A Report On

Teaching Learning Based Optimization Model to Generate Software Test Cases

National Institute of Technology,

Warangal Dept. of Computer

Science and Engineering

By

Sandeep Jha

M.Tech CSIS

Roll No : 197569

(2019-21)

M.Jha 02/03/2021

TABLE OF CONTENTS

Chapter No.	Topics	Page No.
Chapter-1	**Introduction**	**3-4**
	1.1 Problem with manual testing	3
	1.2 Need of automated testing	3-4
	1.3 Need of optimization in automated testing	4
Chapter-2	**Literature Survey**	**5**
Chapter-3	**Observation and Motivation**	**6**
Chapter-4	**Problem Statement**	**7**
Chapter-5	**Proposed Approach**	**8-10**
Chapter-6	**Proposed Model**	**11**
Chapter-7	**Implementation**	**12-15**
	7.1 CFG of program	12-14
	7.2 Fitness Function	15
	7.3 Optimization	15
Chapter-8	**Results**	**17**
Chapter-9	**Future Work**	**20**
Chapter-10	**References**	**21-22**

1. Introduction

Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is detect free. Software can be tested using manual testing or automated testing. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements.

Test case generation in software testing is the process of identifying 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.

In the automated testing process,The generation of multiple-path test cases can greatly enhance the efficiency of path-wise testing. However for quality testing, coverage of critical path(a path is critical if the probability of covering the path is low) is more important than percentage of code coverage.

To improve efficiency, a test case generation method for multiple-path coverage is proposed in this report which will emphasize on covering the critical path rather than achieve maximum coverage of the code. We propose a fitness function(Branch Distance Function), using Teaching Learning Based Optimization(TLBO) to generate test cases automatically based on the path coverage criteria.

1.1 Problem with Manual Testing

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

It covers functionality only at human speed.

The major investment is done in the process itself but not in error detection.

This process is limited by resource availability and not by necessity.

The tests can be repeated inexactly.

There is no regression testing

1.2 Need for Automated Testing

There are several benefits of automatic testing over manual testing. Some major benefits are:

It’s almost 40 times faster and more accurate than the manual testing.

Automation tools have better access to data, objects and operating system files than manual tools.

Once tests are developed, the number of next tests can be increased as the application matures. Engineers can constantly add onto test suite and not have to test the same functionality over and over again.

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

1.3 Need for Optimization in Automated Testing

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.

The test software requires a wide range of test data. Random generators generate a large amount of test data. But testing with these random data involves a lot of human effort, costs and time. In order to verify the correct execution of a software, in order to satisfy the customer requirements obtained, a tester must optimize the test data set by applying optimization techniques such as Teaching Learning Based Optimization(TLBO).

2. Literature Survey

Vigorous research has been carried out III the automatic test case generation for number of years. In 1990 an alternative approach of test data generation was proposed 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 [1] while a hybrid method called GA-NN was proposed which was then considered and studied in order to understand when and why a learning algorithm is effective for testing problem [2].

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 [3]. Another optimization algorithm is Ant Colony Optimization is used for test data generation [4] [5].

Using Tabu Search, an efficient cost effective approach for optimizing the cost of testing was proposed[6]. Particle swarm optimization (PSO) has been applied successfully on generating test cases [7]. In 2005 an automatic test data generation technique that uses a genetic algorithm (GA) was proposed[8]. The most widely used meta heuristic 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[9],[10],[11].

Recently, the use of Teaching Learning Based Optimization in optimization became the focus of several research studies [12][13][14][15].TLBO on cluster was applied successfully[16].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, to check whether TLBO works better than other optimization techniques for test case generation, in this project we’ll apply TLBO on automated test case generation in this project.

3. Observation and Motivation

For a large project, a large number of test cases are required. Generating optimal test data reduces that set of tests.

Generating a reduced set of optimal test data reduces the time and costs involved in the test.

Automation of the test data generation process minimizes the work required for manual testing.

Many researchers have generated new technologies and methodologies to generate test data, but they never suggest which technology is better. The work focuses on comparing the result of proposed method with similar study.

4. Problem Statement

A critical path is a path with low probability of coverage at the time of automatic test case generation.

In the process of automated test case generation, if the aim would be maximum percentage of coverage only, then a critical path may be skipped, thus generated test cases won't be effective.

If the critical path will be covered by generated test cases even if total coverage is less percentage of coverage, then this would not have much impact on quality test case generation.

Our objective is to cover the critical path using two approaches :

1. TLBO with CFF(Combined Fitness Function using Approach Level and normalized Branch Distance Function together)
2. TLBO with AL(Approach Level) alone as the fitness function

Then compare the results of both the approach to check which one is better and then comparing the result of better approach between two with the already proven methods like PSO and APSO with different type of fitness function.

5. Proposed Approach

We have to implement a program to generate the test cases for the given source code using Branch Distance Function and TLBO and check the result of experiment with other optimization techniques.

The Various techniques and algorithm that we'll use will be:

CFG(Control Flow Graph)
Fitness Functions((1)Approach Level function and (2)a combination of Approach level with normalized Branch Distance Function)
TLBO Algorithm

1. Control flow graph (CFG):

The CFG of a programme F is a directed graph

$$G = (N, E, s, e)$$

where N is a set of nodes, E is a set of edges, and s and e are unique entry and exit nodes in the graph.

Each node $n \in N$ is a statement in the programme, with each edge $e = (ni, nj) \in E$, representing a transfer of control from node ni to node nj. Nodes corresponding to decision statement (such as 'if' or 'while' statements) are referred to as branching nodes; the outgoing edges of these nodes are referred to as branches.

The condition determining whether a branch is taken is termed the branch predicate.

2. Fitness Function:

The key step in converting test cases generation into an optimization problem is the construction of fitness function. We use Branch Distance Function and Approach level together as the Combined Fitness Function to compute fitness value.

And we also take the Approach Level alone as the second Fitness Function and comapare it with the Combined Fitness Function.

3. Teaching Learning Based Optimization(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.

Learners also learn from the interaction among themselves which also helps in improving their results

There are two phases in TLBO:

Teacher Phase : During this phase a teacher emphasizes to increase the mean result of the entire population

Student Phase : Learners increase their knowledge by interaction among themselves.

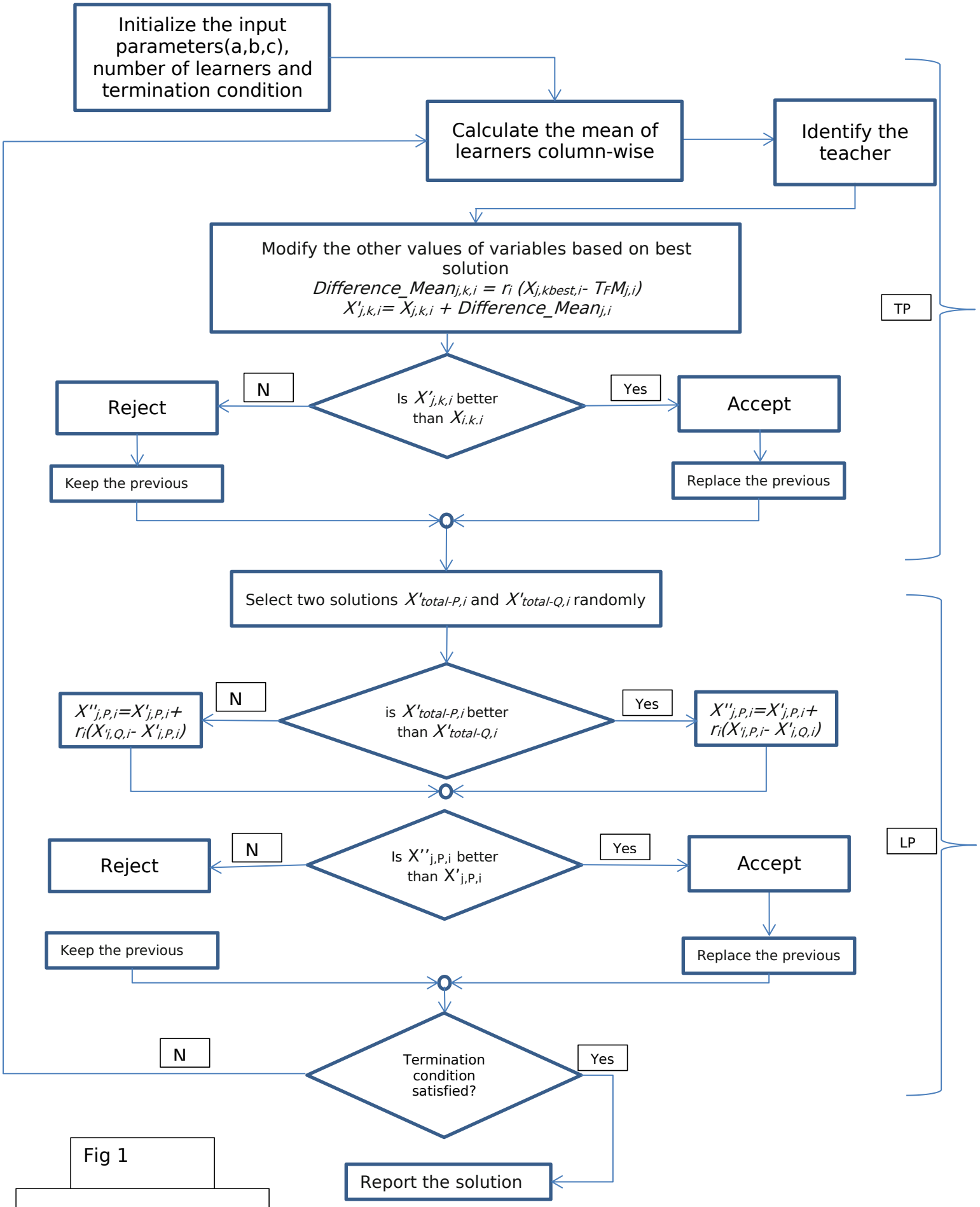

Fig 1

Working of TLBO:

6. Proposed Model :

1. In this proposed work, a source program(Triangle Classification Problem) will be taken as input.
2. A control flow graph(CFG) corresponding to given source program will be generated.
3. CFG of given source code and probability of path testing coverage will be combined.
4. On the basis of number of variables in the source program test cases will be generated randomly.
5. A test case will be a tuple of 'n' values if there are 'n' variables in the program , m number of test cases will be generated and clubbed together to form a Test Suit(TS).
6. Initially 'k' number of test suits will be taken to start the optimization process.
7. In the next step Fitness function of each 'k' number of test suits will be computed.
8. On the basis of fitness value TS with minimum fitness value will be declared as teacher and other test suits as students.
9. After which TLBO is applied to optimize the fitness value. 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.

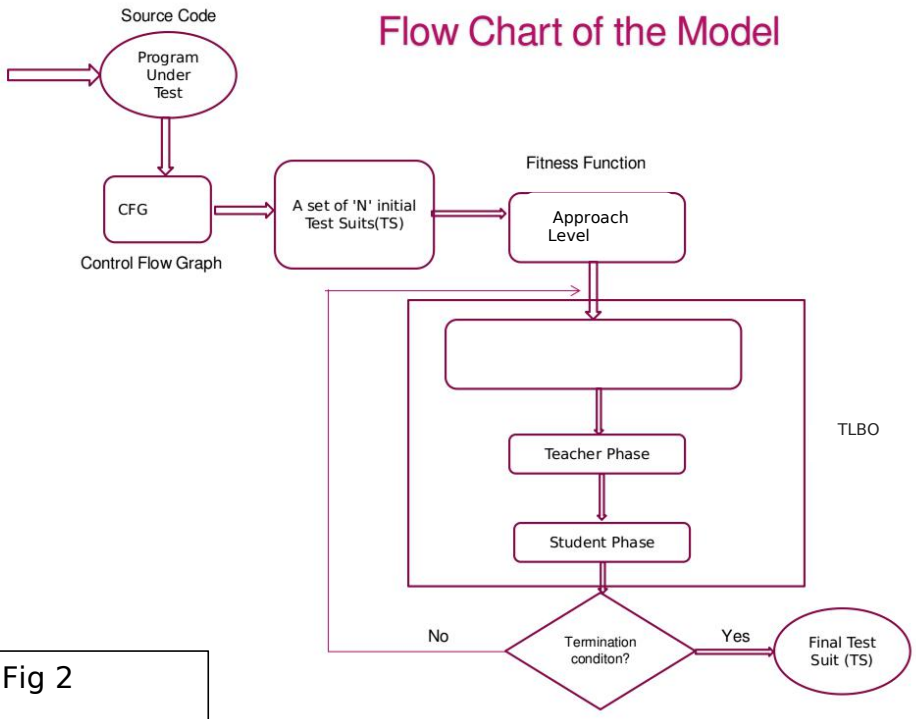

Fig 2

Flow Chart of the above Model

7. Implementation :

1. Take Triangle-classification program as the input and generate its CFG

```
    double Trig_Area(int a, int b, int c)
 {
     int match; double s=0,h,p;
1 If (a+b>=c && b+c>=a && a+c>=b && a>0 &&
      b>0 && c>0)
2       {match=0;
3       If (a==b)
4          match=match+1;
5       If (a==c)
6          match=match+2;
7       If (b==c)
8          match=match+3;
9     If (match==0)      {
10-11    p=(a+b+c)/2; s=sqrt(p*(p-a)*(p-b)*(p-c));
12         printf("Anomalistic\n");}
13      else if (match==1) {
14-15        h=sqrt(pow(a,2)-pow(c/2,2));s=c*h/2;
16           printf("Isoceles,and First=Second\n");}
17      else if(match==2) {
18-19        h=sqrt(pow(a,2)-pow(b/2,2));s=b*h/2;
20           printf("Isoceles,and First=Third\n");}
21      else if(match==3){
22-23        h=sqrt(pow(b,2)-pow(a/2,2)); s=a*h/2;
24           printf("Isoceles,and Second=Third\n");}
25      else {
26-27        s=sqrt(3)*a*a/4;printf("Equilateral\n");}}
28        else printf("Not a Triangle!\n");
e            return s;
             }
```

Fig 3 Triangle Classification Problem

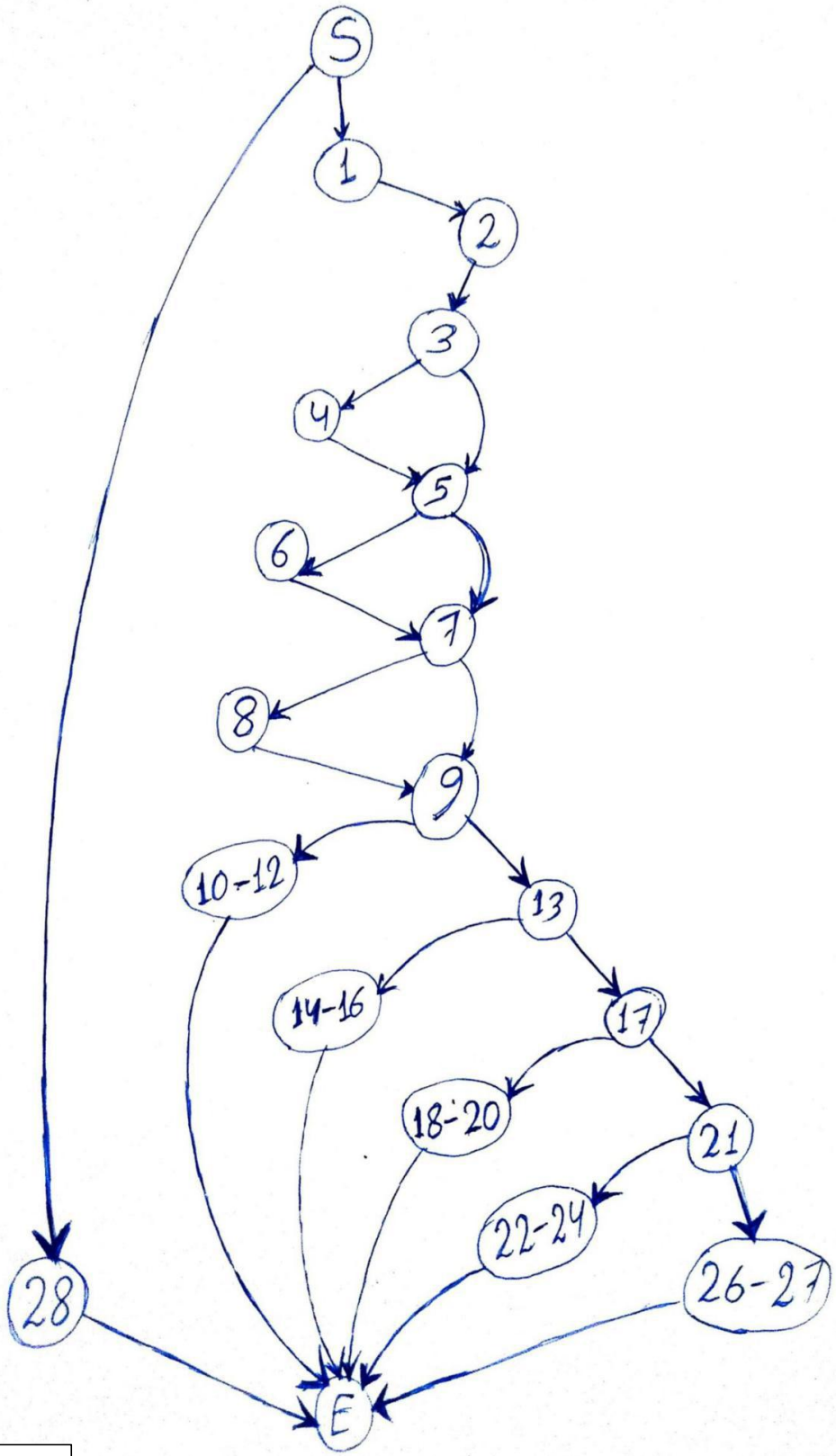

Fig 4 CFG of the above program

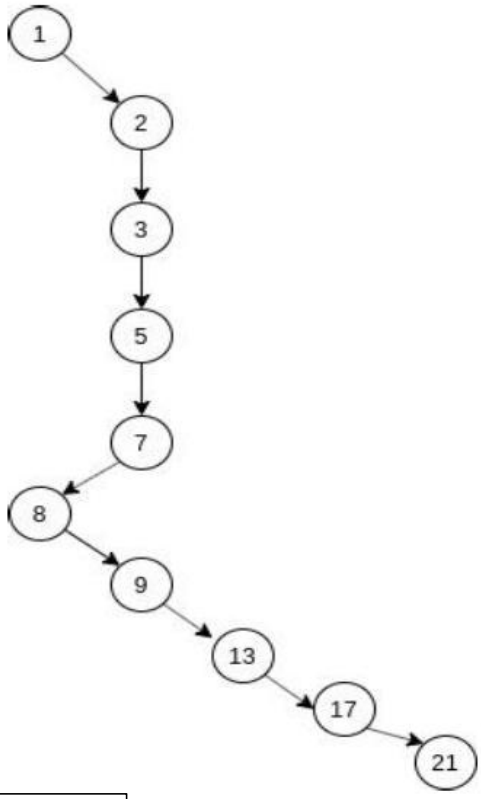

Fig. 5.a

Target Path

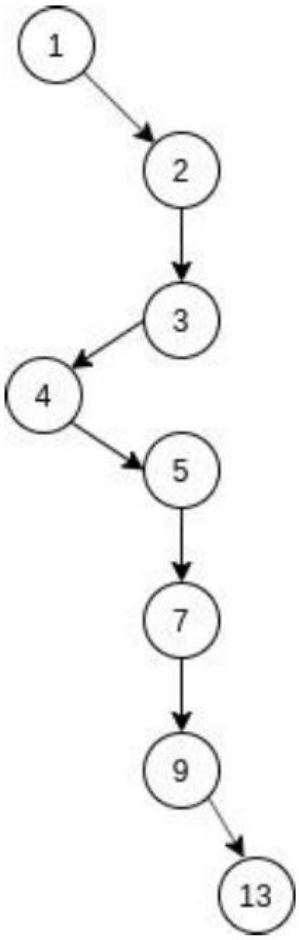

Fig. 5.b

Traversed Path

Suppose path E is selected as the target path and a test case t={4,4,5}, then according to the test case traversed path is C.

The Combined Fitness Function is computed as:
The **fitness function** FF(A,B,C) of input (A,B,C) to the target path P is defined by formula below:

FF(P (A,B,C) , P) = AL(P (A,B,C) , P) + normalized(BD(P (A,B,C) , P))

Suppose that the traversed path of input (A,B,C) is P(A,B,C) and the target path is P. The **Branch Distance** (BD) between P(A,B,C) and P is the sum of the BD of those branch predicates that exist in both paths. Thus, the formula can be defined as follows:

BD(P (A,B,C) , P) =Σ BD(Pr)

The **Approach Level** (AL) of the input (A,B,C) to the target path P is defined as AL(A,B,C), which is the number of mismatched branch predicates both included in the traversed path P(A,B,C) and the target path P.

The fitness value of the test case t ={4, 4, 5} is 3.0013 for target path E. Similarly, we can compute the fitness value of any test case sample with respect to given Target Path.

Optimization using TLBO :

After generating specified number of test cases randomly, we apply TLBO to optimize our test-case suit to effectively test the given input program.

One example of the optimization of following set of 10 test cases:

Initially generated set of 5 test cases

Table 1.a

Side 1	Side 2	Side 3	Fitness Value
2	1	2	3
1	2	2	0
1	1	2	3
1	1	1	3
2	1	1	0

As the test case (1,2,2) has minimum fitness value of 0.0, it will be teacher for next iteration

*** Target path given : E
So, the algorithm will try to optimize test cases in such a way that side 2 = side 3 but side 1 != side 2

After Teacher Phase of iteration #1

Table 1.b

Side 1	Side 2	Side 3	Fitness Value
1.8	**1.4**	**2.2**	**1.0**
1	2	2	0.0
0.8	**1.4**	**2.2**	**1.0**
0.8	**1.4**	**1.2**	**1.0**
2	1	1	0.0

After Learner Phase of iteration #2

Table 1.c

Side 1	Side 2	Side 3	Fitness Value
1.8	1.4	2.2	1.0
1	2	2	0.0
0.8	**1.4**	**1.4**	**0.0**
0.8	1.4	1.2	1.0
2	1	1	0.0

As the test case (1,2,2) has minimum fitness value of 0.0, it will be teacher for next iteration

After Teacher Phase of iteration #2

Table 1.d

Side 1	Side 2	Side 3	Fitness Value
1.8	1.4	2.2	1.0
1	2	2	0.0
0.8	1.4	1.4	0.0
0.8	1.4	1.2	1.0
2	1	1	0.0

After Learner Phase of iteration #2

Table 1.e

Side 1	Side 2	Side 3	Fitness Value
1.8	1.4	2.2	1.0
1	2	2	0.0
0.8	1.4	1.4	0.0
0.8	1.4	1.2	1.0
2	1	1	0.0

As there is no more change in any test case, optimization has been achieved.

8. Results :

We compare the performance of Combined Fitness Function (Approach level and Normalized Branch Distance Function) and Fitness Function using Approach level only, For the same input set of size 100 and size 1000.

Initial path count of initially generated 100 Test Cases with value of triangle sides in range [1-20]:

Table 2.a

A	B	C	D	E	F
44	46	3	4	3	0

Fitness Function using combination of Approach Level and normalized Branch Distance Function

Table 2.b

Target Path		No of iteration	AFV	Optimized test cases falling in path : A	B	C	D	E	F
	A	1	4.0	**44**	56	0	0	0	0
	B	5	0.00144	1	**99**	0	0	0	0
	C	10	1.4212	4	67	**29**	0	0	0
	D	5	2.102	12	53	11	**24**	0	0
	E	11	0.73269	4	69	0	0	**27**	0
	F	6	4.97174	36	31	10	13	9	**1**

Fitness Function using Approach Level

Table 2.c

Target Path		No of iteration	AFV	Optimized test cases falling in path : A	B	C	D	E	F
	A	1	4.0	**44**	56	0	0	0	0
	B	1	0.0	44	**56**	0	0	0	0
	C	1	1.14	50	30	**16**	4	0	0
	D	1	1.36	42	14	9	**32**	3	0
	E	1	0.56	49	26	0	0	**25**	0
	F	2	3.15	51	5	7	13	11	**13**

Initial path count of initially generated 1000 Test Cases with value of triangle sides in range [1-100]:

Table 3.a

A	B	C	D	E	F
507	476	3	8	6	0

Fitness Function using combination of Approach Level and normalized Branch Distance Function

Table 3.b

Target Path	No of iteration	AFV	Optimized test cases falling in path : A	B	C	D	E	F
A	1	4.0	**507**	493	0	0	0	0
B	9	0.00052	7	**993**	0	0	0	0
C	10	0.59	10	294	**696**	0	0	0
D	8	1.0596	23	137	305	**535**	0	0
E	7	0.166285	4	156	0	0	**840**	0
F	6	4.53304	243	319	243	33	31	**131**

Fitness Function using Approach Level

Table 3.c

Target Path	No of iteration	AFV	Optimized test cases falling in path : A	B	C	D	E	F
A	1	4.0	**507**	493	0	0	0	0
B	1	0.0	507	**493**	0	0	0	0
C	1	0.93	493	228	**276**	3	0	0
D	1	1.389	510	113	117	**258**	2	0
E	1	0.478	472	248	0	0	**280**	0
F	2	2.883	411	34	91	134	135	**195**

* AFV=Average Fitness Value of test cases

Comparison of the proposed approach with other models:

*Triangle Classification Problem with population of test cases=1000
And range of value in [1-20]

Table 4

Models	Average number of test cases generated for				Iterations to achieve critical path
	Not a triangle	Scalene	Isosceles	Equilateral	
PSO with existing BDF (Pachauri and Mishra ,2015)	447.33	440.33	109.67	2.67	3
APSO with existing BDF (Pachauri and Mishra ,2015)	451	429.67	116.67	2	3
PSO with existing CFF (Garg and Garg et al, 2015)	464	434	100.67	1.33	3
APSO with existing CFF (Garg and Garg et al, 2015)	490	407.67	100.33	2	2
PSO with the ICF (RR Sahoo and Mitrabinda Ray, 2019)	471.67	417	106.33	5	2
APSO with the ICF (RR Sahoo and Mitrabinda Ray, 2020)	461	421.33	112.67	5	2
Proposed TLBO Model with the Approach Level Function	**495.2**	**399.4**	**102.6**	**2.8**	**2**
Proposed TLBO Model with the Combined Fitness Function	**495.2**	**399.4**	**102.6**	**2.8**	**6**

9. Conclusion and Future Work

We have proposed Combined Fitness Function(Approach level with normalized Branch Distance Function) and Approach level based fitness function that guides the searching techniques to automatically generate test cases for path coverage. Critical path coverage with less number of iterations is our main objective. We conduct experiments on bench mark case study which is TCP(Triangle Classification Problem).

Using TLBO with CFF approach, we can get slightly better average value of Fitness functions than the TLBO with ALF approach. But from the results we can also observe that the convergence ratio of the TLBO with ALF approach is much better than the TLBO with CFF approach.

As the TLBO approach use only two parameters 1.)Population size and 2.)Number of iterations, it is simpler to implement also.

As the convergence ratio of TLBO with ALF approach is almost equal to many previously proposed method and is even better than some methods for TCP, This approach should be tested on other problems like RCP(Remainder Calculation Problem) and SSSP(Single Source Shortest Path) also to prove that this method can be successfully accepted to generate the test cases for software testing.

Our future direction is to compare TLBO with Approach Level function with other existing models on the bench mark case studies like RCP and SSSP problems and also compare the time factor of our model with other existing models.

10. References

1. Bogdan Korel, Member, "Automated Software Test Data Generation", IEEE IEEE Transactions on Software Engineering. Vol. 16. NO.8. AUGUST 1990.

2. M.R.Keyanpour, "Automatic Software Test Case Generation", Journal of Software Engineering 5(3): 91-10 1,2011

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

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

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

6. Anu Sharma, "Test cost Optimization using Tabu Search", journal software engg & appl ication, pp-477-486,2010.

7. Andreas Windisch, Joachim Wegener, "Applying Particle Swarm Optimization to software testing" , GECCO'07, July 7-11, London ACM, 2007.

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

9. 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).

10. 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).

11. 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'OO) (2000).

12. R. Venkata Rao, YD. Kalyankar, "Parameter optimization of modern machining processes using Teaching Learning based optimization

algorithm" Engineering Applications of Artificial Intelligence (Elsevier), Volume 26, Issue I, Pages 524-531, January 2013.

13. R.Y Rao, YD. Kalyankar, "Multi Pass turning process parameters optimization using Teaching Learning based optimization algorithm" Scientia Iranica (Elsevier), Available online 10 January 2013

14. 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.

15. R.Y. Rao, YJ. 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

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

17. Rashmi Rekha Sahoo, Mitrabinda Ray, “PSO based test case generation for critical path using improved combined fitness function” Journal of King Saud University- Computer and Information Sciences, Volume 32, Issue 4, May 2020, Pages 479-490

18. A. Pachauri, G. Mishra, “Futuristic Trends on Computational Analysis and Knowledge Management” (2015), pp. 49-55

19. D. Garg, P. Garg, “Basis Path Testing Using SGA & HGA with ExLB Fitness Function, Elsevier” (2015), pp. 593-602