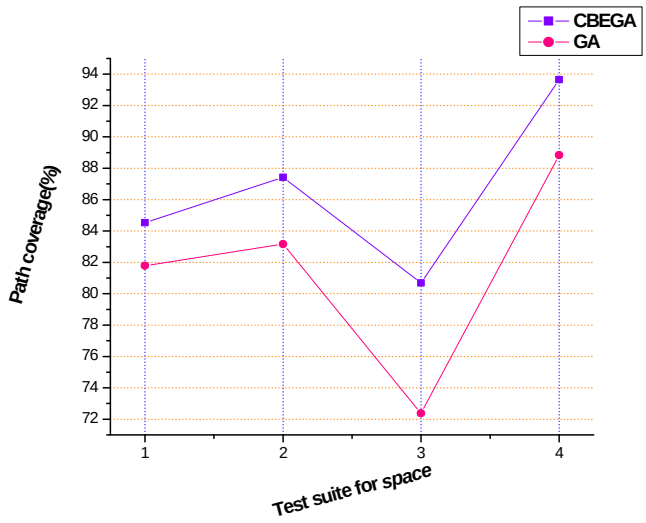


Figure 4. Showing the results of *printtokens*, *schedule* and *space* test suites.

5. CONCLUSION

Thus, the class-based elitist genetic algorithm has been used for automatic generation of object-oriented test cases. The fitness depends on the multiple criterion path coverage and execution time of the test cases. The results for sample java programs show that the efficiency of CBEGA over simple GA in terms of coverage as well as reduction in test suite size. When the no of classes and conditions increases, the execution time will also be increased. Experiments have been done for sample java programs and medium- complex also. The new algorithm can be used for similar type of software engineering problems and also for real-time applications.

REFERENCES

[1] Anon Sukstrienwong, (2011) "Solving multi-objective optimization under bounds by genetic algorithms", International Journal of Computers, Vol.5, Iss.1, pp 18-25.

[2] Ahmed S. Ghiduk, (2010) "Automatic Generation of Object-Oriented Tests with a Multistage-Based Genetic Algorithm", Journal of computers, Vol. 5, No. 10, pp 1560-1569.

[3] Dhirendra Pal Singh & AshishKhare, (2011) "Different Aspects of Evolutionary Algorithms, Multi-Objective Optimization Algorithms and Application Domain", International Journal of Advanced Networking and Applications, Vol. 2, Iss.04, pp 770-775.

[4] Mark Harman, Kiranlakhotia & Phil McMinn, (2007) "A Multi-Objective Approach to Search-Based Test Data Generation", Proceedings of the Genetic and Evolutionary Computation Conference GECCO'07, pp 1-8.

[5] Abdullah Konak, David W. Coit & Alice E. Smith, (2006) "Multi-objective optimization using genetic algorithms: A tutorial", Reliability Engineering and System Safety, Elsevier, pp 992-1007.

[6] Bruce A. Conway, (2011) "A Survey of Methods Available for the Numerical Optimization of Continuous Dynamic Systems", Journal of Optimization Theory and Applications (JOTA) of Springer, pp 1-36.

[7] RuchikaMalhotra & MohitGarg, (2011) "An Adequacy Based Test Data Generation Technique Using Genetic Algorithms", Journal of Information Processing Systems, Vol.7, No.2, pp 363-384.

[8] Andreas S.Andreou, Kypros A. Economides, & Anastasis A. Sofokleous, (2007) "An Automatic software test-data generation scheme based on data flow criteria and genetic algorithms", 7th International Conference on Computer and Information Technology, pp 867-872.

[9] Yong Chen, & Yong Zhong, (2008) "Automatic Path-oriented Test Data Generation Using a Multi-population Genetic Algorithm", Fourth International Conference on Natural Computation, pp 566-570.
[10] Kalyanmoy Deb, (2005) "Single and Multi-Objective Optimization using Evolutionary Computation", KanGALRt- No. 2004002, Technical Report, pp 1-24.
[11] Yuanyuan Zhang, (2010) "Multi-Objective Search-Based Requirements Selection and Optimization", Ph.D Thesis, University of London, pp 1-276.
[12] C. A. Coello, Gary B. Lamont David A. Van Veldhuizen, (2007) "Evolutionary Algorithms for Solving Multi-Objective Problems", 2nd Edition, Springer.
[13] Omar Al Jadaan, Lakishmi Rajamani & C. R. Rao, (2008) "Non-dominated Ranking Genetic Algorithm for solving Multi-Objective Optimization Problems: NRGA", Journal of Theoretical and Applied Information Technology, pp 60-67.

AUTHORS

Mrs. P. Maragathavalli received her B.E. degree in Computer Science and Engineering from Bharathidasan University, Trichirappalli in 1998 and M.Tech. degree in Distributed Computing Systems from Pondicherry University, in 2005. She joined Pondicherry Engineering College in 2006 and currently working as Assistant Professor in the Department of Information Technology. Now she is pursuing her PhD degree in Computer Science and Engineering. She has published research papers in International and National Conferences. She is a Life member of Indian Society for Technical Education.

Dr. S.Kanmani received her B.E and M.E in Computer Science and Engineering from Bharathiar University and Ph.D from Anna University, Chennai. She has been the faculty of department of Computer Science and Engineering, Pondicherry Engineering College from 1992. Presently she is headin g the department of in Information Technology. Her research interests are in software engineering, software testing and object oriented systems. She is a member of Computer Society of India, ISTE and institute of engineers India. She has published about 50 papers in international conferences and journals.