An Improved Artificial Bee Colony Optimization Algorithm for Test Suite Minimization

Neeru Ahuja, Pradeep Kumar Bhatia
Dept. of Computer Science and Engineering, Guru Jambheshwar University of Science and Technology,
Hisar, Haryana, India-125001

***Abstract*—Software testing is essential process for maintaining the quality of software. Due to changes in customer demands or industry, software needs to be updated regularly. Therefore software becomes more complex and test suite size also increases exponentially. As a result, testing incurs a large overhead in terms of time, resources, and costs associated with testing. Additionally, handling and operating huge test suites can be cumbersome and inefficient, often resulting in duplication of effort and redundant test coverage. Test suite minimization strategy can help in resolving this issue. Test suite reduction is an efficient method for increasing the overall efficacy of a test suite and removing obsolete test cases. The paper demonstrates an improved artificial bee colony optimization algorithm for test suite minimization. The exploitation behavior of algorithm is improved by amalgamating the teaching learning based optimization technique. Second, the learner performance factor is used to explore the more solutions. The aim of the algorithm is to remove the redundant test cases, while still ensuring effectiveness of fault detection capability. The algorithm compared against three established methods (GA, ABC, and TLBO) using a benchmark dataset. The experiment results show that proposed algorithm reduction rate more than 50% with negligible loss in fault detection capability. The results obtained through empirical analysis show that the suggested algorithm has surpassed the other algorithms in performance.**

Keywords—Test suite; test suite minimization; TLBO; ABC; nature inspired algorithm

I. INTRODUCTION

Software engineering deals with the design, analysis, implementation, maintenance and testing of software. Once software is evolved, its defects and shortcomings are analyzed with the help of software testing [4]. Out of various testing methods, the regression testing is very important as it involves the modification or insertion of a code into the already working code [8]. More is the size of test suite more will be testing time for each run. This testing time even varies in weeks also. It becomes difficult for a developer to get early feedback of the software developed. Therefore, the developer cannot fix the problems arising into the software timely [1]. So, no more changes could be done in the software and root cause of test suite failure could not be fixed. Software engineers must utilize their time and resources effectively to prevent these issues [2]. A lot of researchers have carried out their work in this particular area [3]. Test case selection (TCS), test case prioritization (TCP), and test case minimization (TCM) are three techniques used to handle complex problems associated with testing.

Test case selection (TCS) [21] is used to choose the test cases for the modified portion of the software. The test cases chosen from the test suite may depend on how well the selection process works. According to a specific criterion, test cases are prioritized, and the highest priority test cases are run first to achieve particular objectives. During test suite reduction, redundant test cases are removed from test suite [31]. Test case minimization, which gets rid of unnecessary ones, speeds up regression testing. The utilization of TCM depends on how it will help minimize time and cost for a particular software [6]. Based on criteria, minimization can reduce the size of test suite. There exist different criteria like path coverage, statement coverage, fault coverage, etc. It's important to always keep in mind that minimization of any kind carries risk and might not offer complete coverage. It might also omit some fault revealing test cases. The size of the test suite, effectiveness, fault detection cost, and execution time are the important parameters of software that should be measured at the time of reduction. The effectiveness of software is decided by measuring these parameters. The comparison of a reduced suite with a corresponding unreduced suite using criteria other than suite size is also an important issue. As the testing basically involves the detection of faults in the software, it can be said that fault detection capability is one of the measures for test suite quality. An important shortcoming of the reduction process is the complete elimination of test cases from a suite, which may lead to a decrease in the fault detection capability and effectiveness of the remaining suite. Thus, a proper exchange between fault detection effectiveness and execution time should be taken into account before the implementation of TSR [7]. Therefore, the problem of test suite minimization could be considered an NP-hard problem. Optimization techniques efficiently solve these problems. Therefore, it can also be said that effectiveness and efficiency can be increased in regression testing through optimization techniques. Optimization helps us extract the best fit solution for the problems [36]. Soft computing uses artificial intelligence and natural selection to solve very complex problems that analytical (hard computing) formulations cannot solve. By incorporating soft computing techniques in optimization, we can further improve the accuracy and speed of regression testing. Soft computing can handle uncertainty and imprecision in data, making it a valuable tool for optimizing complex systems. Many researchers have investigated different techniques to minimize test suite while maintaining the coverage as maximum as possible [34,37]. CMIMX technique with a prominent soft computing technique (ABC) was implemented to ensure the reduced set with maximum fault coverage [20]. This paper proposed a new hybrid algorithm for test suite minimization. The proposed algorithm attains the minimize test suite with maximum coverage. The minimize test case achieve it by selecting two objectives. First, the algorithm aims to identify the minimized test suite that achieves maximum statement coverage. Second, it should also satisfy the fault detection capability of selected test cases. To accomplish our objective we have selected Artificial Bee Colony and Teaching Learning based optimization. The key contributions of our presented work are summarized as follows:

- Integration of TLBO operator into ABC algorithm for enhanced search performance.
- Introduction of learner's performance concept to accommodate varying abilities.
- Minimization of test suite while maintaining optimal test case coverage.
- Analysis of fault coverage loss resulting from test suite minimization.

The remaining of this paper is organized as follows:

Section II describes the literature study related to minimization techniques. The technical background of the techniques explained in Section III. Section IV illustrates proposed methodology. Experimental details are present in Section V. Section VI describes conclusion and future work.

II. State-of-the-Art

In this section, we present a comprehensive review of the existing literature on test suite minimization techniques for addressing NP-hard problems. Researchers have extensively explored the application of soft computing techniques to tackle these challenging problems [35]. We summarize the key findings and contributions of various studies in Table I, highlighting the author names, techniques used, coverage criteria, and defined results.

ABC and TLBO algorithms have emerged and popular algorithm of literature and demonstrated the promising results across the different domains like long-term economic dispatch problem in hydropower systems [11]. Rao et al. initially proposed simple TLBO, which lack an inertia weight value [15]. I-TLBO, an enhanced version of TLBO was developed with the concept of adaptive teaching strategy and self-learning [13]. It was implemented to strengthen the exploration and exploitation capabilities. Zhang et al. [17] integrated TLBO algorithm into onlooker bee phase. Additionally, a novel searching approach for the bee phase was used to enhance population variety and quicken convergence. One such technique is FATLBO in which F and A stands for fuzzy and adaptive. This technique used fuzzy logic to adaptively adjust the parameters of TLBO algorithm [18]. LebTLBO was proposed by Chen et al. [14] to increase the performance of TLBO by incorporating the learning enthusiasm mechanism. It is clear from the literature review that previous research has mostly focused on test suite minimization using various coverage criteria. However, there are a number of gaps in the available literature, including a lack of analysis on fault coverage and limited use of the ABC algorithm and TLBO algorithm in the context of minimization principles. Therefore, we presented ABC and TLBO technique with addition of learning factor to enhance the capability to search more solution space. We suggest a unique method for test suite minimization termed Learner performance-based ABC and TLBO (LpABTLO) to fill these gaps. This method uses statement coverage as the coverage requirement and subsequently analyses fault detection loss.

Our suggested LpABTLO technique aims to fill these gaps in order to develop test suite minimization techniques and deliver more thorough insights into fault coverage analysis.

III. Technical Background

This section gives a technical overview of the TLBO algorithm and ABC, two key optimization strategies. These approaches gained a lot of focus in the different fields to solve optimization problems.

A. *Artificial Bee Colony Algorithm (ABC) [38]*

The novel swarm-based stochastic optimization method known as ABC, developed by Karaboga, mimics the foraging behavior of honey bees [33]. Honey bees are organized into three groups in ABC: workers, bystanders, and scouts [9]. Employee bees compose the initial half of the colony, and observer bees made up the remaining half [10]. During the employee bee phase, bees search for a food source and collect data on its quality. As a result, only one bee is assigned to each food source [11]. The food source with the higher fitness value is more likely to be picked by onlookers than the one with the lower fitness value. Each onlooker bee then searches for a nearby food source close to the selected one and reserves the best among them. If a food source remains unchanged for a specific period of time, the associated employed bee turns scout and explores a new food source randomly [12]. Fig. 1 explains the working of algorithm.

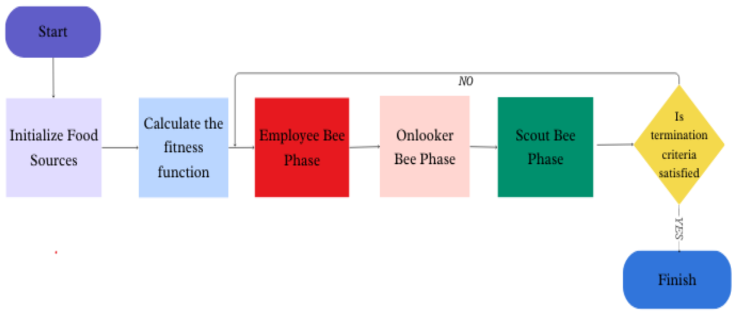

Fig. 1. Flowchart of ABC algorithm

TABLE I. DEFINES THE LITERATURE STUDIES OF VARIOUS RESEARCHERS

Author Name	Technique	Coverage Criterion	Defined Result
Khan et al. [27]	GA and mutation analysis	Software testing efficiency	Optimized software testing efficiency through GA and mutation analysis
Mala et al.[28]	ABC algorithm	Path coverage	ABC: 99% coverage in 50 cycles, GA: 90% coverage in 300 generations
Khari et al. [22]	ABC and cuckoo search methods	Test case size and path coverage	ABC and cuckoo search minimize size of test case with path coverage
Zeeshan Anwar et al. [5]	Hybrid-adaptive neuro-fuzzy inference system with GA and PSO	Regression test suites optimization	Optimized regression test suites using hybrid technique
Yoo and Harman [23]	Pareto optimal solution	Test suite reduction	Provided Pareto optimal solutions for test suite reduction
Sivaji et al. [25]	AB-CNS algorithm	Recall value, accuracy, execution time	Reduced test cases using AB-CNS algorithm with improved effectiveness
Bala et al. [24]	Hybridized technique using harmony search and PSO	Non-functional properties testing	Achieved 100% coverage in less execution time compared to GA, HS, PSO
Coviello et al. [26]	GASSER technique based on NSGA-II	Statement coverage, fault coverage	Reduced test suite size by maximizing statement coverage, compared to traditional approaches
Suri et al. [32]	Hybris approach using GA and ABC	Fault coverage	Cover 100% fault with redcution in test suite size

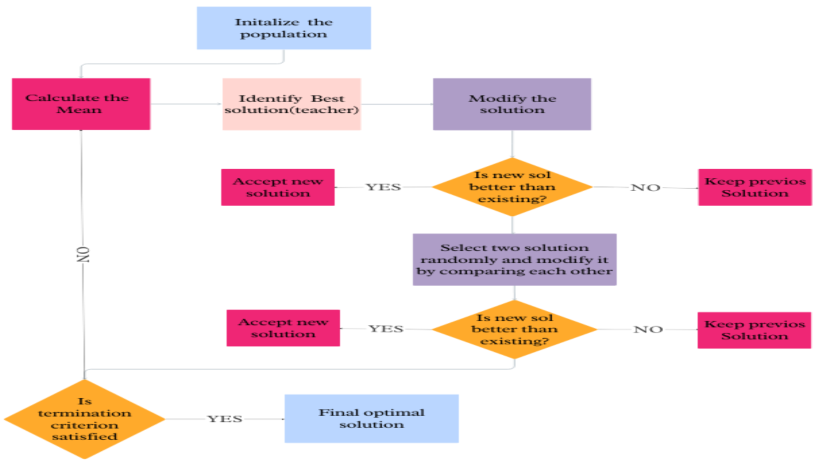
Initalize the population
Calculate the Mean
Identify Best solution(teacher)
Modify the solution
Accept new solution
YES
Is new sol better than existing?
NO
Keep previos Solution
Select two solution randomly and modify it by comparing each other
Accept new solution
YES
Is new sol better than existing?
NO
Keep previos Solution
NO
Is termination criterion satisfied
YES
Final optimal solution

Fig. 2. Flowchart of TLBO.

B. Teaching-Learning-based Optimization (TLBO) Algorithm [15]

Rao et al. developed the TLBO algorithm that is a population-based optimization algorithm used to handle mechanical design optimization issues [15]. However, when solving complex optimization problems, the TLBO algorithm frequently encounters the issue of getting trapped in local optima [14]. Although it has consistently outperformed other evolutionary algorithms in theoretical and practical tests, the impact of control parameters on TLBO's performance cannot be overlooked [16]. The TLBO algorithm is composed of two distinct phases: the Teacher Phase and the Learner Phase. In the Teacher Phase, the teacher aims to enhance the students' knowledge. However, the progress of the students is not solely determined by the teacher's expertise, but also influenced by the mean level of the entire class [15]. Fig. 2 illustrates the working of TLBO.

IV. PROPOSED METHODOLOGY

This section presents a proposed model for test suite minimization problem as shown in Fig. 3. The proposed work is composed of three modules. The pseudo code of proposed work is described in algorithm 1.

We have developed model for test suite minimization problem as illustrated in Fig. 3. There are three components comprise the proposed work. Algorithm 1 defines the pseudo code of proposed work.

Algorithm 1. Hybrid Artificial Bee Colony

```
1.   Begin
2.   Define max_gen , max_population
3.   Initialize Food source(population)
4.   Evaluate fitness function
5.   For i=1: max_gen
6.       For i=1: max_population
7.           Teacher based employee bee phase as shown in
             Algorithm 2
8.           Learner based onlooker bee phase as shown in
             Algorithm 3
9.           Update teacher and mean
10.          Memorize the best solution
11.          Scout Phase
12.      End for
13.  End for
14.  Find the fault detection capability as shown in
     Algorithm 4
15.  End
```

A. Dataset Extraction Module

We have taken programs from different sources. As we know minimization techniques are based on adequacy criteria. Statement coverage and fault coverage matrix are extracted from source code.

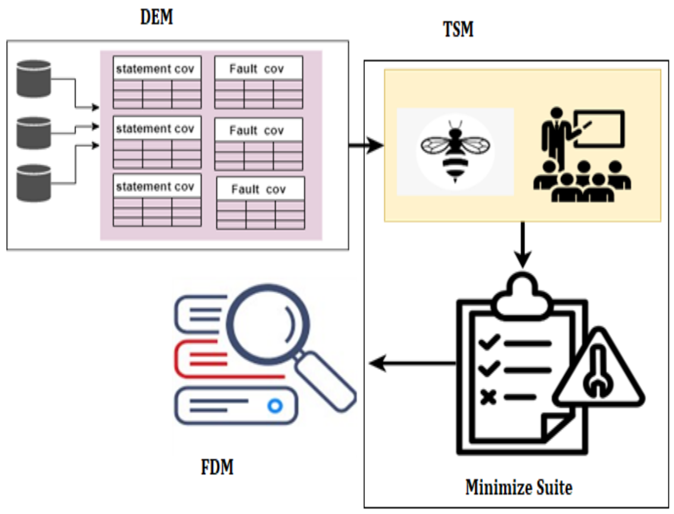

Fig. 3. Framework of proposed algorithm.

B. Test Suite Minimization Module

Second module describes the working of hybrid technique. There are two important factors related to heuristic optimization methods. The first is called "exploration," and it entails a comprehensive search of the space while providing a variety of potential solutions at each stage. This factor demonstrates a method's ability to perform well in a global search. Exploitation, or the quality of solutions attained during each cycle, is the other consideration. This metric demonstrates a technique's capability in performing a local search and locating the optimal response close to a solution. These two considerations are in opposition to one another and deserve to be pointed out. Thus, if you prioritize global search over local search (i.e. exploitation), you may end up with a poor final best answer, and vice versa [30]. In the proposed technique two popular soft computing techniques ABC and TLBO are amalgamated with each other. It is developed by incorporating the exploitation capability of TLBO with changes. Both the techniques are complementary to each other.

In the hybrid model we incorporated two significant changes to algorithms. First, instead of employee and onlooker bee phase, teacher and learner phases are embedded into model. ABC algorithm uses the same perturbation for Employee bee and onlooker bee. Therefore, ABC algorithm stagnates from the capability of exploitation. To solve this issue we have embedded the teacher and learner phase of TLBO algorithm into employee and onlooker bee phase with modification in operator.

Second, from the literature we have observed that TLBO and its variants use same approach to update the knowledge of learners in the both phases of TLBO. Every student is unique, and their level of zeal for learning varies [14]. Some students are more interested to gain knowledge on other hand others shows less interest in studies and neglect the knowledge gained from others. As the result of this thought we have divided the learners according to learner performance.

The main aim of this approach is to identify a smaller group of test cases that can cover the maximum statements while still providing the same level of coverage like the original test suite. Additionally, the proposed approach also checks the fault coverage for selected test cases.

1) Population initialization: The algorithm starts with random initialization of food source (test suite) using permutation encoding. In permutation encoding, test cases will not be repeated, and only unique test case can be part of a test suite. Like, TS1= [8, 99, 21, 43, 23, 56]. Set dimension, trail counter and limit.

2) APSC: It is essential to assess the quality of all the population's food sources to evaluate the effectiveness of a potential solution provided by that food supply. The objective function allows selecting the best individual that leads to a good solution and validates the process's deviation from its optimization target. The objective of the problem is to reduce the number of test cases while still providing maximum statement coverage.

The average percentage of statement coverage (APSC) formula is defined in (1).

$$\text{Maximize } APSC = 1 - \frac{\sum_{i=1}^{n} \mathrm{scov_i}}{n*m} + \frac{1}{2*n} \tag{1}$$

Here, n and m represents the total number of test cases in test suite and statements.

3) Fitness function: Fitness functions are employed in simulations to steer them towards the best possible design solutions.

Fitness function is calculated using (2)

$$Fit(f(x) = \begin{cases} \frac{1}{1+f(x)} & f(x) \geq 0 \\ 1 + abs(f(x) & f(x) < 0 \end{cases} \tag{2}$$

Here, f(x) represents the objective function, i.e., calculated in eq. 1.

4) Teacher-based employee bee phase: In the basic TLBO bees positions are updated using eq.3.

$$x_{i,new} = x_{i,old} + rand\left(x_{teacher} - t_f * x_{mean}\right) \tag{3}$$

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

$$x_{mean} = \frac{\sum_{i=1}^{np} x_i}{\mathrm{np}} \tag{4}$$

Here np is number of bees.

But it suffers from poor diversity issue. To overcome this issue, DE (differential equation) mutation [19] is utilized. Though teacher strategy is hybridized with DE (differential equation) mutation is done to update learner's position by eq. 5. The steps of teacher based employee bee phase are shown in algorithm 2.

$$x_{i,new} = \begin{cases} x_{i,old} + rand * \left(x_{teacher} - t_f * x_{mean}\right) & if\ rand < .5 \\ x_{r1} + f\left(x_{r2} - x_{r3}\right) & else \end{cases} \quad (5)$$

Here rand denotes uniformly distributed random number within [0,1]. r1, r2, and r3 are random solutions from {1…np} and r1, r2, r3 cannot be equal. *f* represents the scaling factor between 0 and 2.

Algorithm 2 Teacher based Employee Bee Phase

```
Step1 Select the best bee as $X_{best.}$
Step2 Calculate mean ($X_{mean}$) of all learner bees.
Step3 Generate the new bee according to DE mutation eq. 5
Step4 Calculate fitness function for new bees.
Step5 Accept new bee if it is better than old ones.
```

5) Tournament selection: Roulette wheel selection is based on a proportionate selection technique in which the best wheel can be chosen or not. We adopted tournaments to solve this problem. A tournament is a match involving several known-sized competitors who are randomly selected to compete. The value of each individual's fitness is used to select the winner. The size of the tournament is determined by the number of participants.

6) Learner-based onlooker bee phase: As we have already discussed, in basic TLBO no variation exists while gaining knowledge among learners. Therefore, in the learner phase, we have used a factor called learner performance (Lp). Learner performance formulates variation in the best student's and other student's knowledge. If students have knowledge greater than the learner performance value, they will improve their knowledge by interacting with the best one as in equation 6. Otherwise, the criteria for gaining knowledge gain will be the same as basic TLBO as specified in equations 7 and 8. We have tried the value of Lp from [0.7, 1] by trial and error method, out of which 0.8 is the best-suited value. Those students with a learning performance of 0.8 or more will learn from the best learner or teacher.

$$x_{nbest} = x_{best} + rand\left(x_{best} - x_j\right) \quad (6)$$

$$x_{new,i} = x_{i,old} + rand(x_{best} - x_i) + rand\left(x_i - x_j\right) \quad (7)$$

$$x_{new,i} = x_{i,old} + rand(x_{best} - x_i) + rand\left(x_j - x_i\right) \quad (8)$$

The steps to Learner-based onlooker bee phase are described in algorithm 3.

Algorithm 3 Learner based Onlooker Bee Phase

```
1.  Begin
2.  Initialize   food source(population)
3.  for each learner $x_i$
4.      If $f(x_i)<f(x_j)$
5.          If  $Lp>f(x_j)$
6.              Update the new bee using equation  6
7.          else
8.              Update the new bee using equation 7
9.          end If
10.     else
11.             Update the new bee using equation 8
12.     end If
13.     Calculate fitness function for new bees.
14.     Accept new bee if it is better than old ones.
15. end for
16. End
```

7) Scout phase: When an employee bee's food source reaches a predetermined limit, it is reclassified as a scout bee. Similar to the worker bee phase, the new random scout bee is formed randomly. If this happens, the previous methods should be updated with the new bee solution.

C. Fault Detection Capability

There are different criteria for minimizing test suites. In the proposed technique, we have selected statement coverage as the criterion. However, the effectiveness of the technique can be checked by detecting losses for other criteria. Since one criterion can affect others, we have evaluated fault-revealing capability of the reduced test suite, as shown in Algorithm 4.

Algorithm4. Fault Detection Capability Algorithm

```
1.  Begin
2.  Define Reduced test suite (RTS), Total Faults(TF), Fault
    matrix (FM)
3.  Initialize  variable F=0
4.  For i=1: size (RTS)
5.      a= RTS[i]
6.      For  j=1: size (FM)
7.          If  (FM[a][j]==1)
8.              F=F+1
9.          End if
10.     End for
11.     Calculate detected faults  by  ((F/TF)*100)
12. End for
13. End
```

V. Experimental Evaluation and Results

In this subsection, we use benchmark programs to evaluate our proposed algorithm. Using Matlab2016a, we performed the experiments on a personal computer with a 2.00 GHz/core (i3) CPU and 4GB of memory. In order to validate the effectiveness of the proposed algorithm, a comparative analysis was conducted between its results and those obtained from the ABC, TLBO, and GA algorithms. Each algorithm was executed 20 times. The evaluation metrics used, the data sets explored, and the outcomes of the experiments are all detailed here.

A. Evaluation Metrics

To rigorously evaluate and compare the effectiveness of our proposed approach with state-of-the-art methods, we leveraged well-established structural coverage measures. These measures have been widely adopted and extensively studied as reliable indicators of a test suite's efficacy. By employing these measures, we aimed to provide a robust performance evaluation of our algorithm. Table II illustrates the evaluation metrics.

B. Dataset

For the validation of the algorithm, we used datasets from different sources [30, 29]. These programs vary in size from small to large, helping to determine whether performance variations are due to test suite size scalability.

C. Computational Analysis and Discussion on Results

In this subsection, we calculated the outcomes of the proposed algorithm and evaluated it in comparison to the other algorithms. Our proposed approach has undergone a comprehensive evaluation, focusing on two key factors: efficiency and effectiveness. Through rigorous testing and analysis, we have thoroughly assessed the performance of our approach in terms of these crucial aspects. The effectiveness of the program is measured by the APRS, APSL, and APFL, while efficiency is measured by the execution cost. Therefore, APCR is used as efficiency metric.

1) Efficiency: The efficiency of the algorithm is assessed by measuring the execution cost required to find the reduced test suite. As we know, the larger the test suite, the higher the computational cost. Reduction of the test cases can help to decrease the total computational cost. Table III illustrates the cost reduction percentage achieved by reducing the test suite size compared to the original cost. As determined from Fig. 4, TLBO performed best for large-sized programs, followed by the proposed algorithm, ABC-TLBO, GA, and ABC. There is a minor difference in the result of the proposed and TLBO algorithms. For small-sized programs, the proposed algorithm performed best. The difference between the proposed and ABC-TLBO algorithms was less for small-sized programs. As evident from Table III, the execution cost difference between algorithms is merely a fraction of a second. Therefore, while weighing the benefits of the algorithm, efficiency plays a relatively small role in algorithm selection. TLBO performed best, followed by the proposed algorithm, ABC-TLBO, GA, and ABC.

2) Effectiveness: Removing unnecessary test cases may affect the fault detection capability. Table IV shows the experimental outcomes of LpABTLO and other algorithms on the minimization problem. It is clear from Fig. 5 that LpABTLO reduces more test cases than other algorithms. LpABTLO reduces 51% of test cases without compromising the fault detection rate. Moreover, as the size of the program and the number of test cases increase, the reduction rate also increases for all the techniques. However, LpABTLO reduction is higher than other techniques. It provides the optimal result. The comparative analysis for statement coverage is presented through Fig. 6. For small-sized programs, the statement coverage of LpABTLO and ABC-TLBO is similar, but ABC lags behind GA and TLBO. ABC-TLBO is the second best after LpABTLO in coverage without any fault loss in small-sized programs. For large-sized programs, LpABTLO and ABC-TLBO attain 96.91% and 96.43%, respectively, while GA, ABC, and TLBO get 94.32%, 92.56%, and 94.51%, respectively. By utilizing statement coverage as a metric for testing, we have also examined the loss in fault detection capabilities resulting from reduced test cases. The percentage of fault-detection loss is depicted in Fig. 7. The figure shows that LpABTLO has the least fault coverage loss compared to other algorithms. GA has the maximum fault coverage loss after ABC. The technique that can reduce the size of the test case, cover maximum statements, and without loss in fault detection capability is the best choice for selection. LpABTLO covers all the conditions. Hence, LpABTLO is superior to other techniques.

TABLE II. Represents the Evaluation Metrics

Metric	Formula	Definition
Average percentage of reduced size	APRS = ((OTS – Ored) / OTS) * 100	Percentage of reduced test suite's size compared to the original test suite. Higher value of RSP is better.
Average percentage of fault detection capability loss	APFL = ((F - Fred) / F) * 100	Quantifies the degree of fault coverage loss in the reduced test suite compared to the original test suite. Lower value of APFL is better.
Average percentage of statement coverage loss	APSL = ((S - Sred) / S) * 100	Quantifies the degree of statement coverage loss in the reduced test suite compared to the original test suite. Lower value of APSL is better.
Average percentage of Cost reduction	APCR = ((E - Ered) / E) * 100	Measures the extent of cost reduction achieved during the test suite reduction process.

TABLE III. COMPARISON OF COST REDUCTION OF PROPOSED APPROACH (LPABTLO) WITH OTHER ALGORITHMS

Dataset	Algorithms	APCR
Traingle classification Problem(TCP)	**LpABTLO**	**88.433**
	GA	85.212
	ABC-TLBO	87.876
	ABC	83.650S
	TLBO	84.985
Quardratic equation(QE)	**LpABTLO**	**89.980**
	GA	84.785
	ABC-TLBO	88.675
	ABC	82.490
	TLBO	86.456
Crossword	**LpABTLO**	75.700
	GA	74.132
	ABC-TLBO	74.235
	ABC	70.214
	TLBO	**76.870**
Freemind	**LpABTLO**	69.320
	GA	67.875
	ABC-TLBO	69.231
	ABC	65.773
	TLBO	**69.750**

TABLE IV. COMPARATIVE STUDY OF ALGORITHMS FOR DIFFERENT METRICS

Program Versions	Algorithms	APSC	APRS	APSL	APFL
Traingle classification Problem(TCP)	**LpABTLO**	**97.56**	**51.275**	**0**	**0**
	GA	95.21	46.584	**0**	1.25
	ABC-TLBO	96.32	50.923	**0**	0
	ABC	94.83	43.552	**0**	1.3
	TLBO	95.83	48.237	**0**	0
Quardratic equation(QE)	**LpABTLO**	**98.86**	**55.869**	**0**	0
	GA	96.7	48.538	**0**	2.31
	ABC-TLBO	97.56	52.512	**0**	0
	ABC	96.102	44.325	**0**	1.42
	TLBO	97.20	47.675	**0**	0
Crossword	**LpABTLO**	**95.53**	**65.762**	**0**	**.546**
	GA	93.76	62.453	.643	1.642
	ABC-TLBO	96.56	66.675	.025	.679
	ABC	92.23	63.218	.854	1.758
	TLBO	94.35	64.77	**0**	.783
Freemind	**LpABTLO**	**96.91**	**68.523**	.046	**1.641**
	GA	94.32	61.768	.875	2.321
	ABC-TLBO	96.43	65.762	.065	1.897
	ABC	92.56	62.605	.947	2.987
	TLBO	94.51	65.543	**0.43**	1.934

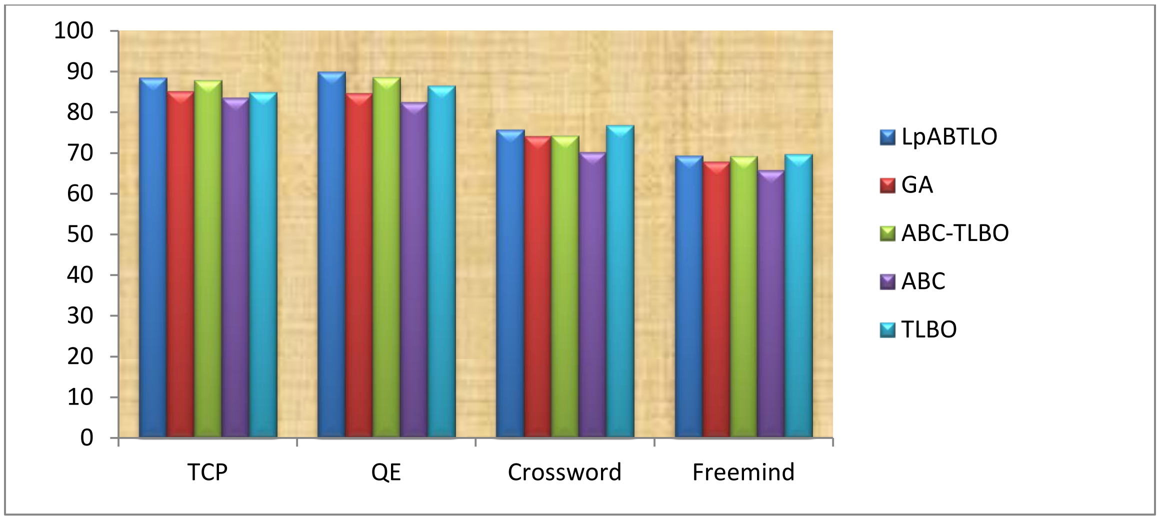

Fig. 4. Program wise comparative analysis of algorithms for cost reduction.

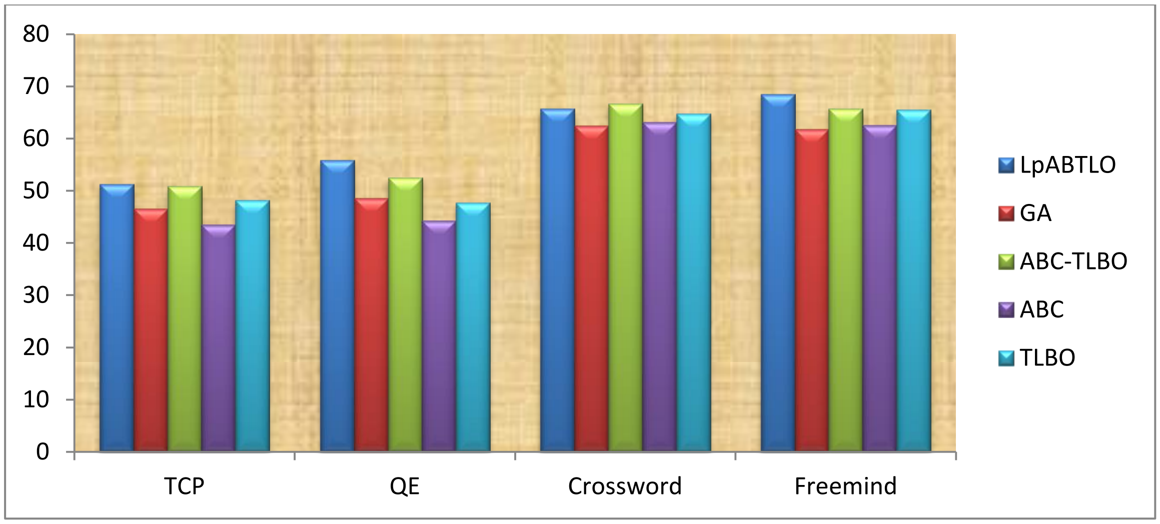

Fig. 5. Program wise comparative analysis of algorithms for size reduction.

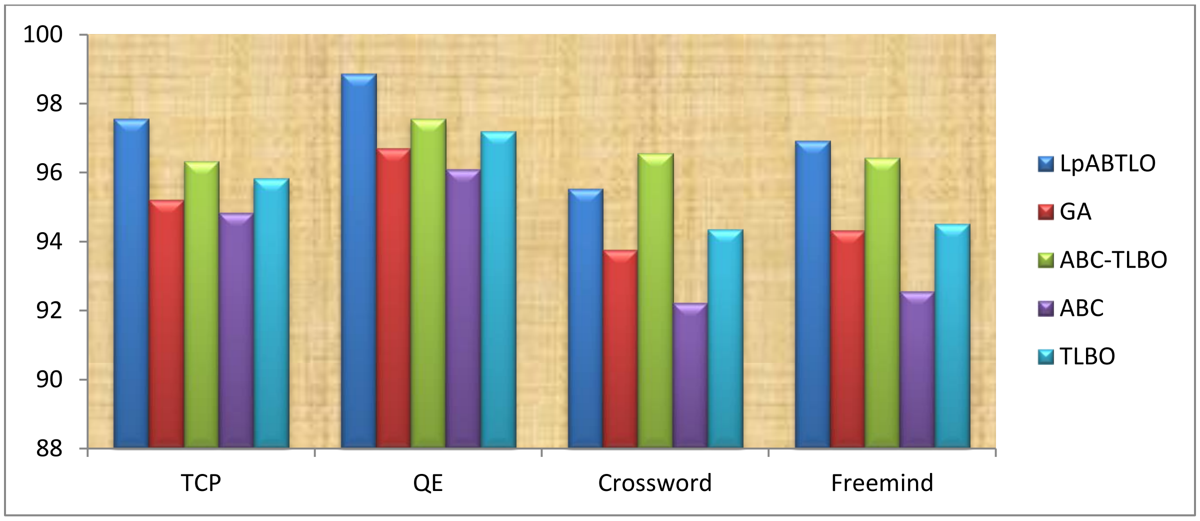

Fig. 6. Program wise comparative analysis of algorithms for statement coverage.

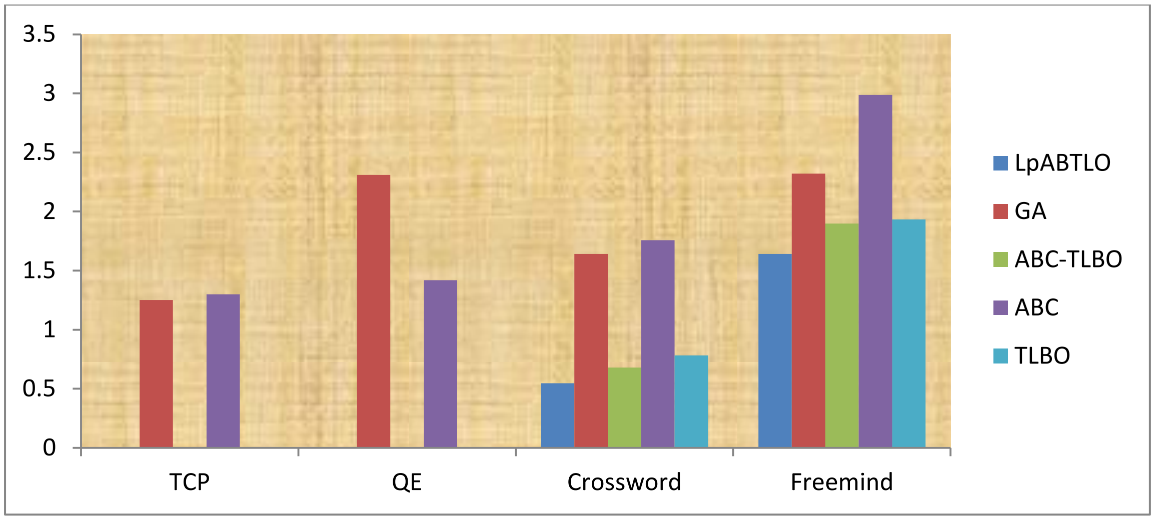

Fig. 7. Analysis of fault detection loss of programs using different algorithms.

VI. Conclusion

Regression testing is considered an NP-hard problem. Optimizing methods can be used to solve these issues by identifying the optimal approach. We have proposed a hybrid algorithm with a combination of ABC and TLBO. As the ABC algorithm uses the same perturbation for the Employee bee and onlooker bee, it stagnates from the capability of exploitation. To solve this issue, we have embedded both phases of the TLBO algorithm into the employee and onlooker bee phase with modification in the operator. The algorithm found the results in two steps. Firstly, it removes redundant test cases to have maximum structural coverage. Further, it checks the fault revealing capability loss due to minimization. We have tested the proposed algorithm against ABC, TLBO, and ABC-TLBO on different-sized programs. It has been determined that the proposed algorithm outperforms than the constituent.

References

[1] Noemmer, R., & Haas, R. (2020, January). An evaluation of test suite minimization techniques. In *International Conference on Software Quality* (pp. 51-66). Springer, Cham.

[2] Manish Asthana, Kapil Dev Gupta and Arvind Kumar Test Suite Optimization Using Lion Search Algorithm Y.-C. Hu et al. (eds.), Ambient Communications and Computer Systems, Advances in Intelligent Systems and Computing 1097, https://doi.org/10.1007/978-981-15-1518-7_7.

[3] Singh, L., Singh, S. N., Dawra, S., & Tuli, R. (2019, March). A new technique for test suite minimization in regression testing. In *Proceedings of 2nd International conference on advanced computing and software engineering (ICACSE)*.

[4] Khan, F. A., Bora, D. J., & Gupta, A. K. (2017). An Efficient Heuristic Based Test Suite Minimization Approach. *Indian Journal of Science and Technology*, *10*(29).

[5] Anwar, Z., Afzal, H., Bibi, N., Abbas, H., Mohsin, A., & Arif, O. (2019). A hybrid-adaptive neuro-fuzzy inference system for multi-objective regression test suites optimization. *Neural Computing and Applications*, *31*(11), 7287-7301.

[6] Shi, A., Gyori, A., Mahmood, S., Zhao, P., & Marinov, D. (2018, July). Evaluating test-suite reduction in real software evolution. In Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis (pp. 84-94).

[7] Parsa, S., & Khalilian, A. (2010). On the optimization approach towards test suite minimization. *International Journal of Software Engineering and its applications*, *4*(1), 15-28.

[8] Yoo S, Harman M (2012) regression testing minimization, selection and prioritization: a survey. Softw Test Verif Reliab 22(2):67–120.

[9] Jagdish Chand Bansal, Harish Sharma, K.V. Arya, Kusum Deep & Millie Pant (2014) Self-adaptive artificial bee colony, Optimization, 63:10, 1513-1532, DOI: 10.1080/02331934.2014.917302.

[10] Cui, L., Li, G., Wang, X., Lin, Q., Chen, J., Lu, N., & Lu, J. (2017). A ranking-based adaptive artificial bee colony algorithm for global numerical optimization. *Information Sciences*, *417*, 169-185.

[11] Liao, X., Zhou, J., Zhang, R., & Zhang, Y. (2012). An adaptive artificial bee colony algorithm for long-term economic dispatch in cascaded hydropower systems. *International Journal of Electrical Power & Energy Systems*, *43*(1), 1340-1345.

[12] Song, X., Zhao, M., Yan, Q., & Xing, S. (2019). A high-efficiency adaptive artificial bee colony algorithm using two strategies for continuous optimization. *Swarm and Evolutionary Computation*, *50*, 100549.

[13] 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.

[14] Chen, X., Xu, B., Yu, K., & Du, W. (2018). Teaching-learning-based optimization with learning enthusiasm mechanism and its application in chemical engineering. *Journal of Applied Mathematics*, *2018*.

[15] 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.

[16] Shukla, A. K., Singh, P., & Vardhan, M. (2020). An adaptive inertia weight teaching-learning-based optimization algorithm and its applications. *Applied Mathematical Modelling*, *77*, 309-326.

[17] Zhang, M., Pan, Y., Zhu, J., & Chen, G. (2018, July). ABC-TLBO: A hybrid algorithm based on artificial bee colony and teaching-learning-based optimization. In *2018 37th Chinese Control Conference (CCC)* (pp. 2410-2417). IEEE.

[18] Din, F., & Zamli, K. Z. (2017, October). Fuzzy adaptive teaching learning-based optimization strategy for pairwise testing. In *2017 7th IEEE International Conference on System Engineering and Technology (ICSET)* (pp. 17-22). IEEE.

[19] Chen, X., Xu, B., Mei, C., Ding, Y., & Li, K. (2018). Teaching–learning–based artificial bee colony for solar photovoltaic parameter estimation. Applied energy, 212, 1578-1588.

[20] 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.

[21] Sampath, S., Bryce, R., & Memon, A. M. (2013). A uniform representation of hybrid criteria for regression testing. *IEEE transactions on software engineering*, *39*(10), 1326-1344.

[22] Khari, M., Kumar, P., Burgos, D., &Crespo, R. G. (2018). Optimized test suites for automated testing using different optimization techniques. *Soft Computing*, *22*(24), 8341-8352.

[23] 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.

[24] Bala, N. M., & bin Safei, S. (2022). A Hybrid Harmony Search and Particle Swarm Optimization Algorithm (HSPSO) for Testing Non-functional Properties in Software System. *Statistics, Optimization & Information Computing*, *10*(3), 968-982.

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

[26] Coviello, C., Romano, S., Scanniello, G., & Antoniol, G. (2020, October). GASSER: Genetic Algorithm for teSt Suite Reduction. In *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)* (pp. 1-6).

[27] Khan, S., Nadeem, A., & Awais, A. (2006). TestFilter: A statement-coverage based test case reduction technique. *2006 IEEE International Multitopic Conference*. https://doi.org/10.1109/inmic.2006.358177.

[28] Mala, D. J., & Mohan, V. (2009). ABC tester-artificial bee colony based software test suite optimization approach. *International Journal of Software Engineering*, *2*(2), 15-43.

[29] https://www.cs.umd.edu/~atif/Benchmarks/UMD2007b.html.

[30] Pandey, A., & Banerjee, S. (2018). Test suite minimization in regression testing using hybrid approach of ACO and GA. *International Journal of Applied Metaheuristic Computing (IJAMC)*, *9*(3), 88-104.

[31] Panichella, A., Di Penta, M., Oliveto, R., & De Lucia, A. (2013). An empirical comparison of test suite reduction techniques for software maintenance. Empirical Software Engineering, 18(4), 609-639.

[32] Zaman, M., Nabi, N., & Shafique, M. (2015). An effective test suite reduction technique for regression testing using genetic algorithm. Journal of Systems and Software, 110, 148-159.

[33] Nabi, N., & Shafique, M. (2015). A hybrid evolutionary approach for test suite reduction using genetic algorithm. Information and Software Technology, 57, 285-297.

[34] Suri, B., Mangal, I., & Srivastava, V. (2011). Regression test suite reduction using an hybrid technique based on BCO and genetic algorithm. Special Issue of International Journal of Computer Science & Informatics (IJCSI), ISSN (PRINT), 2231-5292.

[35] Kulkarni, N. J., Naveen, K. V., Singh, P., & Srivastava, P. R. (2011). Test case optimization using artificial bee colony algorithm. In Advances in Computing and Communications: First International Conference, ACC 2011, Kochi, India, July 22-24, 2011, Proceedings, Part III 1 (pp. 570-579). Springer Berlin Heidelberg.

[36] Khari, M., Kumar, P., Burgos, D., & Crespo, R. G. (2018). Optimized test suites for automated testing using different optimization techniques. Soft Computing, 22, 8341-8352.

[37] Zhang, Y. N., Yang, H., Lin, Z. K., Dai, Q., & Li, Y. F. (2017). A test suite reduction method based on novel quantum ant colony algorithm. In *2017 4th International Conference on Information Science and Control Engineering (ICISCE)* (pp. 825-829). IEEE.