ABTLO: A Hybrid Algorithm for Test Suite Minimization

Neeru Ahuja, Pradeep Kumar Bhatia

Deptt. Of Computer Science and Engineering,
Guru Jambheshwar University of Science and Technology
Hisar, INDIA
neeru902@gmail.com

Abstract –
The process of modifying software can sometimes cause certain functionalities to be affected, leading to an increase in the number of test cases required to ensure that the modified software functions properly. This increase in test cases can result in a significant rise in cost and execution time, making it necessary to minimize the size of the test suite. This article utilizes optimization techniques to enhance the effectiveness of a test suite by identifying the smallest possible set of test cases required to cover all statements, while also minimizing the amount of time needed to run the tests. This research proposes a method to minimize test suite size using a combination of two optimization algorithms: the Artificial Bee Colony (ABC) and Teaching-Learning-Based Optimization (TLBO). The TLBO technique also utilizes a chaotic operator during population exploration. These two techniques are complementary to each other, as the former is capable of exploration, while the latter for exploitation respectively. Their amalgamation gives better result as compared to ABC, TLBO, CABC and CTLBO. Here, CABC and CTLBO are techniques where c denotes the chaotic operator. Although previous studies have combined ABC and TLBO with chaotic operators, this study introduces a new approach that merges ABC and CTLBO. The proposed technique is applied to three benchmark programs, and statistical analysis is performed using a t-test and standard deviation. The experiments showed that the proposed technique outperforms the other techniques in terms of execution cost, average percentage of reduced size, and convergence iteration, indicating that the proposed technique is more scalable.

Keywords *- ABC, TLBO, Chaotic map, Test suite minimization; Regression testing.*
DOI Number: 10.48047/NQ.2022.20.13.NQ88476 **Neuroquantology 2022; 20(13): 3907-3921**

1. Introduction

Software development requires a number of changes depending upon the demand of users. New test cases are generated as a particular program is tested repeatedly. Software testing requires much labor, and for this reason, people generally avoid it. So it can be said that software testing is a hard task. Regression testing involves the saving of test cases and running the program after some changes in its other components. There is a problem faced by the developers as to which test case is to be included in regression testing. The inclusion of a large number of test cases is not possible, as a small number of test cases may not cover the functionality of the software effectively. The problem can be solved by coverage criteria. The coverage criterion includes the statement, path and fault coverage [2] that will reduce the testing time by selecting a proper subset of the existing test suite.

The selection of test cases involves i) the selection of all test cases, ii) selection of test cases randomly and iii) minimization techniques .Selecting all test cases can be resource-intensive and time-consuming and is therefore not recommended. It is not worth checking all test cases. The second option, selection of test cases randomly, is not a good idea to test software. Randomly similar test cases can be selected for testing, while others may remain in starvation. The last technique is the minimization technique.

Test suite minimization selects a subset of test cases from a set of test cases that provide coverage of the modified area of the program. It aims to tackle the test suite by removing redundant test cases. The selection of the subset of test cases resembles the problem of the set cover problem and is covered under the NP-hard problems. Therefore, it can be said that test suite minimization is also an NP-hard problem that is solved by hundreds of authors.

The objective of the test-suite minimization problem is to find a reduced set of test suite that covers all test requirements. Let us have a test suite T= {t1, t2, t3, …, tm} and a set R= {R1, R2, R3, …, Rn} here, t1,t2…tm represents m test cases, and R1,R2…Rn shows n requirements. Each test case should cover at least one requirement, and a requirement can be covered by at least one test case. For this problem, we have to find the minimum subset T' from T that fulfils all the requirements.

The test requirement will be a coverage matrix. Based on the coverage matrix, a subset of test cases will be selected. Different coverage matrices, such as APSC, APBC, APPC, and APFD, are used as bases to solve the problem.

Many authors have tried to solve it with greedy algorithms, although others have tried with heuristic algorithms [16]. However, it does not always produce a minimal set but yields the optimum result. A plethora of techniques exist to optimize the test suite. Nature-inspired techniques are problem-solving tactics and have been gaining much reorganization for their competence [4]. These techniques are optimization approaches that mimic natural phenomena to solve complex problems. They are popular not only in engineering but also in medicine, effort estimation, agriculture and others [15].

The following is a summary of this paper's significant contributions:

- The test suite was reduced using a hybrid model (ABC+TLBO) (TSR).
- As far as we know, no one has ever tried to use ABC and TLBO together to solve the TSM problem before.
- We considered the statement coverage as the base criteria for TSR.
- A chaotic logistic map improves hybrid model and algorithm performance.
- Define the different metrics for evaluating performance using benchmark programs.
- Statistical analysis to illustrate the algorithm's efficacy.

The rest of this paper is structured as follows:

In Section 2, the literature study on minimization techniques is explained. In Section 3, the ABC and TLBO techniques are explained. In Section 4, the proposed work is shown. In section 5, details about the experiment are given. Section 6 discusses the results and suggestions for the future.

2. State-of-the-Art

Reducing the number of test cases while maintaining coverage criteria is the goal of the test case minimization technique [3]. In this section, we take a look back at the most up-to-date studies on optimising test suites for maximum coverage.

Xia et al. [5] proposed a multiobjective technique that contains cluster analysis and an evolution algorithm. For the clustering stage k, clustering and genetics are used as evolutionary algorithms. The results of the proposed technique remove unnecessary test cases, detect faults and prove its efficiency compared to other algorithms.

Kiran et al. [6] implemented three variants of the selftunable Adaptive Neuro-fuzzy Inference System. TLBO-ANFIS, FA-ANFIS, and HSANFIS for multiobjective optimization. They used execution history as input, which contains requirement coverage, execution time, fault detection rate and impact of requirement failure. For evaluation of the proposed technique, two-benchmark test suites are used, and they calculate statistical results. A hybrid-adaptive neuro-fuzzy inference system was also implemented by Anwar et al. and removes up to 48% of unnecessary test cases while compromising the fault detection rate [7].

A framework called RDUNET was introduced by Mongiov et al. [30], it uses Randoop to automatically generate test cases, assesses code coverage, and decreases the test suite. By treating reduction as a set cover problem, their method incorporated it into integer linear programming and network-based optimization.

The Pareto optimal solution was proposed by Yoo and Harman [8] for the reduction of the test suite. They considered coverage and fault detection as objective functions.

For the reduction problem, Gaurav Kumar et al. [10] developed a neuro-fuzzy technique, and the effectiveness of the algorithm was defined by the execution time. Together, neural networks and fuzzy logic form a potent tool for reducing and prioritising test cases by combining expert knowledge with numerical data."The proposed neuro-fuzzy model has been derived from many existing approaches to obtain the benefits of parent models. The model was validated through data obtained from the PROMISE Software Engineering Repository of NASA projects. The proposed model has been implemented in MATLAB R2013 using ANFIS (Adaptive Neuro-Fuzzy Inference System) with a hybrid learning algorithm of the least-squares method and backpropagation gradient descent".

Another nature-inspired technique, namely, ACO, was implemented to achieve the smallest possible test case while maintaining criteria like maximum fault coverage and minimize execution time [11].

A new hybrid Q-learning sine cosine algorithm (QLSCA) [13] was proposed for the combinatorial test suite minimization problem. The authors integrate the new two operators to enhance its searching capability.

Recently, Sivaji et al.[14] introduced African Buffalo-based convolution neural slicing (AB-CNS) for minimizing test cases, and simulation of the technique was performed on Python software. The effectiveness of the technique was measured by attaining 99.32% accuracy, 20 s execution time, recall value 98.86 and utilizing low memory. The results were compared with AGC, FEST and ANN-Whale, and the proposed technique outperformed the existing techniques.

Malhotra and Khari [18] proposed an improved ABC algorithm to solve the minimization problem of test suites. To improve the global search capability, the author added a mutation operator of the genetic algorithm with onlooker bees and scout bees.

All of these techniques were developed to solve the minimization problem of the test suite. Our review of the literature revealed that most authors adopted fault coverage as the coverage criterion for reducing the test suites, whereas only a small number of authors preferred requirement coverage. To the best of our knowledge, no one has previously attempted to use the combination of ABC and TLBO to solve the TSM problem. In this research, we developed the integration of ABC and TLBO for TSM to cover the gaps by using requirement coverage as the coverage criterion.

3. Basic of ABC

ABC, a swarm intelligence optimization technique proposed by Karabogain et al. [21], is known for its ability to optimize problems using a small number of control parameters. ABC can be classified into the following three categories [9,27]: i) Employee, ii) Onlooker and iii) Scout bees. Similar to the aim of honey bees to find nectar, ABC finds the optimized solution of the problem.

Employee Bee Phase: To determine a new source of honey based upon previous information. It consists of the following steps:

- Update variable and partner for each food source using eq. 1

$$Xnew = X + \emptyset(X - Xp) \text{ .... } (1)$$

- Update variable
- Apply the greedy algorithm if no updating increases the trail counter by 1.

Onlooker Bee Phase: Employee honeys share this information with onlookers. Before the onlooker bee phase probabilities have to be evaluated using eq. 2

$$p_i = f_i \,/ \sum_{i=1}^{n} f_i \qquad (2)$$

The following steps are used to execute the onlooker bee phase:

- Select random variable r
- check if r< p
- Update food source based on random variable and probability

Scout Bee Phase: Randomly search a new source of honey near the hive. Bees size in employee and onlooker bees are equal. The pseudo code of the ABC algorithm is shown in Fig. 1.

Basic of TLBO

Rao et al.[22] proposed an algorithm based upon teacher-learner interactions that provides the impact of teaching on the output of learners. It is a population-based optimization algorithm that does not require any specific parameter as required by other optimization algorithms [17]. Its advantages are its simplicity, ease of implementation without any specified parameters and its rapid convergence. TLBO consists of two phases namely teacher and the

Algorithm 1. Artificial Bee Colony

```
Begin
   Define max_generation , max_population
   Initialize food source(population)
   Evaluate fitness function
   For i=1: max_generation
       For i=1: max_population
            Employee Phase
            Calculate probabilities
            Onlooker Phase
            Memorize the best solution
            Scout Phase
       End for
   End for
End
```

Fig. 1. Steps of the ABC Algorithm

learner phase. A teacher phase comprises the best solution of the optimized set, which is referred to as a teacher. The teacher shares his knowledge of the subject with his learners to increase their output.

Teacher Phase: A good teacher is believed to bring the students to the same level as himself or herself. However, in practise, this cannot rest solely on the shoulders of the teacher. Let a teacher make an effort to raise the mean of its class to a new level M_{new}.

Assume the best one is teacher $\mathbf{Y_{teacher}}$. Calculate the mean($\mathbf{Y_{mean}}$) of the population.

Update the position of each learner by eq. 3

$$Y_i^{New}=Y_i^{Old} + rand(Y_{teacher} - T_f * Y_{mean}) \text{ .... } \qquad (3)$$

Here, T_f represents the teacher factor whose value can be 1 or 2 and is calculated as T_f= round[1+ rand(0,1)].

Mean($\mathbf{Y_{mean}}$) can be calculated by using eq. 4

$$y_{mean} = \frac{\sum_{i=1}^{np} y_i}{np} \text{ .... } \qquad (4)$$

Learner phase: A learner gains knowledge either from the teacher or by interaction with other learners through formal communication, presentation or group discussion. Suppose initialize the population of size n and randomly select two learners yi and yj where i≠ j execute this phase after the teacher phase.

Update the learner using eq. 5 and 6

$Y_{new,i}=Y_{old,i}+r_i(Y_i-Y_j)$ if $f(Y_i)<f(Y_j)$…. (5)
$Y_{new,i}=Y_{old,i}+r_i(Y_j-Y_i)$ if $f(Y_j)<f(Y_i)$…. (6)
The pseudo code of the algorithm is explained in Fig. 2.

Algorithm 2. Teacher Learner Based Optimization

```
Begin
  Define max_generation , max_population
  Initialize random population
  Evaluate fitness function
  For i=1: max_generation
     For i=1: max_population
          Teacher Phase
          Learner Phase
          Update teacher and mean
     End for
  End for
End
```

Fig. 2. Steps of the TLBO Algorithm

Chaos Theory

Exploration may be defined as the tendency of the algorithm to have extremely random behavior. A greater number of chances of significant exploration will be the result of the problem. However, in exploitation, the solutions tend to search in local space. Hence, good algorithms balance exploration with exploitation. Due to the stochastic approach, balancing these two is difficult[25]. Both of these concepts, viz., exploration and exploitation, have a great disagreement between them in which they tend to strengthen one result and weaken the other. Therefore, none of them can uniquely emphasize better outputs.

Chaos theory addresses the study of chaotic dynamical systems. These systems are nonlinear systems that are vastly rational to their initial conditions. A small change in the initial condition may lead to high variation in the outcome of the system. Prima facie, a chaotic system, behaves randomly, but randomness is not a necessary condition for the system to show chaotic behavior. For these merits, such techniques have been utilized by researchers to increase the performance of nature-inspired algorithms.

Kuang et al. [24] proposed a self-adaptive chaotic ABC algorithm that was based on tent maps. Mirjaali et al. [25] embedded 10 chaotic maps with GSA to improve its exploitation and exploration. Ding et al. applied the chaotic bee colony approach to system identification of a small-scale unmanned helicopter state-space model in hover conditions[23]. Farah et al. [26] improved the convergence performance by using a logistic chaotic map after the teacher and learner phases. Considering the results of these studies, we decided to incorporate the Chaotic function into our ABC and CTLBO algorithms. We adopted the logistic map function because of its simplicity and based on the results of the literature survey. The next section presents the proposed algorithm.

4. Proposed Algorithm

The hybrid algorithm is an amalgamation of swarm intelligence optimization and population-based algorithms. The ABC algorithm has good exploration capability, but it suffers from exploitation because employee bees and onlooker bees use the same mechanism in the search process. To solve this issue, we embedded the teacher and learner phases of the TLBO algorithm into the employee and onlooker bee phases with a chaotic logistic map. TLBO is good in exploitation, not in exploration. Both techniques are complementary to each other. The flowchart and framework of the proposed algorithm are shown in Figs. 3 & 4.

Initialization

Initialize the number of food sources, i.e., the first step of our process. Food sources (test cases) are used for all algorithmic phases. Food sources are initialized randomly with permutation encoding, which is shown in Fig 5.

Objective function

For each food source in the population, it is very important to assess food source's efficiency. It deals with the measurement of the fitness of a possible solution represented by that food source. The objective function confirms the process's diversion towards its optimization target by permitting the best individual to lead to a good solution. The fitness function of the problem is to cover maximum statement coverage while simultaneously decreasing the total number of test cases.

Total statement coverage [19]

According to P.K. Gupta to determine the total statement coverage, each independent path should be executed at least once during the path testing. The average percentage of statement coverage (APSC) formula is defined in equation 7.

Maximize *APSC= 1- $\sum_{i=1}^{n} scov_i/nm$+1/2n............. (7)*

Fitness Function

The fitness function is measured by using eq. 8

Fit(f(x))= 1/(1+f(x)) , f(x) >=0
1+abs(f(x)) , f(x)<0 .... (8)

Here, f(x) represents the objective function, i.e., calculated in eq. 7

Proposed Algorithm operator

Teacher-based employee bee phase

In the employee bee phase, the bee is generated using eq. 1. However, in the proposed algorithm, a new searching strategy is adopted specifically based on the teacher phase of the TLBO algorithm. The best teacher is selected, and the teacher tries to bring his or her learner up to his or her level

```
Hybrid Artificial Bee Colony
Initial food source.
iteration=1
Evaluate Fitness function
Employee bee Phase
        Calculate mean by using equation (4)
        generate new bee according to equation (10)
        calculate fitness function f(xi)
        if new bee is better than replace old ones.
        Apply greedy algorithm if no updation increase trail counter by 1.
Evaluate tournament selection probabilities (p) using eq 2.
Onlooker bee Phase
        generate new bee using equation (11) and (12)
        calculate fitness function f(xi)
        if new bee is better than replace old ones
        Update food source based on random variable and probability
Memorize the best solution
Scout Phase
        if trail counter >limit apply scout phase
        evaluate fitness and set trail.
iteration =iteration+1
Stop when reach to max iteration.
```

Fig. 3. Flowchart of the proposed algorithm

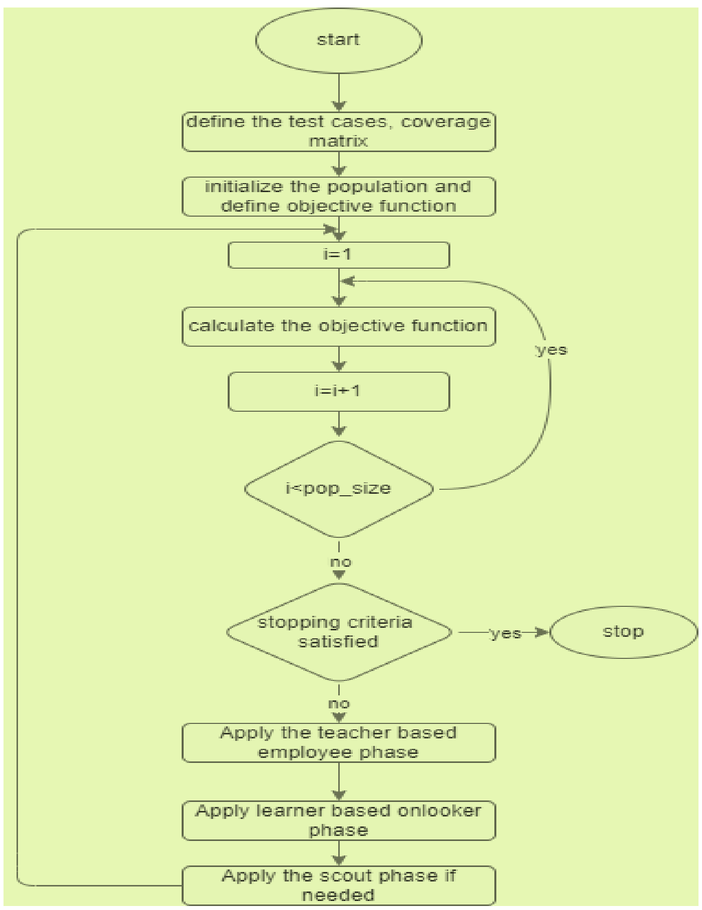


Fig. 4. Framework of the proposed algorithm

in terms of knowledge. Similarly, the best bee is selected, and new food sources are formulated using eq. 3 with the addition of the logistic map that is represented in eq. 9.Eq. 10 represent the way to generate new bees.

$x_{m+1} = \alpha x_m (1 - x_m)$…………………. (9)

Here, x, m and α represent the chaotic variable, number of iterations and constant value between [0,4]. While calculating the x value, it gives a very small value.

$Y_i^{New} = x_m (Y_i^{Old} + rand(Y_{teacher} - T_f * Y_{mean}))$……… (10)

After the new food source has been generated, the objective function will be calculated and then compared to the previous iterations. If the new food source is not a significant improvement over the previous ones, the trail is zero; otherwise, it increases by one. Select the best bee from the set of populations and calculate the probability using eq. 2.

Tournament Selection

Roulette wheel selection is based on a proportional selection strategy in which the best wheel can be selected or not. To eliminate this issue, we selected tournaments. A tournament is a competition between a number of individuals of known sizes, which is selected randomly. Winner is decided based on the fitness value of individuals. The individual who obtained the best fitness value is selected as shown in Fig. 6. The number of individuals in the tournament is known as the size of the tournament.

Learner-based onlooker bee phase

The tournament method is used by an onlooker bee to choose a food source through a probability-based selection process. Onlooker bees continue to explore for new food sources utilizing chaotic operator x_m and learner phase equations 5 and 6 of TLBO.

$$y_i^{new} = x_m(y_i^{old} + r_i\,(y_i - y_j)\;\; if\;\; f(y_i) < f(y_j) \ldots..(11)$$

$$y_i^{new} = x_m(y_i^{old} + r_i\,(y_j - y_i)\;\; if\;\; f(y_i) > f(y_j) \ldots..(12)$$

Scout phase

If the food source of employee bees cannot be updated after a certain number of limits, employee bees become scout bees. The new random scout bee will be generated similar to the employee bee phase. If the new bee solution is superior to the previous one, update it.

5. Experimental Evaluation and Results

In this section, the effectiveness of our proposed approach is evaluated with respect to benchmark programmes. The details of the evaluation metrics, datasets used and experimental results are presented below:

Evaluation Metrics

To evaluate the proposed algorithm, We used structural coverage measures, which have been studied for a long time as a way to evaluate how well a test suite works.

Convergence iteration[1]: The number of iterations in which the best coverage is found was achieved.

Average Execution Time: Average execution time for all the statements to cover test data.

Average percentage of reduced size: It is defined as the percentage of reduced size of test suite size to the original test suite.

APRS= ((Original Test suite Size –Reduced Test suite Size)/Original Test suite Size)*100

Dataset

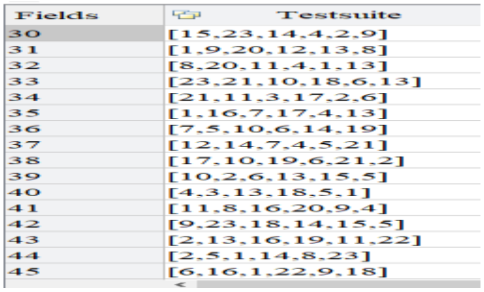
Experimental setup, we have considered the three benchmark programs triangle classification problem, quadratic equation and largest of three numbers.

Description of dataset: these programs are taken from Singh [28].

Triangle classification program: this problem can be solved by knowing the value of the sides of triangle. If all the sides are equal, it will be an equilateral triangle. If two sides are same, then it will be isosceles and otherwise it will scalene. In a similar way, a triangle may be right angle, acute angle or obtuse angle triangle.

Quadratic equation: The quadratic equation, denoted by the formula ax2 + bx + c =0, is a polynomial equation of degree2, with the coefficients variables a, b, and c, and here a≠0.The quadratic equation has two roots, which may be equal, real or imaginary.

Largest of the three numbers: Program defines the three variables and decides which one is greater among three variables.



Fields	Testsuite
30	[15,23,14,4,2,9]
31	[1,9,20,12,13,8]
32	[8,20,11,4,1,13]
33	[23,21,10,18,6,13]
34	[21,11,3,17,2,6]
35	[1,16,7,17,4,13]
36	[7,5,10,6,14,19]
37	[12,14,7,4,5,21]
38	[17,10,19,6,21,2]
39	[10,2,6,13,15,5]
40	[4,3,13,18,5,1]
41	[11,8,16,20,9,4]
42	[9,23,18,14,15,5]
43	[2,13,16,19,11,22]
44	[2,5,1,14,8,23]
45	[6,16,1,22,9,18]

Fig. 5. Initialization of population

Fig. 6. Tournament selection

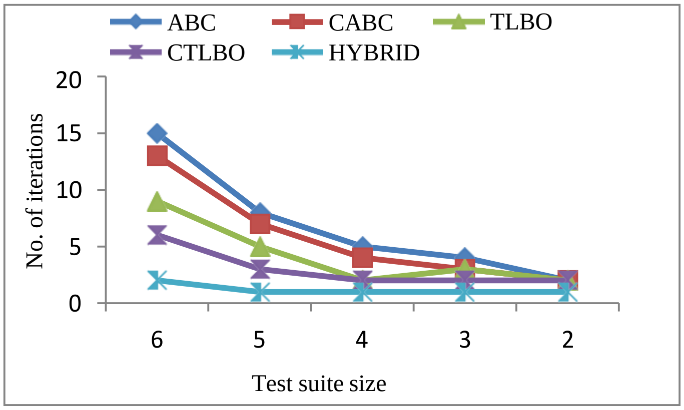


Fig. 7. Test suite size vs no. of iterations

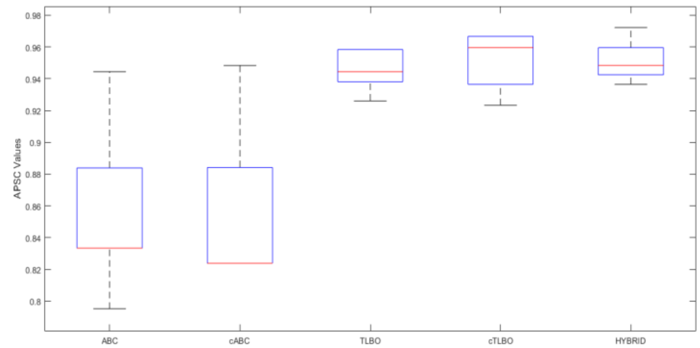


Fig. 8. APSC

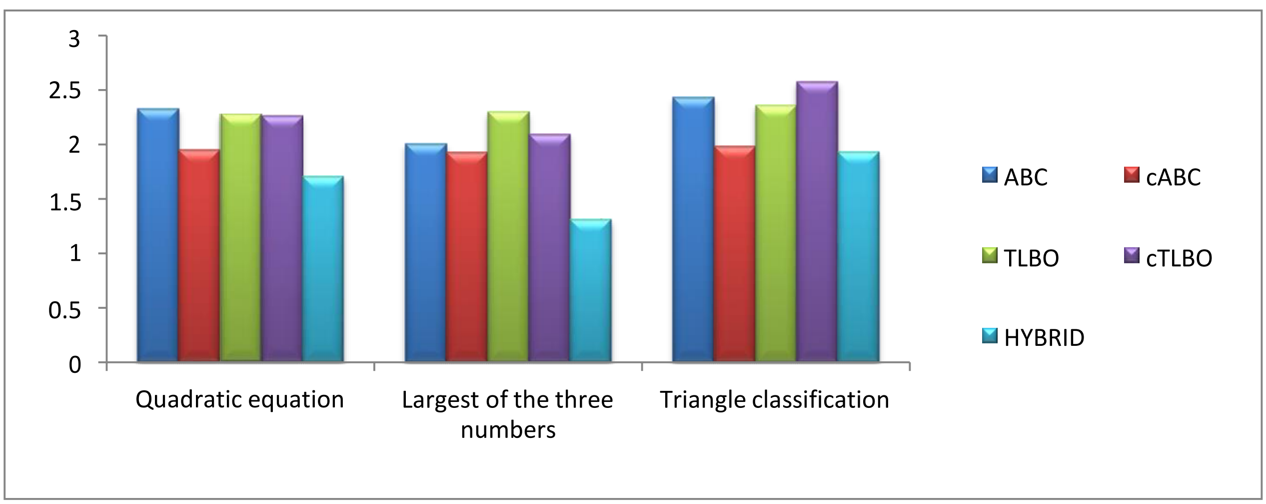


Fig.9. Execution time for statement coverage

Computational Analysis

The results based upon the analysis of our proposed algorithm are described under this section. For experimental purpose, a population size of 50 and the decision variable of varying size were taken into consideration. The algorithm terminates after twenty generations. The parameters of ABC, cABC, TLBO, cTLBO and Hybrid are illustrated in table 1.

Effectiveness of program is measured by APSC, Convergence iteration and reduced size percentage while efficiency can be measured by execution cost.

Effectiveness: Proposed algorithm covers 97.22 % statements in 6 iteration that is highest coverage percentage from all other algorithms. Proposed algorithm perform best than others, cTLBO had second position from five algorithms. ABC and cABC have minute difference in coverage. Table2, fig 7 and 8 shows that all algorithms perform well while in case of iteration ABC gives worst result comparatively with others. The results are clear from table 2 that performance decreases in the order HYBRID, cTLBO, TLBO, cABC, ABC. To show effectiveness we have also performed reduction in size of test suite, which is demonstrated in table 3. As it is evident from table 3 cTLBO and proposed have equivalent results for largest of three numbers and quadratic equation. However, proposed technique performed well in all three programs.

Efficiency: The efficiency of proposed algorithm has been assessed by measuring the time for finding the reduced test suite. Average execution time is used to evaluate the for test suites to cover the maximum statements. Fig 9 and table 3 shows average execution time for statement coverage.

Total execution time refers to time utilized for execution of each instruction in the whole algorithm. Table 4 & fig 10 illustrate the total execution time. As it is evident from fig. 9, 10 and table 3, 4 that triangle classification program takes more time than other programs due to more convergence iterations. From table3, 4 it is clear that proposed technique is more efficient for computational cost than ABC, cABC, TLBO and cTLBO. Therefore, the cost-effectiveness analysis has shown that the proposed algorithm proves itself better than that of the other nature-inspired algorithms.

Statistical Analysis

We have carried out a statistical test to analyze the performance of the proposed algorithm. In view of that, we prepared the research questions and related hypotheses shown in table 5. Here, H0 represents the null hypothesis, and Ha indicates the alternative hypothesis.

RQ1 : To answer this research question we used t-test. Table 6 presents the result of t-test which illustrate mean difference of statement coverage and significance difference of staement coverage. If significance difference of t test (as shown in column 4 in table 6) is less than .05 then we reject null hypothesis and the difference between the algorithms is statistically significant. We compared every algorithm with other and row 1 demonstrate the result of proposed algorithm. It is evident from result that there is significance difference between proposed and ABC , Proposed and cABC algorithm. From statement coverage pont of view proposed algorithm is better than ABC and cABC.Table7 represents the standard deviation value of each algorithm. Effectiveness of algorithm can be measured by lowest standard deviation. Lowest standard deviation means algorithm have more stability. TLBO got first rank in race of statement coverage by leaving behind hybrid technique on second position. There is little difference between hybrid and cTLBO.

RQ 2: RQ2 is defined to find any difference between algorithm w.r.t convergence iteration. To investigate the answer to this question t test has been conducted. Table 8 shows result that cTLBO performance is significantly different from other algorithm. There is no significant difference in the performance of ABC, cABC, TLBO and HYBRID. To reveal difference among algorithms Standard deviation has been computed and presented in Table 9. As we said above lowest standard deviation is best i.e. hybrid in this case. The best performance was of HYBRID, followed by the cTLBO. It can also be noticed that there is no sigificance difference of proposed with TLBO,cTLBO.

6. Conclusion

In the present paper we put a tenacious effort in attempt of reducing size of test suite using statement coverage as base criteria .Many techniques have been explained and executed for minimizing test suite size. For this particular purpose, we have implemented amalgamated ABC and TLBO techniques. The efficiency and effectiveness have been determined by execution time, APSC and convergence criteria. A comparison has been done between proposed technique and ABC, TLBO and variants of both. The statistical analysis of results has shown that the propound methodology is more extensible than others. In the real world scenario where we have multiobjective to solve at a time this efficacious advocated model can be flourishingly used to solve multiobjective optimization problem. The dominance rule can be introduced in the situation. Dominance rule is defined in [29] eq.13

$(x1) \leq f(x2)$................................ (13)

According to eq. if solution of first objective i.e. x1 is not larger than second objective x2 for all objectives and x1 is smaller than x2 in at least one objective then we can conclude and thus express in nutshell that solution x1 dominate x2.

References:

[1] Pandey, A., & Banerjee, S. (2019). Test suite optimization using firefly and genetic

algorithm. International Journal of Software Science and Computational Intelligence (IJSSCI), 11(1), 31-46.

[2] Rothermel, G., Harrold, M. J., Ostrin, J., & Hong, C. (1998, November). An empirical study of the effects of minimization on the fault detection capabilities of test suites. In Proceedings. International Conference on Software Maintenance (Cat. No. 98CB36272) (pp. 34-43). IEEE..

[3] Djam, X. Y., Blamah, N. V., & Ezema, M. E. (2021). A comparative evaluation of test coverage techniques effectiveness. Journal of Software Engineering and Applications, 14(4), 95-109.

[4] Tahir, M. A., Khan, H. F., & Khan, M. M. (2020, December). Comparative Study of Nature-Inspired Algorithms. In International Working Conference on Transfer and Diffusion of IT (pp. 353-361). Springer, Cham.

[5] Xia, C., Zhang, Y., & Hui, Z. (2021). Test suite reduction via evolutionary clustering. IEEE Access, 9, 28111-28121.

[6] Kiran, A., Butt, W. H., Shaukat, A., Farooq, M. U., Fatima, U., Azam, F., & Anwar, Z. (2020). Multi-objective regression test suite optimization using three variants of adaptive neuro fuzzy inference system. Plos one, 15(12), e0242708..

[7] Z. Anwar, H. Afzal, N. Bibi, H. Abbas, A. Mohsin, O. Arif, "A hybrid-adaptive neuro-fuzzy inference system for multi-objective regression test suites optimization", Neural Computing &Applications . Nov2019, Vol. 31 Issue 11, p7287-7301.

[8] Yoo, S., & Harman, M. (2010). Using hybrid algorithm for pareto efficient multi-objective test suite minimisation. Journal of Systems and Software, 83(4), 689-701.

[9] Mittal, S., & Sangwan, O. P. (2018). Prioritizing test cases for regression techniques using metaheuristic techniques. Journal of Information and Optimization Sciences, 39(1), 39-51.

[10] Kumar, G., Chahar, V., & Bhatia, P. K. Software Test Case Reduction and Prioritization.

[11] Bharathi, M., & Sangeetha, V. (2020). Fault-type coverage based ant colony optimization algorithm for attaining smaller test suite. IAES International Journal of Artificial Intelligence, 9(3), 507.

[12] N. Gupta, A. Sharma, M. K. Pachariya, "Multi-objective test suite optimization for detection and localization of software faults" ,Journal of King Saud University - Computer and Information Sciences,2020, ISSN 1319-1578,https://doi.org/10.1016/j.jksuci.2020.01.009.

[13] K.Z. Zamli, F. Din, B.S. Ahmed, M. Bures, "A hybrid Q-learning sine-cosine-based strategy for addressing the combinatorial test suite minimization problem", PLoSONE 13(5), 2018: e0195675. https://doi.org/10.1371/journal. pone.0195675.

[14] Sivaji, U., & Rao, P. S. (2021). Test case minimization for regression testing by analyzing software performance using the novel method. Materials Today: Proceedings.

[15] J. S. Samagh, D. Singh, "Machine Learning Based Hybrid Model for Heart Disease Prediction" Annals of the Romanian Society for Cell Biology, 2021. https://www.annalsofrscb.ro/index.php/journal/article/view/275.

[16] N. Ahuja, P. K. Bhatia, "Different computational perspective of test suite minimization in software testing" , International Journal Of Scientific & Technology Research, Volume 9, Issue 02, February 2020, ISSN 2277-8616.

[17] Zou, F., Chen, D., & Xu, Q. (2019). A survey of teaching–learning-based optimization. Neurocomputing, 335, 366-383.

[18] Malhotra, R., & Khari, M. (2014). Test suite optimization using mutated artificial bee colony. In Proc. of Int. Conf. on Advances in Communication, Network, and Computing, CNC, Elsevier (pp. 45-54)..

[19] Gupta, P. K. (2021). K-Step Crossover Method based on Genetic Algorithm for Test Suite Prioritization in Regression Testing. J. Univers. Comput. Sci., 27(2), 170-189.

[20] Bharathi, M., & Sangeetha, V. (2018, June). Weighted Rank Ant Colony Metaheuristics Optimization Based Test Suite Reduction in Combinatorial Testing for Improving Software Quality. In 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS) (pp. 525-534). IEEE.

[21] Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization (Vol. 200, pp. 1-10). Technical report-tr06, Erciyes university, engineering faculty, computer engineering department.

[22] Rao, R. V., Savsani, V. J., & Vakharia, D. P. (2011). Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Computer-aided design, 43(3), 303-315.

[23] Ding, L., Wu, H., & Yao, Y. (2015). Chaotic artificial bee colony algorithm for system identification of a small-scale unmanned helicopter. International Journal of Aerospace Engineering, 2015.

[24] Kuang, F., Jin, Z., Xu, W., & Zhang, S. (2014, July). A novel chaotic artificial bee colony algorithm based on tent map. In 2014 IEEE congress on evolutionary computation (CEC) (pp. 235-241). IEEE.

[25] Mirjalili, S., & Gandomi, A. H. (2017). Chaotic gravitational constants for the gravitational search algorithm. Applied soft computing, 53, 407-419. https://doi.org/10.1016/j.asoc.2017.01.008.

[26] Farah, A., Guesmi, T., Abdallah, H. H., & Ouali, A. (2016). A novel chaotic teaching–learning-based optimization algorithm for multi-machine power system stabilizers design problem. International Journal of Electrical Power & Energy Systems, 77, 197-209.

[27] Ahuja, N., & Bhatia, P. K. (2022). Test Suite Minimization Based upon CMIMX and ABC. In Proceedings of Data Analytics and Management (pp. 347-356). Springer, Singapore.Singh, Y.

[28] In Software Testing; Cambridge University Press: Cambridge, (2013).

[29] Wang, R. (2016). An improved nondominated sorting genetic algorithm for multiobjective problem. Mathematical Problems in Engineering, 2016.

[30] Mongiovì, M., Fornaia, A., & Tramontana, E. (2020). REDUNET: reducing test suites by integrating set cover and network-based optimization. Applied Network Science, 5(1), 1-21.

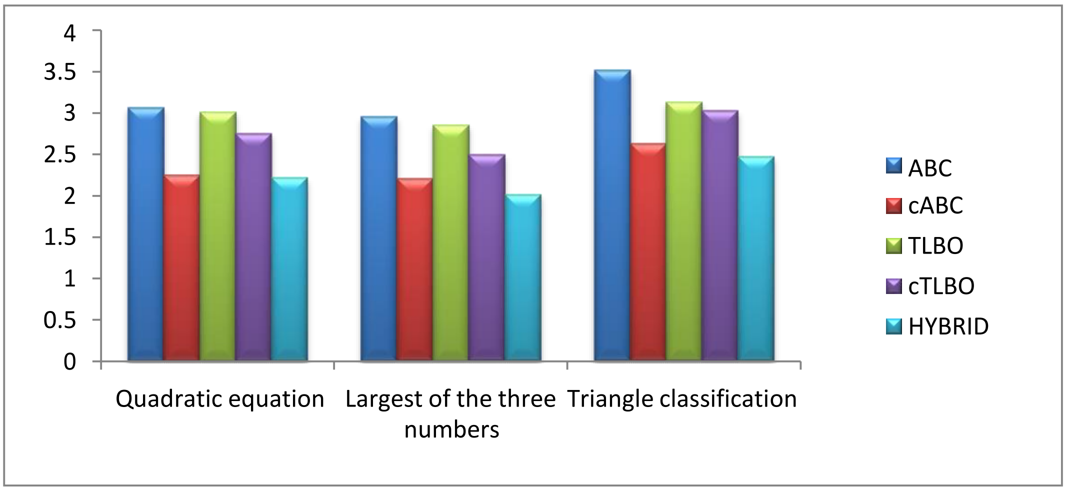


Fig. 10. Total execution time

TABLE I. PARAMETERS CONSIDERED FOR VARIOUS ALGORITHMS

Algorithms	**Parameter values**
ABC	Population size =50 Max iteration =20 L= round(0.6*nPop) Employee bee= onlooker bee Decision variable= (2-10) Roulette wheel selection
CABC	Population size =50 Max iteration =20 L= round(0.6*nPop) Employee bee= onlooker bee Decision variable= (2-10) Roulette wheel selection Logistic chaotic map
TLBO	Population size =50 Max iteration =20 Decision variable= (2-8)
CTLBO	Population size =50 Max iteration =20 Decision variable= (2-8) Logistic chaotic map
HYBRID	Population size =50 Max iteration =20 Decision variable= (2-7) L= round(0.6*nPop) Employee bee= onlooker bee Tournament Selection Logistic chaotic map

TABLE II. COMPARISON OF METRIC FOR ALL ALGORITHMS

Program	Algorithm	Convergence iteration	Size of test suite	APSC
Triangle Classifier	*ABC*	*15*	*6*	*94.44*
	CABC	*13*		*94.83*
	TLBO	*9*		*95.83*
	CTLBO	*6*		*96.77*
	HYBRID	*2*		*97.22*
	ABC	*8*	*5*	*83.43*
	CABC	*7*		*84.48*
	TLBO	*5*		*94.44*
	CTLBO	*3*		*95.23*
	HYBRID	*1*		*96.66*
	ABC	*5*	*4*	*79.54*
	CABC	*4*		*82.38*
	TLBO	*2*		*93.65*
	CTLBO	*2*		*93.80*
	HYBRID	*1*		*95.24*
	ABC	*4*	*3*	*79.54*
	CABC	*3*		*82.38*
	TLBO	*3*		*93.65*
	CTLBO	*2*		*93.80*
	HYBRID	*1*		*94.84*
	ABC	*2*	*2*	*79.52*
	CABC	*2*		*82.38*
	TLBO	*2*		*92.33*
	CTLBO	*1*		*92.60*
	HYBRID	*1*		*93.65*

TABLE III. VALUE OF CONVERGENCE ITERATION AND APRS FOR ALL PROGRAMS

Algorithm/Program	**Quadratic equation**			**Largest of the three numbers**			**Triangle classification**		
	Convergence iteration	**Execution Time**	**APRS**	**Convergence iteration**	**Execution Time**	**APRS**	**Convergence iteration**	**Execution Time**	**APRS**
ABC	4	2.327	44	3	2.007	46	10	2.432	43
cABC	4	1.95	46	3	1.931	46	10	1.984	44
TLBO	4	2.273	47	3	2.299	57	8	2.36	48
cTLBO	3	2.261	49	2	2.095	58	7	2.578	46
HYBRID	3	1.71	52	2	1.309	62	7	1.932	50

TABLE IV. TOTAL EXECUTION TIME

Program/ Algorithm	**ABC**	**cABC**	**TLBO**	**cTLBO**	**HYBRID**
Quadratic equation	3.064	2.249	3.012	2.755	2.215
Largest of the three numbers	2.955	2.204	2.855	2.493	2.012
Triangle classification	3.521	2.632	3.132	3.0315	2.474

TABLE V. QUENATIONARIES FOR STATISTICAL RESEARCH

RQ1	H0: There is no significant difference in the statement coverage of algorithms.
	Ha: There is significant difference in the statement coverage of algorithms.
RQ2	H0: There is no significant difference in the convergence iteration of algorithms.
	Ha: There is significant difference in the convergence iteration of algorithms.

TABLE VI. STATEMENT COVERAGE SIGNIFICANCE

t-Test: Paired Two Sample for Means			
Algorithm(I)	**Algorithm(J)**	**Mean Difference of coverage (I-J)**	**Significance difference of coverage**
HYBRID	ABC	6.417109104*	0.000361
	cABC	5.968324068*	0.00056

	TLBO	2.297676079	0.055179739
	cTLBO	-0.182817985	0.860123049
cTLBO	ABC	6.133311806*	0.000475253
	cABC	5.516729*	0.000891
	TLBO	2.116149453	0.072130462
	HYBRID	0.182817985	0.860123049
TLBO	ABC	5.658598224*	0.000767589
	cABC	5.184902387*	0.000767589
	cTLBO	-2.116149453	0.072130462
	HYBRID	-2.297676079	0.055179739
cABC	ABC	-0.126315689	0.903033665
	TLBO	-5.184902387*	0.000767589
	cTLBO	-5.516729*	0.000891
	HYBRID	-5.968324068*	0.00056
ABC	cABC	0.126315689	0.903033665
	TLBO	-5.658598224*	0.000767589
	cTLBO	-6.133311806*	0.000475253
	HYBRID	-6.417109104*	0.000361

*Represents significance difference between algorithms

TABLE VII STANDARD DEVIATION OF STATTEMENT COVERAGE

	ABC	**Cabc**	**TLBO**	**cTLBO**	**HYBRID**
Std. Dev.	0.053759195	0.056752738	0.011859	0.017581	0.012747974

TABLE VIII. CONVERGENCE ITERATION SIGNIFICANCE

t-Test: Paired Two Sample for Means			
Algorithm(I)	**Algorithm(J)**	**Mean Difference of convergence iteration (I-J)**	**Significance difference**
HYBRID	ABC	-2.681914	0.055119
	cABC	-2.547704	0.06346
	TLBO	-2.631174	0.058115

	cTLBO	-3.086975*	0.036682
cTLBO	ABC	-2.484138	0.067909
	cABC	-2.256304	0.087036
	TLBO	-2.057983	0.108701
	HYBRID	3.086975*	0.036682
TLBO	ABC	-2.525343	0.064986
	cABC	-2.13800	0.099300
	cTLBO	2.057983	0.108701
	HYBRID	2.631174	0.058115
cABC	ABC	-3.162278*	0.034109
	TLBO	2.13800	0.099300
	cTLBO	2.256304	0.087036
	HYBRID	2.547704	0.06346
ABC	cABC	3.162278*	0.034109
	TLBO	2.525343	0.064986
	cTLBO	2.484138	0.067909
	HYBRID	2.681914	0.055119

* Represents significance difference between algorithms

TABLE IX. STANDARD DEVIATION OF CONVERGENCE ITERATION

	ABC	**cABC**	**TLBO**	**cTLBO**	**HYBRID**
Std. Dev.	5.069517	4.438468204	2.949576	1.732051	0.447214