Automated Test Case Generation on the Basis of Branch Coverage Using Teaching Learning Based Optimization

Maneesh Kumar[1], Rajdev Tiwari[2], Rajeev[3]
[1-3]*CSE Department, Noida Institute of Engineering & Technology, Greater Noida*
[1]maneeshniet1987@gmail.com,
[2]rajdevtiwari@yahoo.com

***Abstract* - The most expensive and the time consuming step of software development life cycle is its testing. There are several research proposed to develop a low cost, scalable and effective method for software testing. With the help of the techniques of automatic generation of test cases one can easily and very efficiently find an optimal set of cases that allow an appropriateness criterion to be fulfilled, which helps in reducing the cost of software testing and resulting in more efficient software testing. In this paper we are trying to discuss a new technique for automated test case generation using teaching learning based optimization. This technique extends the random testing by the use of teaching learning based optimization where the fitness function is based on the branch coverage.**

***Keywords* - Automatic test generation, Software testing, teaching learning based optimization.**

I. INTRODUCTION

Software testing is very labor intensive and expensive approximately 50% of the cost of a software system development is spending on testing. If the process of testing is automated then this cost of testing can be reduced - an automate test case generator is one of the most important component in a testing environment –that automatically generates test data for a given program [1]. Test case generation in software testing is the process of indentifying program input data. A test data generator is a tool that helps programmer in the development of test data for a program. The automation of test data generation is an important step in reducing the cost of software development and maintenance.

1.1 Quality characteristics of a test case

- How effective in detecting defects?
- How exemplary? (The more exemplary, the fewer test cases needed)
- How economic?
- How evolvable? (Maintenance effort)

1.2 Problem with Manual Testing?

The manual testing is not so effective because of the following reasons:

- It covers functionality only at human speed
- Major investment in process and not in errors detection
- Limited by resources availability and not necessity
- Inexact repetition of tests
- Inaccurate result checking
- Difficult regression testing means or NO regression testing.

The paper is organized as follows: Section 2 gives some important reasons why we have need of test automation. Section 3 describes the different optimization techniques and proposed technique, which is used to reduce the cost of software testing. Section 4 describes the literature which is used in the paper. Section 5 describes the purposed methodology. Section 6 shows TLBO technique for automatic test-data generation, and the results of applying this algorithm to an example program. And presents the results of the experiments that are conducted to evaluate the effectiveness of the proposed TLBO compared to the random testing technique, to evaluate the effectiveness of the new fitness function and the technique used to reduce the cost of software testing. Section 7 presents the conclusions and future work.

II. NEED OF TEST AUTOMATION

2.1 Faster and more accurate

It's faster and more accurate than manual testing. Automation is 50 times faster, depending upon the speed of the driver machine and the speed of the application to process information (inserts, updates, deletes and views).Using Test tools for inputs are much more accurate than manual test inputs. The average typist makes 3 mistakes for every 1,000 keystrokes. Also, automation tools never tire, get bored, take shortcuts or make assumptions of what works.

2.2 Ease of access -Automation tools allow access to data, objects, operating, and communication protocols, that manual tester cannot access. This allows for a test suite with much greater depth and breadth.

2.3 Accumulation – long-term benefits can be derived through reuse of tests, once tests are developed. The numbers of tests are always increasing as the application/architecture matures. Engineers can constantly add onto test suite and not have to test the same functionality over and over again.

2.4 Ability to manage – Ability to manage artifacts through automation tools.

2.5 Identification of issues – Automated testing assists with the identification of issues early in the development process, reducing costs.

2.6 Repeatable Process – An automation suite provides a repeatable process to check out the functionality on the functional side and scalability on the performance side.

2.7 24x7 Availability – Scripts can run any time during the day or night unattended.

2.8 Formal process – Automation forces a more formal process on test teams, due to the nature of the explicitness of the artifacts and the flow of information that is needed.

2.9 Retention of customers – When sites are not functioning correctly or performing poorly, customers may leave and never come back. Then decide the cost to your business of that scenario? Performing correct and systematic automated testing helps assure a quality experience for the customer – both internal and external.

2.10 Job satisfaction – The Test Engineers no longer manually execute the same test cases over and over.

Test case generation is categorized as optimization problem in which an optimal test case is intended to be generated so that the software can be tested to the maximum possible extend. Various optimization techniques that can be applied to generate the test cases are briefed in section below.

III. OPTIMIZATION TECHNIQUES

3.1 Genetic Algorithm - GA are adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics. GA are a part of evolutionary computing, a rapidly growing area of AI. GA is inspired by Darwins theory about evolution –"survival of the fittest". It is generally used in situation where the search space is relatively large and cannot be traversed efficiently by classical search methods [2]. They use the same mixture of selection, recombination and mutation to evolve a solution to a problem [3].

3.2 PSO - Particle Swarm Optimization (PSO) was initially proposed to find optimal solutions for continuous space problems by Kennedy and Eberhart [4, 5] in 1995. PSO is inspired by social descriptions of behavior and swarm theory, simple methods were developing for resourcefully optimizing non-linear mathematical functions. PSO pretends swarms such as crowds of animals, groups of birds or institutes of fish.

Similar to genetic search, the system is prepared with a population of random solutions, called particles. Where each particle sustains its own current location, its present rapidity and its personal best position discovered so far. The swarm is also attentive of the global best position achieved by all its members.

3.3 TLBO – Rao et al. (2011, 2012) and Rao and Patel (2012) proposed a (Teaching–learning-based optimization algorithm is a teaching–learning process inspired algorithm) TLBO. It is a population- based iterative learning algorithm that shows some mutual features with other evolutionary computation (EC) algorithms. The algorithm mimics teaching–learning ability of teacher and learners in a class room. A high quality teacher is usually considered as a highly learned person who trains learners so that they can have better results in terms of their marks or grades. Moreover, learners also learn from the interaction among themselves which also helps in improving their results [6]. Two vital components of the algorithm are Teacher and learners and describe two basic modes of the learning, through teacher (known as teacher phase) and interacting with the other learners (known as learner phase). Working of both the phases is explained below.

3.3.1 Teacher phase

During this phase a teacher emphasizes to increase the mean result of the class in the subject taught by him or her as per his/her capability. At any iteration i, assume that there are m number of subjects (i.e., design variables), n number of learners (i.e., population size, k01, 2,..., n) and Mj, i be the mean result of the learners in a particular subject j (j01, 2,..., m) The best overall result Xtotal-kbest,i considering all the subjects together obtained in the entire population of learners can be considered as the result of best learner kbest. However, as the teacher is usually considered as a highly talented, learned person who trains learners so that they can produce better results, the algorithm identified the best learner as the teacher. The difference between the existing mean result of each subject and the corresponding result of the teacher for each subject is given by,

$$\textit{Difference Mean}_{j,i} = r_i\,(X_{j,kbest,i} - TF{*}M_{j,i}) \qquad (1)$$

Where, Xj,kbest,i is the result of the best learner (i.e., teacher) in subject j. TF is the teaching factor which decides the value of mean to be changed, and ri is the random number in the range [0, 1]. Value of TF can be either 1 or 2. The value of TF is decided randomly with equal probability as,

$$TF = round\,[1+rand(0,1)\{2-1\}] \qquad (2)$$

The value of TF is not given as an input to the algorithm and its value is randomly decided by the algorithm using Eq. (2). After conducting a number of experiments on many benchmark functions it is concluded that the algorithm performs better if the value of TF is between 1 and 2. Simplify the algorithm, the teaching factor is suggested to take either 1 or 2 depending on the rounding up criteria given by Eq.(2).

Based on the Difference_Mean j,k,i, the existing solution is updated in the teacher phase according to the following expression.

$$X'_{j,k,i} = X_{j,k,i} + \textit{Difference\_Mean}_{j,k,I} \qquad (3)$$

where X'j,k,i is the updated value of Xj,k,i. X'j,k,i is accepted if it gives better function value. All the accepted function values at the end of the teacher phase are maintained and these values become the input to the learner phase. The learner phase depends upon the teacher phase.

3.3.2 Learner phase

Learners increase their knowledge by interaction among themselves. A learner interacts randomly with other learners for enhancing his or her knowledge. A learner learns new things if the other learner has more knowledge than him or her. Considering a population size of n, the learning phenomenon of this phase is expressed below.

Randomly two learners P and Q are selected such that X'total-P,i ≠ X'total-Q,i (where, X'total-P,I and X'total-Q,i are the updated values of Xtotal-P,i and Xtotal-Q,i respectively at the end of teacher phase)

$$X''_{j,P,i} = X'_{j,P,i} + r_i\,(X'_{j,P,i} - X'_{j,Q,i}),\ \text{If } X'_{total\text{-}P,i} > X'_{total\text{-}Q,i} \qquad (4)$$

$$X''_{j,P,i} = X'_{j,P,i} + r_i\,(X'_{j,Q,i} - X'_{j,P,i}),\ \text{If } X'_{total\text{-}Q,I} > X'_{total\text{-}P,i} \qquad (5)$$

(Above equations is for maximization problem, reverse is true for minimization problem)

X"j,P,i is accepted if it gives a better function value.

IV. LITERATURE REVIEW

Vigorous research has been carried out in the automatic test case generation for number of years. In 1990 Bogdan Korel a proposed an alternative approach of test data generation which is based on actual execution of the program under test, function minimization method and dynamic data flow analysis. Test data are developed for the program using actual values of input variables [17] while M R Keyvanpour proposed hybrid method called GA-NN are then considered and studied in order to understand when and why a learning algorithm is effective for testing problem [18].

Automated test case generation is a classical problem of optimization therefore a number of optimization techniques have been applied to improve the testing capabilities of a tool. Artificial Bee Colony Optimization is used for the test data generation optimization [8]. Another optimization algorithm is Ant Colony Optimization is used for test data generation [7][11].

Anu Sharma proposes an efficient cost effective approach for optimizing the cost of testing using Tabu Search (TS) [9]. Andreas Windisch has applied particle swarm optimization (PSO) on generating test cases [12]. Moheb R .Girgis, 2005[13] presents an automatic test data generation technique that uses a genetic algorithm (GA). The most widely used metaheuristic techniques in this yield is genetic algorithm (Gold berg, 1989). This techniques is based on the principle of genetic and Darwin's theory of evolution, Miller et al [14], Pargas et al [15], Lin-Yeh[16].

Recently, the use of Teaching Learning Based Optimization in optimization became the focus of several research studies [19-22]. Babak Amiri is also applied TLBO on cluster[10].As discussed previously optimization techniques like GA, PSO, ACO, ABC etc have already been used by researchers to optimize the test data generation process but no one could claim that a particular technique is the most superior one as compared to the others in all circumstances. So, the scope of applying other optimization techniques for test case generation remains open, keeping this in view it is planned to apply TLBO for generating the optimal test cases in this paper.

V. PROPOSED METHODOLOGY

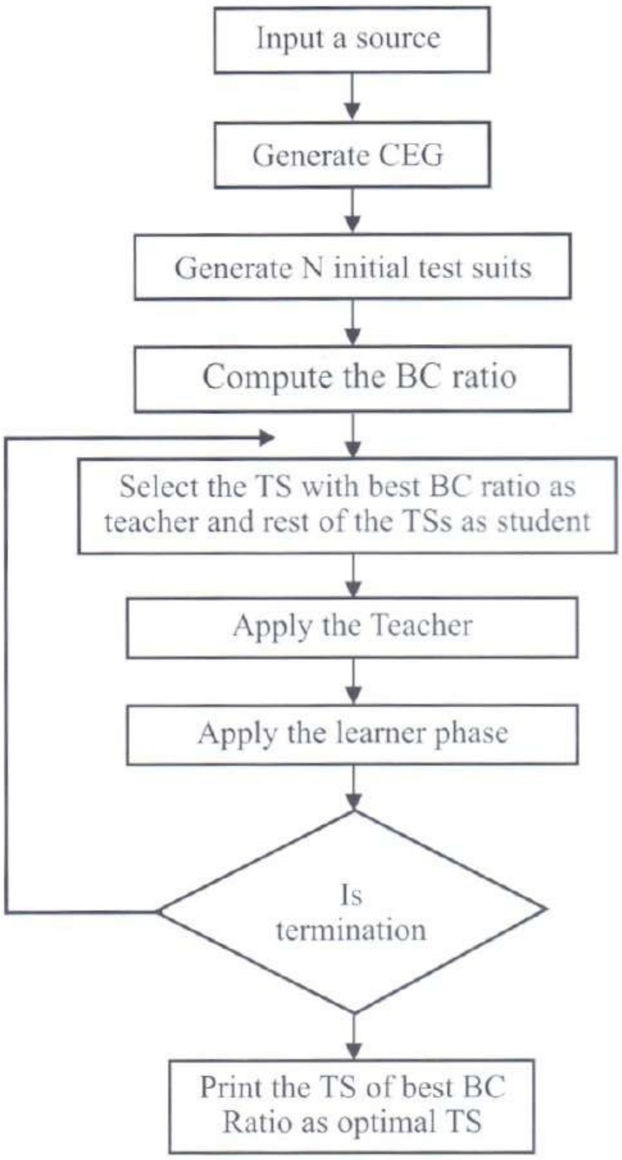
In this proposed method (Fig.1) a source program preferably written C language will be taken as input. A control flow graph corresponding to given source program will be generated. On the basis of number of variables in the source program test cases will be generated randomly. A test case will be a tuple of 'n' values if there are 'n' variables in the program, m number such test cases will be clubbed together to form a test suit.

Initially 'k' number of test suits will be taken to start the optimization process. In the next step branch coverage ratio of each 'k' number of test suits will be computed. On the basis of BC ratio test suit with maximum BC ratio will be declared as teacher and other test suits as students. After which TLBO is applied to optimize the BC ratio. We apply the process of teaching phase and learner phase for much iteration until we have on the termination criteria. After termination we will get the optimized test suit. That has maximum branch coverage ratio.

VI. AN EXAMPLE

6.1 Source Program

```
Int a,b,c;
scan ("%d,%d,%d", &a,&b,&c);
1,2,3: if (a<=0 II b<=0 II c<=0)
4:     printf("\nwrong input"); Else {
5:     if (a<b)
6:     swap (a,b);
7:     if (a<c);
8:     swap (a,c);
9:     if (b<c);
10:    swap (b,c);
11:    if (a>=b+c)
12:    printf ("\nwrong input");
13:    else if (a==b){
14:    if (b==c)
15:    printf (\nEquilateral"); else
16:    printf ("\nIsoslist");} else {
17:    if (b==c)
18:    printf ("nIsoslist"); Else
19:    printf ("\nScalen");}
```



Input a source
Generate CEG
Generate N initial test suits
Compute the BC ratio
Select the TS with best BC ratio as teacher and rest of the TSs as student
Apply the Teacher
Apply the learner phase
Is termination
Print the TS of best BC Ratio as optimal TS

Fig. 1 The proposed methodology

6.2 Control flow graph

It is shown in Fig.2.

6.3 Test Suit Generation

Random Test Suits

TS (0)—{(3, 4, 5) (5, 7, 8) (11, 12, 13) (6, 8, 8)}

TS (1)—{(0, 1, 3) (4, 4, 8) (5, 5, 5) (2, 4, 6)}

TS (2)—{(3,4,7) (11,12,15) (15,21,18) (18,12,12)}

TS (3)—{(7, 8, 7) (12, 0, 16) (10, 5, 8) (2, 2, 2)}

Path Followed By Test Case

(3, 4, 5)---	1,2,3,5,6,7,8,9,10,11,13,17,19
(5, 7, 8)---	1,2,3,5,6,7,8,9,10,11,13,17,19
(11, 12, 13)---	1,2,3,5,6,7,8,9,10,11,13,17,19
(6, 8, 8)---	1,2,3,5,6,7,9,10,11,13,14,16
(0, 1, 3)---	1,4
(4, 4, 8)---	1, 2,3,5,7,8,9,11,12
(5, 5, 5)---	1,2,3,4,5,7,9,11,13,14,15
(2, 4, 6)---	1,2,3,5,6,7,8,9,10,11,12
(3, 4, 7)---	1,2,3,5,6,7,8,9,10,11,12
(11, 12, 15)---	1,2,3,5,6,7,8,9,10,11,13,17,19
(15, 21, 18)---	1,2,3,5,6,7,9,10,11,13,17,19
(18, 12, 12)---	1,2,3,5,6,7,9,11,13,17,18
(7, 8, 7)---	1,2,3,5,6,7,9,11,13,17,18

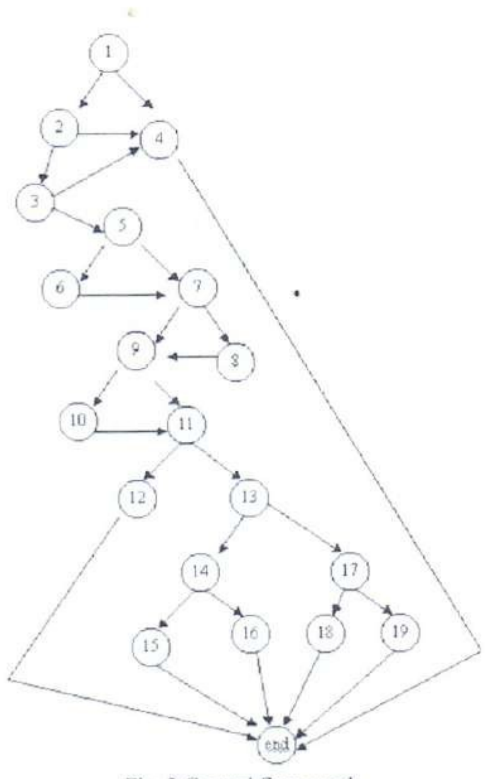


Fig. 2 Control flow graph

0.5(16-1*8) = 4

Updated TC—(5+1, 7+(-3), 8+4) → (6, 4, 12)

Next iteration.

0.5(12-1*6) = 3

0.5(0-1*4) = -2

0.5(16-1*12) = 2

Updated TC—(6+3, 4+(-2), 12+2) → (9, 2, 14)

Next iteration.

0.5(12-1*9) = 1.5 → 2

0.5(0-1*2) = -1

0.5(16-1*14) = 1

Updated TC—(9+2, 2+(-1), 14+1) → (11, 1, 15)

Next iteration.

0.5(12-1*11) = 0.5 → 1

0.5(0-1*1) = -0.5 → -1

0.5(16-1*15) = 0.5 → 1

Updated TC—(11+1, 1+(-1), 15+1) → (12, 0, 16)

TS (1) is update according with Teacher.

TS (1)—{(4,4,8) (0,1,3) (5,5,5) (2,4,6) }

TS (3)—{(7,8,7) (12,0,16) (2,2,2) (10,5,8)}

Updated TS (1) --- {(4,4,8) (12,0,16) (2,2,2) (2,4,6)}

BCR TS (1) ---20/29=0.689

TS (2) is update according with Teacher.

TS (2)—{(3,4,7) (15,21,18) (11,12,15) (18,12,12)}

TS (3)—{(7,8,7) (10,5,8) (12,0,16) (2,2,2)}

Updated TS(2)---{(3,4,7) (15,21,18) (12,0,16) (2,2,2)}

BCR TS (2) ---22/29=0.758

Update BCR of TS

BCR TS(0)--- {(3,4,5) (12,0,16) (6,8,8) (2,2,2)}=> 23/29=0.793

BCR TS(1)--- {(4,4,8) (2,4,6) (2,2,2) (12,0,16)}=> 20/29=0.689

BCR TS(2)--- {(3,4,7) (15,21,18) (12,0,16) (2,2,2)}=> 22/29=0.758

BCR TS(3)--- {(7,8,7) (12,0,16) (10,5,8) (2,2,2)}=> 20/29=0.689

On the Basis of updated BCR TS (0) is Teacher.

TS (1) is Update According with Teacher

TS(1)---{(3,4,5) (6,8,8) (2,2,2) (12,0,16)}

BCR TS (1) ---23/29=0.793

TS (2) is Update According with Teacher

TS(2)--- {(6,8,8) (15,21,18) (12,0,16) (2,2,2)}

BCR TS (2) ---23/29=0.793

TS (3) is Update According with Teacher

TS(3)--- {(7,8,7) (12,0,16) (3,4,5) (2,2,2)}

BCR TS (3) --- 23/29=0.793

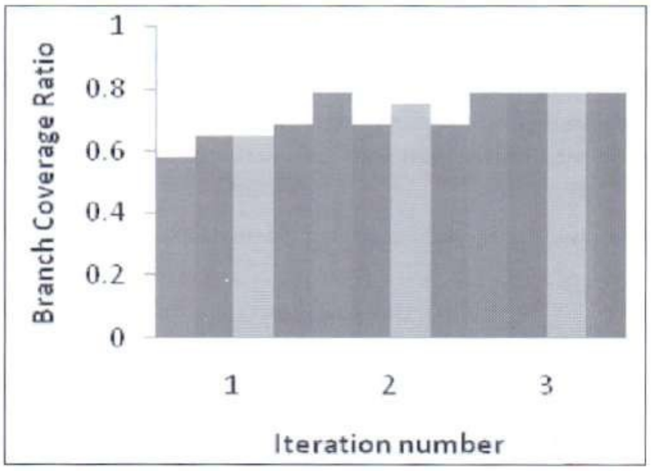


Fig.3 Iteration-wise Branch Coverage Ratio.

It is observed from chart shown in Fig. 3 that in every iteration branch-coverage of test suits is increased by applying TLBO. In the graph TS shows Test Suit while on X-axis shows the number of iteration and on Y-axis shows branch coverage ratio.

VII. CONCLUSION AND FUTURE SCOPE

It is apparent from the example that coverage ratio is improving after every pass therefore it is expected to have more optimized test cases by the proposed method. Development of an automated tool to generate the optimized test cases based on the methodology discussed in this paper is under process. Here we are considering Branch coverage ratio only as a fitness function, including this we can use other factor like Close to boundary value, and Likelihood.

REFERENCE

[1] J.Edvardsson, "A survey on automatic test data generation in PSCCSE in Linkoping; pp 21-28 ECSEL-oct 1999

[2] Harsimran Singh, "Automatic Generation of software test cases using genetic algorithm", a thesis in Thapar University Patiala may2004.

[3] Moheb R.Girgis, "Using genetic algorithm and Dominance Concept for generating Reduced Test Data" Informatica 34 pp-377-385, 2010.

[4] J. Kennedy and R. Eberhart, "Particle swarm optimization," In Proc. Of the IEEE Int. Conf. on Neural Networks, Piscataway, NJ, pp.1942–1948, 1995.

[5] M. Dorigo and T. Stützle, "The particle swarm: social adaptation in information-processing system", New Ideas in Optimization, McGraw-Hill, London, pp. 279-387, 1999.

[6] R. VenkataRao, V.D.Kalyankar, "Engineering Applications of Artificial Intelligence" 26 (2013) 524–531

[7] Huaizhong Li, C.Peng Lam, "software test data generation using Ant colony optimization" world academy of science, engg & tech; 1,2007

[8] D.Jaya Mala, V.Mohan, "ABC tester, Artificial Bee Colony based software test suit optimization approach" International journal of software engg, IJSE vol2 no-2 july 2009

[9] Anu Sharma, "Test cost Optimization using Tabu Search", journal software engg & application, pp-477-486, 2010.

[10] Babak Amiri, "Application of teaching learning based optimization algorithm on cluster analysis", J. Basic. Appl. Sci. Res., 2(11)11795-11802, 2012

[11] Km Baby, Praveen Rajan Srivasatava "Automated Software Testing Using Metahurestic Technique Based on An Ant Colony Optimization" Electronic System Design (ISED), 2010 International Symposium pp-235-240, 22 Dec 2010

[12] Andreas Windisch, Joachim Wegener, "Applying Particle Swarm Optimization to software testing" , GECCO'07, july7-11, London ACM, 2007.

[13] M. R. Girgis, "Automatic Test Data Generation for Data Flow Testing Using a Genetic Algorithm". Journal of Universal computer Science, vol. 11, no. 5, pp. 898-915(2005)

[14] M. Roper, I. Maclean, A. Brooks, J. Miller, and M. Wood, "Genetic Algorithms and the Automatic Generation of Test Data",Technical report RR/95/195[EFoCS-19-95] (1995).

[15] R. P. Pargas, M. J. Harrold, R. R. Peck, "Test Data Generation Using Genetic Algorithms", Journal of Software Testing, Verifications, and Reliability, vol. 9, pp. 263-282 (1999).

[16] Jin-Cherng Lin and Pu-Lin Yeh, "Using Genetic Algorithms for Test Case Generation in Path Testing",Proceedings of the 9th Asian Test Symposium (ATS'00) (2000).

[17] Bogdan Korel, Member, "Automated Software Test Data Generation", IEEE IEEE Transactions on Software Engineering. Vol. 16. NO. 8. AUGUST I990.

[18] M.R.Keyanpour, "Automatic Software Test Case Generation", Journal of Software Engineering 5(3): 91-101, 2011

[19] R. Venkata Rao, V.D. Kalyankar, "Parameter optimization of modern machining processes using teaching–learningbased optimization algorithm" Engineering Applications of Artificial Intelligence (Elsevier), Volume 26, Issue 1, Pages 524-531, January 2013.

[20] R.V. Rao, V.D. Kalyankar, "Multi-pass turning process parameters optimization using teaching-learningbased optimization algorithm" Scientia Iranica (Elsevier), Available online 10 January 2013

[21] P. J. Pawar, R. Venkata Rao, "Parameter optimization of

machining processes using teaching–learning-based optimization algorithm" International Journal of Advanced Manufacturing Technology (Springer), DOI: 10.1007/s00170-012-4524-2, 2012.

[22] R.V. Rao, V.J. Savsani, J. Balic, "Teaching–learning-based optimization algorithm for unconstrained and constrained real-parameter optimization problems" Engineering Optimization (Taylor & Francis), Volume 44, Issue 12, 1447-1462, 2012

Maneesh Kumar received his B.Tech degree in year 2011 in Information Technology from the Noida Institute of Engineering Technology, Greater Noida, affiliated to UPTU, Lucknow. He is currently pursuing his M.Tech degree in Computer Engineering from Noida Institute of Engineering Technology. His research interests include software testing.

Rajdev Tiwari is working as Director-MCA at Noida Institute of Engineering Technology, Greater Noida. He is qualified UGC NET (Computer Science), Ph.D. (Computer Science), MCA, B.Sc, M.Sc. and Post Graduate Diploma in Advance Software Design & Development. Dr. Tiwari is a mult-faceted personality with rich corporate, government and academic experience.

Rajeev received his B.Tech. degree in Information technology from the NIET, Gr. Noida in 2011. He is currently pursuing his M.Tech. degree in Computer Engineering from Noida Institute of Engineering Technology. His research interest area is Data mining.