ISSN: 2229-6956 (ONLINE)
DOI: 10.21917/ijsc.2020.0318

A HYBRID BAT APPROACH WITH TABU SEARCH ALGORITHM FOR TEST CASE SELECTION IN OBJECT ORIENTED TESTING

B. Geetha[1] and D. Jeya Mala[2]
[1]*Department of Computer Science and Engineering, Anna University, Chennai, India*
[2]*Department of Master of Computer Applications, Fatima college, India*

Abstract

All research made on the Object-Oriented (OO) paradigms focus on the fundamentals of analysis, programming, and design. The primary problem found in testing the systems which are object-oriented is a methodology of standard testing and this may not be very useful. The test case may execute software using a new set consisting of some input values and will then compare them to the output to check if the test has passed. An optimum test case set is obtained using a process of selection that is viewed to be a problem of optimization. Thus, metaheuristic optimizing or searching is a technique used often for optimizing or searching which is used in automated testing of software. The BAT Algorithm is a metaheuristic that is dependent on the property of echolocation of the miniaturized scale bats. The property further controls the conduct of search of the bats of a small-scale and making them discover prey thus enabling them to identify distinctive types of bugs irrespective of the fact they are found to be dull. The work also proposed a new and hybrid Tabu search algorithm using the BAT for the selection of test case.

Keywords:

Object Oriented (OO) Paradigms, Hybrid Tabu Search, Bat Algorithm, Test Case Selection

1. INTRODUCTION

Software testing is done for guaranteeing both the reliability and quality of software. This Object-oriented (OO) approach is used for developing software in an efficient manner. It also enables one to bring down or eliminate certain typical issues found in procedural software. Also, it can introduce some more new problems which may result in faults that are addressable using techniques of traditional testing. The faults that are state-dependent may occur often in the OO software as opposed to procedural software. Most of the objects will have a new associated state and the member function behaviour will be invoked based on the object which is dependent on the state of the object. These faults are extremely challenging since they may also result in failures at the time the objects have been exercised in certain states [1].

The primary issue in software testing that has been investigated in the paper will refer to the selection of suitable test cases consisting of bugs, errors, and functions. The problem that is serious will be the size of the chosen test cases which may be too big at the time of modifying programs for every version. This may further result in testing time thus increasing errors. The main problem will be the size of the chosen test cases that are too big at the time of modifying every version of the program. This also results in testing time and also increases errors. Retest all, a random technique and a safe test method are the types of selection used for comparing purposes. Studies have shown the retesting method to have various cases that are simple and also introduce consumption of time at the time of software testing. And for this, there is a random technique which is simpler compared to the earlier method at the time of testing test cases that are chosen from this program. However, this will not be able to guarantee accuracy in terms of auditing software. One more safe technique of testing that provides better performance of reduction of test cases that are ineffective and in this, there are some bugs that are produced in comparison to the other old approaches [2].

The test case denotes a test set that takes all input values along with the observed output to be compared to the expected output. The primary feature of this test case was that it had the quality that covered the test objective and also contributed towards the reduction of software testing cost. Another problem in the testing of software is choosing suitable cases of tests from a test suit in connection to the program size. In case the size of the chosen test cases are big, it may be able to affect the performance of the life cycle of software development. In accordance with this, the testing time increases and further produces many more bugs.

The metaheuristic was a procedure of a higher level that identifies heuristics that provide an ideal solution to a problem of optimization. The Tabu Search (TS) is a technique of metaheuristic search which is based on a premise that says for qualifying to be intelligent, solving or problem has to include responsive exploration and adaptive memory. So, this Tabu Search algorithm is dependent on the next K-neighbours and maintains the Tabu list (for memory) of the neighbours that visited the Tabu. This rule features various parameters that were selected based on the problem and its ideal that needs to be solved. This objective was operating (fitness function) for measuring the cost of any solution and for this purpose, there was a suitable candidate list strategy used. This was to choose neighbour candidates that are good and for them to go beyond the local optimum without giving way to exploit or examine parts in the neighbourhood. This is necessary for outlining the short term memory and various other methods.

The hybrid optimization is structured by the strategies of communication among two different algorithms. The idea had been based on the replacement of weaker individuals in accordance with the evaluation of fitness for one algorithm that has stronger individuals from that of the other swarm intelligent algorithms in parallel processing. There are many groups in a structure which are parallel has been created by means of dividing the population into smaller subpopulations for constructing a new and parallel processing algorithm. Every subpopulation independently evolves with regular interactions. They exchange information among the population at the time the strategy of communication has been triggered. This may result in completely taking advantage of individual strengths for every algorithm, thus replacing weaker individuals with ones that are better. This can bring down the population size for every population and cooperation benefit is thus achieved.

The focus of this paper is hybridizing two evolutionary approaches inspired by nature which are the BAT and the Tabu used for the optimization of a test suite. These algorithms had made use of test history for generating the initial population. The calculation of fitness values was made by using running time and fault coverage of test cases. After this, only the fit tests will be carried forward to successive generations for reducing test suites until such time a stopping criterion is arrived at. The Hybrid BAT along with Tabu Search algorithm that had been proposed to choose test cases using a method of OO testing.

The literature that was based on the work had been detailed as in section 2. The techniques that were proposed had been discussed in section 3. All results obtained were explained duly in section 4 and section 5 concluded the work.

2. LITERATURE SURVEY

Thakur and Verma [3] had proposed a new Project Objective as Software Testing using Optimization Technique. The identification, automatic prioritization, and characterization of test cases found in software testing with techniques of optimization were used. Proposing a new approach in the process of software testing, optimizing of test effort, reliability, quality issues, and complexity of testing were employed. In the software development life cycle (SDLC), the most important phase was the testing phase. In the case of regression testing, there were many test cases which were impractical to be tested. So, in order to overcome the problem, the testing phase was done by making use of the chosen test cases for reducing efforts and for obtaining the results accurately.

Lawanna [2] had made a proposal for the software testing and its improvement in choosing small test cases by means of considering functions that were modified, the changed lines of code, the actual bugs produced after program modification. The reason behind proposing an improvement to software testing was to prepare an algorithm that was effective and the number of bugs was found to be lower than that of the traditional methods. Based on the results of the experiment, the size of the chosen test cases was made by means of the proposed model which was lower than the Retest All, the Random and the Safe Test of about 98.70%, 87.86%, and 84.67% respectively. Furthermore, the STI had an ability that was higher than comparative studies which were about 1- 20 times for the number of the bugs that were identified in the modification of the program.

Musa et al. [4] had made a modification to the presentation to reveal the selection of test cases for the object- oriented software that made use of an analysis of dependence graph in the source code. There was an experimental evaluation for this approach that was made by employing a total of nine programs. The performances of the approach of selection with inclusiveness metrics and selection metrics were made. The results proved that the approach was able to increase the efficiency of regression testing while looking at inclusiveness and precision. The conclusion was the choice of modification that revealed all test cases that were based on the statements providing better results for inclusiveness and precision in comparison to the random and the retest-all technique thus bringing down regression testing costs.

For improving the current techniques, there was another new technique which was a combination of the TABU Search and the GA that was presented by Miranda et al [5]. This hybrid technique was a combination of the strength of two different metaheuristic methods that produced test-case sequences that were efficient.

Agrawal and Kaur [6] had aimed at comparison of the performance of two different metaheuristics which were the Ant Colony and the Hybrid Particle Swarm Optimization. Enquiry domain for the paper was Test Case Selection and this was relevant in the case of software engineering needing a good treatment to effectively utilize the software. There were extensive experiments that were performed which made use of a standard flex object from the SIR repository. MATLAB was used for conducting these experiments and here the execution time along with fault coverage had been considered to be a measure of quality that has been reported in the paper and was used for analysis. The motivation behind the paper was the creation of awareness in two different aspects. Comparison of performance of these metaheuristic algorithms, and the demonstration of test case selection significance for software engineering.

Software testing has the main objective of locating the maximum number of bugs in the software by means of using optimum test cases. This optimum set of the test cases were obtained using the procedure of selection that is viewed to be a problem in optimization. Thus, metaheuristic optimizing techniques are used immensely for automating tasks of software techniques. Applying techniques of metaheuristic search in software testing was called Search-Based Testing. The reliable, non-redundant and optimized test cases that were generated using search-based testing making use of less time and effort. There was a systematic review based on various techniques like the TABU Search, Cuckoo Search, Bee Colony Optimization, Ant Colony Optimization, Particle Swarm Optimization, and Genetic Algorithm. There were modified versions for these algorithms as in Sahoo and Ray [7]. Authors provided a one framework, with advantages, future scope, and limitations in the works of research that help in further research of such work.

Kaur and Agrawal [8] had made an evaluation of two different metaheuristic algorithms which are the Bat Algorithm and the Cuckoo Search Algorithm. The factors that were taken into consideration of evaluation of performance were the faults that were detected during execution. The domain of this study was a flex object from a Benchmark repository – the Software Artifact and Infrastructure Repository. There were extensive experiments conducted for collecting and analysing results. There were a statistical test and the Ftest that had been conducted for validating research hypothesis. The results proved that Cuckoo Search Algorithms were able to perform better compared to the BAT algorithm.

3. METHODOLOGY

The section details on various datasets such as the Hybrid BAT with the TABU Search Algorithm, the BAT Algorithm, the TABU Search Algorithm, the JTOPas, the NanoXML, and the seina. There were three datasets that were employed for this experiment.

3.1 SIENA

Siena was a persistence API which was for the Java that had been inspired by a Google App Engine Python Datastore that attempts at drawing a bridge between the SQL and the NoSQL. It also provided the Java Object-DB mapping that was designed following an Active Record pattern to bring an intuitive and simple approach for managing Java objects in respect with the entities of the database.

3.2 NANOXML

The NanoXML denotes a small and non-validating parser used for the Java and this comes in various branches. It denotes a small XML based parser for Java [8]. The NanoXML was non-GUI based with easy-to-use systems that are available freely and are buildable from a source which will not have external libraries. The NanoXML was a non- validating parser used in Java. The approaches of NanoXML are found in diverse branches and are considered as a criterion parser recommended for the ones that prefer to remain autonomous among parsers using the SAX. This NanoXML/Lite was a descendant of a NanoXML 1 which is very small (only 6KB) and has traits that are faster as an algorithm. In case the NanoXML 1 was used, the user may not require to adopt a new code for the API and if coding applications are found to be small (like the embedded codes or the applets), this may be suitable. NanoXML/Lite has a functionality that is limited. Most of the branches were a selection of some classes that are found to be a common source tree.

- The NanoXML/Java was a standard parser which is recommended for the users.
- The NanoXML/SAX was a SAX adapter used for the NanoXML/Java. The branch was recommended for the ones that have to be independent of parsers or using the SAX.
- The NanoXML/Lite succeeded the NanoXML 1. This was quite small (just 6 KB) and the features were faster as an algorithm. It had been recommended only if the NanoXML 1 is used currently and the code cannot be adapted for a new API and in case the applications are coding which is very small.

The JTopas denotes a collection of the modules of Java that result from experiments or solutions of Java. This is a project which provides easy-to-use and small Java library for the problem of the parsing of text data that is arbitrary. Using a few of the alternations, it may be possible to extract the hyperlinks of meta-information of an HTML source. There were some more examples in the JUnit test cases that were provided in the library of the JTopas.

4. TABU SEARCH (TS) ALGORITHM

The TABU search is an approach to metaheuristics used for solving problems in optimization [5]. This has been designed in a manner in which other methods may be guided to move from the local optima. There are some traits of the

TABU Search that are its flexible memory structure that had been designed to ensure the criteria and information relating to the search have been exploited. The TABU has maintained two different types of memory, a short term one and a long term one. There are both intensification strategies and diversification strategies to help in the process of search in order to provide optimal results. The strategies of intensification will help in reinforcing the earlier solutions which are found. These strategies of diversification will help in searching for new ideas which are not explored in earlier situations. For avoiding getting stuck within local optima, there is a list that is created to maintain all recent solutions. This is known as the TABU list. This list contains some forbidden moves for preventing avoiding them from getting stuck within local optima. A TABU Search can look out for better solutions until such time testing criteria are arrived at. The pseudo-code for the TABU Search Algorithm is shown below:

Step 1: Create an initial solution n

Step 2: While i^{th} while stopping criteria is not met

Step 3: Create a set of solutions K that are the neighbors of n and that are not in Tabu list

Step 4: Choose a best solution n^* in K Update the Tabu list based on n^*

Step 5: Let $n=n^*$

The advantages of Tabu Search Algorithm is given below:

- This may be applied to the discrete and continuous spaces.
- Using of the TABU List
- A new meta-heuristic which guides procedures of local search for exploring solution spaces that are beyond their local optimality.
- For problems that are more challenging, TABU search has solutions that surpass the best ones found by alternate approaches.

4.1 BAT ALGORITHM

There are several algorithms that are bio- inspired in existence. The BAT algorithm is part of a class based on swarm intelligence. This was developed by Xin-She Yang in the year 2010. As it had been defined in Yang [9], the BAT algorithm will follow echolocation of the bats using sonar echoes for detection and avoidance of obstacles. This is called sound pulses that have been transformed into some more frequencies that have been reflected from the obstacles. There were three different generalized rules. All bats use echolocation to sense distance, and they also know the difference between food or prey and background barriers in some magical way. Bats fly randomly with velocity V_i at position X_i with different frequency ranges f [min,max], varying wavelength λ and loudness A_0 to search for prey. They can automatically adjust the wavelength of their emitted pulses and adjust the rate of pulse Emission $r \in [0,1]$ depending on the proximity of their target. The loudness varies from a large (positive) A_0 to a minimum value A_{min}. Each bat is randomly assigned a frequency between [f_{min},f_{max}], hence this algorithm is called as frequency tuning algorithm. Every bat is associated with velocity v_i and position in search space at each iteration t with respect to frequency f_i. Hence at each iteration we need to update f_i,v_i and x_i as per the following equation

$$f_i = f_{min} + (f_{max} - f_{min})\times\beta$$

where β is current frequency

$$V_{it} = v_{it-1} + (x_{it-1}-x^*)f_i$$

$$X_i = x_{it-1} + v_{it}$$

- All the bats make use of echolocation for sensing distance and guess differences between their prey or food and their background barriers in a magical manner.
- The bats fly in a random manner using velocity v_i at a position x_i using a fixed frequency which is f_{min} that had different loudness and wavelength the A0to that search for its prey. These will adjust automatically to the wavelength or frequency of emitted pulses and will also adjust the pulse rate emission r which is dependent on the target and its proximity.
- Even though loudness varies in different ways, the assumption is that its loudness will vary from the large (positive) A0 to the minimum constant value which is Amin. The algorithm is as follows [10]: Pseudo-code for the BAT is shown below.

Step 1: Objective function is $f(x)$, $x = (x_1,...,x_d)^T$

Step 2: Initializing of the bat population $x_i (i = 1,2,...,n)$ and v_i

Step 3: Now define pulse frequency f_i at x_i, initializing of pulse rates r_i and loudness A_i

Step 4: while $t <$ The max number of the iterations do

Step 5: Generation of new solutions by means of adjusting frequency, and updating velocities, as well as locations or solutions

Step 6: if rand $> r_i$ then

Step 7: Choose a new solution from among its best solutions

Step 8: Generation of a local solution which is around the chosen best solution

Step 9: end if

Step 10: Generation of a new solution by random flying

Step 11: if rand $< A_i$ and $f(x_i) < f(x*)$ then

Step 12: Accepting new solutions Increasing r_i and reducing A_i

Step 13: end if

Step 14: Ranking the bats and finding the current best x

Step 15: end while

Step 16: The post-process results and their visualization

4.2 PROPOSED HYBRID TABU SEARCH WITH BAT ALGORITHM

The hybrid optimization algorithm has been structured using strategies of communication among two different algorithms. The idea has been based on the replacement of individuals that are weaker in accordance with the evaluation of the fitness of an algorithm having stronger individuals that are from other algorithms found in parallel processing in algorithms that are swarm intelligent. There are many groups within the parallel structure that have been created by dividing the population into smaller subpopulations for constructing algorithms of parallel processing. Every subpopulation thus evolves in regular iterations. Information is exchanged only on triggering communication strategies. This may result in the individual strengths being taken advantage of. This may replace the individuals that are weaker with the one which is better from each other and also reducing the size of the population along with the benefit of cooperation which is achieved.

BAT algorithm and TABU Search hybridization have been proposed and here the Bat algorithm has the capacity to automatically zoom into the region that has promising results under the right conditions. The TABU search may restrict modification definition by means of elimination of solutions treated with the TABU list. The focus of this hybridization was on the modification phase of a new Bat Algorithm solution that takes advantage of the TABU to ensure the solution obtained has been untreated earlier.

5. RESULTS AND DISCUSSION

The Table.1 shows the simulation parameters. The Table.2 to Table.4 shows the code coverage for datasets like Siena, NanoXML and JTOPas respectively.

Table.1. Simulation parameters

Parameter	Value
Population Size	50
Minimum Frequency	0
Maximum Frequency	1
Initial Loudness	1
Loudness Adaption Parameter	0.95
Initial Pulse Rate	0.5
Pulse Rate Adaption Parameter	0.98
Standard Deviation	2
Penalty Coefficient	1
Size of Tabu List	15
Number of Iteration	1-500
Length of candidate Lists	48
Tabu length	10

Table.2. Code Coverage for Siena

Cost	Ref	Tabu	G	Bat-tabu	Imp%
10000	0.82	0.84	0.79	0.85	6%
20000	0.84	0.86	0.78	0.87	9%
30000	0.85	0.87	0.82	0.89	7%
40000	0.88	0.91	0.84	0.92	8%
50000	0.9	0.93	0.83	0.94	11%
60000	0.92	0.94	0.87	0.95	8%
70000	0.92	0.93	0.86	0.94	8%
80000	0.92	0.93	0.88	0.94	6%

The Table.2 shows that the code coverage for Siena dataset. The proposed Hybrid TABU with BAT algorithm yields best result than TABU search algorithm. The above table shows that BAT with TABU search yields 0.04625 % improved than TABU algorithm for using Siena dataset. This shows that the code coverage criteria is very close to the reference value. The reference value indicates the benchmark to be achieved. It is observed that the proposed BAT-TABU achieves the values closest to the reference when compared to BAT algorithm

Table.3. Code Coverage for NanoXML

Cost	Ref	Tabu	G	Bat-tabu	Imp%
10000	0.8	0.82	0.76	0.83	7%
20000	0.81	0.83	0.75	0.84	9%
30000	0.83	0.85	0.79	0.86	7%
40000	0.86	0.88	0.81	0.89	8%
50000	0.87	0.9	0.81	0.91	10%
60000	0.89	0.92	0.82	0.93	11%
70000	0.9	0.91	0.84	0.92	8%
80000	0.89	0.91	0.85	0.92	7%

The Table.3 shows that the code coverage for NanoXML dataset.The proposed Hybrid TABU with BAT algorithm yields best result than TABU search algorithm. The proposed BAT with TABU search yields 0.04875 % improved than TABU algorithm for using NanoXML dataset. The Table.3 further shows that the code coverage criteria is very close to the reference value. It is observed that the proposed BAT-TABU achieves the values closest to the reference when compared to BAT algorithm.

Table.4. Code Coverage for JTOPas dataset

Cost	Ref	Tabu	G	Bat-tabu	Imp%
10000	0.8	0.82	0.76	0.84	8%
20000	0.82	0.84	0.77	0.85	8%
30000	0.83	0.85	0.8	0.86	6%
40000	0.86	0.88	0.82	0.90	8%
50000	0.88	0.91	0.83	0.92	9%
60000	0.9	0.91	0.84	0.92	8%
70000	0.9	0.91	0.85	0.92	7%
80000	0.9	0.9	0.85	0.92	7%

The Table.5 shows that the code coverage for JTOPas dataset. The proposed Hybrid TABU with BAT algorithm yields best result than TABU search algorithm. This shows that the code coverage criteria is very close to the reference value. The proposed BAT with TABU search yields 0.0525 % improved than TABU algorithm for using JTOPas dataset. It is observed that the proposed BAT-TABU achieves the values closest to the reference when compared to BAT algorithm.

6. CONCLUSION

There are large test suites that consist of certain redundancies was the faults are covered using either two or more of these test cases. So it may be advisable to bring down the test suite. As the selection of the manual test case can get time-consuming and prone to errors, the search-based optimization is used to solve this by employing metaheuristic algorithms. A problem of optimization will be to identify an ideal solution from other solutions and for this optimization is needed. The BAT algorithm is a combination of some good features which are of some more nature-inspired metaheuristics. The BAT is a very powerful algorithm found in exploitation (or local search) and sometimes it may get trapped within local optima thus not being able to perform a global search. The proposed hybridization of BAT algorithm with Tabu search achieves the optimal code coverage.

REFERENCES

[1] M. Chaitra, M.K. Prakruthi and N.R. Sarala, "Optimizing Test Cases for Object-Oriented Software", *International Journal on Recent and Innovation Trends in Computing and Communication*, Vol: 4, No. 2, pp. 1-16, 2016.

[2] A. Lawanna, "An Effective Test Case Selection for Software Testing Improvement", *Proceedings of International Conference Computer Science and Engineering*, pp. 1-6, 2015.

[3] P.B. Thakur and T. Verma., "A Survey on Test case selection Using Optimization Techniques in Software Testing", *International Journal of Innovative Science, Engineering and Technology*, Vol. 2, No. 4, pp. 1-13, 2015.

[4] S. Musa, A.B.M. Sultan, A.B.A. Ghani and S. Bahaarom, "Regression Test Cases Selection for Object-Oriented Programs based on Affected Statements", *International Journal of Software Engineering and Its Applications*, Vol. 9, No. 10, pp. 91-108, 2015.

[5] T.B. Miranda, M. Dhinya and K. Sathyamoorthy, "Test Case Optimization Using Genetic and Tabu Search Algorithm in Structural Testing", *International Journal of Computer Application Technology and Research*, Vol. 4, No. 5, pp. 355-358, 2015.

[6] A.P. Agrawal and A. Kaur, "A Comprehensive Comparison of Ant Colony and Hybrid Particle Swarm Optimization Algorithms through Test Case Selection", *Data Engineering and Intelligent Computing*, pp. 397-405, 2018.

[7] R.R. Sahoo and M. Ray, "Metaheuristic Techniques for Test Case Generation: A Review", *Journal of Information Technology Research*, Vol. 11, No. 1, pp. 158-171, 2018.

[8] A. Kaur and A.P. Agrawal, "A Comparative Study of Bat and Cuckoo Search Algorithm for Regression Test Case Selection", *Proceedings of 7th International Conference on Cloud Computing, Data Science and Engineering*, pp. 164-170, 2017.

[9] X.S. Yang, "A New Metaheuristic Bat-Inspired Algorithm", *Proceedings of 7th International Conference on Nature Inspired Cooperative Strategies for Optimization*, pp. 331-337, 2010

[10] J. R. Gonzalez, "*Studies in Computational Intelligence*", Springer Publisher, 2012.

[11] M. Imane and K. Nadjet, "Hybrid Bat Algorithm for Overlapping Community Detection", *IFAC Papers Online Journal*, Vol. 49, No. 12, pp. 1454-1459, 2016.