International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-3, September 2019

Selection of Best Test Cases using Support Vector Machine for ARPT

K. Devika Rani Dhivya, V. S. Meenakshi

Abstract*: Software testing is an essential activity in software industries for quality assurance; subsequently, it can be effectively removing defects before software deployment. Mostly good software testing strategy is to accomplish the fundamental testing objective while solving the trade-offs between effectiveness and efficiency testing issues. Adaptive and Random Partition software Testing (ARPT) approach was a combination of Adaptive Testing (AT) and Random Partition Approach (RPT) used to test software effectively. It has two variants they are ARPT-1 and ARPT-2. In ARPT-1, AT was used to select a certain number of test cases and then RPT was used to select a number of test cases before returning to AT. In ARPT-2, AT was used to select the first m test cases and then switch to RPT for the remaining tests. The computational complexity for random partitioning in ARPT was solved by cluster the test cases using a different clustering algorithm. The parameters of ARPT-1 and ARPT-2 needs to be estimated for different software, it leads to high computation overhead and time consumption. It was solved by Improvised BAT optimization algorithms and this approach is named as Optimized ARPT1 (OARPT1) and OARPT2. By using all test cases in OARPT will leads to high time consumption and computational overhead. In order to avoid this problem, OARPT1 with Support Vector Machine (OARPT1-SVM) and OARPT2-SVM are introduced in this paper. The SVM is used for selection of best test cases for OARPT-1 and OARPT-2 testing strategy. The SVM constructs hyper plane in a multi-dimensional space which is used to separate test cases which have high code and branch coverage and test cases which have low code and branch coverage. Thus, the SVM selects the best test cases for OARPT-1 and OARPT-2. The selected test cases are used in OARPT-1 and OARPT-2 to test software. In the experiment, three different software is used to prove the effectiveness of proposed OARPT1-SVM and OARPT2-SVM testing strategies in terms of time consumption, defect detection efficiency, branch coverage and code coverage.

Keywords* : Adaptive and Random Partitioning Testing, Adaptive Testing, Random Partitioning Testing, Software testing, Support Vector Machine.

I. INTRODUCTION

Software testing [6] can be conducted either manually or in an automated manner. In manual testing, a human tester takes over the role of an end-user interacting with and executing the software under test to verify its behavior and to find any observable defects. On the other hand, in automated testing, test code scripts are developed using certain test tools and are then executed without human tester intervention to test the behavior of the software under test. If the automated testing is implemented properly, it could yield various benefits over manual testing such as reduction of test efforts and repeatability. Otherwise, automated testing will lead to extra costs and effort and could even less effective than manual testing in detecting faults [4].So it is more important to develop a testing strategy to test software effectively. Adaptive and Random Partitioning Testing (ARPT) [8] is a software testing strategy used to test software. It has two viable options are ARPT-1 and ARPT-2 to combine Adaptive Testing (AT) and Random Partitioning Testing (RPT) in a testing process. However, the computation complexity is high for random partitioning of ARPT.

In order to improve the random partitioning in ARPT, test cases were clustered [2] by using Expectation Maximization (EM), K-means, Non Negative Matrix Factorization (NMF) and Self-Organizing Map (SOM) algorithms. Moreover, the ARPT-1 and ARPT-2 consist of various parameters it needs to be estimated over and over for different software.

It may lead to time consumption and computational overhead for ARPT testing strategy. So, Optimized ARPT1 (OARPT1) and OART2 were introduced which used Improvised BAT (IBAT) [3] for automatic parameter selection of ARPT-1 and ARPT-2 for different software. A huge number of test cases are collected for OARPT testing strategy. If all the test cases are used in OARPT testing strategies, it may lead to high computational overhead and time consumption.

So in this paper, best test cases are selected by using an efficient machine learning algorithm called Support Vector Machine (SVM). It selects the best test cases based on the code coverage and branch coverage of test cases. The test case which has high code coverage and high branch coverage is selected as best test cases. This methodology is applied for both ARPT-1 and ARPT-2 testing strategy and it is named as OARPT1-SVM and OARPT2-SVM respectively. It reduces the time consumption and computational overhead of OARPT testing strategy.

The remaining section of the article is scheduled as follows: Section II investigates about the different test case selection approaches which are related to the proposed system. Section III explains the methodology of the proposed software testing strategy. Section IV illustrates the experimental results by using the software. Section V concludes and discusses the proposed work.

Manuscript published on 30 September 2019
* Correspondence Author
K. Devika Rani Dhivya,*, Research Scholar, Bharathiar University, Coimbatore, Tamilnadu, India, devika58@gmail.com.
Dr. V. S. Meenakshi, Assistant Professor, Department of Computer Science, Chikkanna Government Arts College, Tirupur, Tamilnadu, India, meenasri70@yahoo.com.

II. LITERATURE SURVEY

Multi-Objective Genetic Algorithms (MOGAs)[10]was suggested to improve multi objective test case selection by injecting diversity in genetic algorithms. The main intension of the proposed algorithm was promoting diversity between solutions which are candidate sub-test suites in the genotype space. The higher genotype diversity indicated that different solutions select different sets of test cases. Then, two genetic operators were introduced for orthogonal design to build a diversified initial population and for orthogonal exploration of the search space through Singular Value Decomposition (SVD). The orthogonal exploration was acted during evolution by injecting individuals that according to the SVD were diversified enough. Hence the Diversity based Genetic Algorithm (DIV-GA) combined with orthogonal design and orthogonal evolution into MOGA to solve multi criteria test case selection problem. However, the efficiency of the projected algorithm is low.

A Similarity Approach for Regression Testing (SART) [1] was presented a selection of best test cases. That approach combined a similarity based test case selection technique with model based testing approaches. It automatically identified test cases exercising new, modified or affected parts of the specification model using only information from test cases. It compared two sets of test cases from a baseline and a delta version of the specification model. As the test cases were described through steps indicated that expensive modifications were performed to a point where the sequences of steps had significantly changed. The usage of SART was allowed to automatically select the desired subset of test cases. It was then combined with the test case generation the robustness of the SART was improved. It is a feasible and quick alternative for automatic test case selection. But it takes more time to analyze the test results.

An approach was projected [7] for the selection of best test cases for regression testing of different versions of Business Process Execution Language (BPEL). The changes in composite services mainly covered three types are the processes, bindings and interfaces. By the this approach, the changes of composite services were identified by performing control flow analysis and compared the paths in a new version of composite service with those in the old one using a kind of extensible BPEL flow graph (XBFG). Message sequence was added to XBFG path so that XBFG can fully describe the behavior of composite service. The predicate and binding constraint information include in different XBFG elements which were used to select the path and to generate test cases. However, it is applied only for regression testing.

A Software Testing Improvement (STI) [5] was suggested to increase the ability of test case selection regarding the concept of regression test selection which is Retest- All (RA), Random Technique (RT) and Safe Test Technique (ST) method. STI was developed by considering the changes in the requirements and modified programs. It was also used to predict the number of functions modified and lines of codes changed by using four steps are finding the function modified, determining the lines of codes changed, finding the approximate number of test cases and selecting the test cases. The output of predicting the functions modified and lines of code changed was used for the new software version. However, the STI may not cover some important situations such as knowledge and skill of testers, limitation of testing software and the environment by using the software.

Adaptive Testing with Gradient Descent (AT-GD) [9] was offered to investigate the asymptotic behavior of adaptive testing and improving the global performance without losing local optimality of adaptive testing. It was a local optimal testing strategy converges to the globally optimal solution as the assessment process proceeds. The Gradient was extensively utilized in deciding a search direction when a step size choice was created to solve an optimization problem. This was introduced in the original Adaptive Testing framework which investigated the asymptotic behavior of AT-GD and the upper bound of AT strategies was explored. But this method still has computational overhead problem.

From the previous researchers associated with the software testing strategies for test case selection were discussed in detail. The problems in test case selection are solved by a Machine learning algorithm, which is studied in detail. This study is useful for identifying the challenges involved in test case selection and based on these challenges, the proposed solution for the best test case selection is presented.

III. PROPOSED METHODOLOGY

In this section, the proposed methodology for test case selection using SVM is described in detail. The ARPT consists of two variants are ARPT-1 and ARPT-2. In ARPT-1, AT is used to choose a certain number of test cases and then RPT is used to select a number of test cases before returning to AT. In ARPT-2, AT is used to choose the first x test cases and then switch to RPT for remaining tests. Initially, random partitioning in ARPT testing strategy is improved by grouping similar test cases using EM, K-means, NMF and SOM. Then, the parameters of ARPT-1 and ARPT-2 are optimally selected by OARPT-1 and OARPT-2. But, the best test cases have to select for OARPT-1 and OARPT-2 testing strategy. In order to select the best test cases for OARPT-1 and OARPT-2, efficient machine learning SVM is introduced to select the best test cases for OARPT-1 and OARPT-2 testing strategy rather than selecting a certain number of test cases. The SVM is applied to OARPT-1 and OARPT-2 individually to select the best test cases of OARPT-1 and OARPT-2.

A. Support Vector Machine based test case selection for ARPT 1 and ARPT 2

OARPT-1 and OARPT-2 are two variants of ARPT testing strategy which consists of two segments are AT segment and RPT segment. In OARPT-1, AT is processed with $l(i)$ test cases and RPT is processed with $k(i)$ test cases. In OARPT-2 for m testing length, AT testing process is carried out and in the remaining length the RPT testing strategy is carried out. SVM is applied individually to select the best test cases for OARPT-1 and OARPT-2. It is hypothetically a very well-motivated algorithm [11].

It selects the best test cases based on code coverage and branch coverage. Generally, the SVM is based on the concept of decision planes that define decision boundaries. A decision plane is the one which separates between a set of objects having different class membership. It constructs hyper planes in a multi-dimensional space which is used to separate test cases which have high code and branch coverage and test cases which have low code and branch coverage. The code coverage determines how much code is being tested. It ensures to maintain the test quality over the life cycle of a project and to know how well ARPT testing strategy actually tests the code. It can be calculated based on a coverage matrix. Let T be the set of test cases selected and C be set of code elements [12], Then,

$$cov(T, C) = \frac{|c \in C | c \text{ covered by } T|}{|C|} \quad (1)$$

The goal of branch coverage is to execute all of the branches in the program. Branches are the outgoing edges from a decision point. In other words, the branch coverage C_{Branch} of T for P is the fraction of the branches of program P executed by at least one test case in T. The branch coverage is calculated as [13],

$$C_{Branch} = \frac{Num_{exe\_branch}}{Num_{branch}} \times 100\% \quad (2)$$

where,$'Num_{exe\_branch}'$ denotes the number of executed branches, $'Num_{branch}'$denotes the number of branches. These measures are used in the SVM to select the best test cases for ARPT-1 and ARPT-2 testing strategy. The training database of SVM is denoted as$X = \{(x_1, y_1), \dots (x_n, y_n)\}$, $x_i \in R^n, y_i \in \{-1, +1\}$represents two different classes, where +1 denotes the class with test cases has high code coverage and branch coverage and -1 denotes the class with test cases has low code coverage and branch coverage. Based on the code coverage and branch coverage of the test cases, the SVM algorithm finds an optimal hyper plane, which requires solving the following optimization problem.

Maximize

$$\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1, i \neq j}^{n} \alpha_i \alpha_j \, y_i y_j K(x_i, x_j)$$

Subject to,

$$\sum_{i=1}^{n} \alpha_i \, y_i = 0 \quad (3)$$

where, 'n' denotes the number of test cases, $'\alpha_i'$ is the weight of the training sample. The weight of the training samples (test cases) is assigned based on the code coverage and branch coverage of test cases in ARPT 1 testing strategy. The highest weight value is assigned to test cases which obtain high code coverage and branch coverage. The kernel function K is used to measure the similarity between two test cases and is defined as Radial Basis Function (RBF).

$$K(x_i, x_j) = exp\left(-\gamma \|x_i - x_j\|^2\right), \gamma > 0 \quad (4)$$

After the assignation of weights based on code coverage and branch coverage, the test samples $'x'$ is classified as,

$$y = Sign\left(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x_j)\right) \quad (5)$$

$$Sign(i) = \begin{cases} +1, \ if \ i > 0 \\ -1, \ Otherwise \end{cases} \quad (6)$$

The value of γ is determined by the cross validation process which is carried out on the training database. The cross validation process is used for estimating the generalization capability on new samples which are not in the training dataset. The training dataset is divided into k subsets with equal size by using k-fold cross validation and classifier are constructed by using the remaining samples for classification. This process is continued for k times on each subset for obtaining the cross validation over the entire training database. Based on SVM classification, the best test cases are selected for OARPT-1 and OARPT-2. The selected best cases are used to test software. By using the best test cases, the defect detection efficiency of OARPT-1 and OARPT-2 testing strategies are improved.

Algorithm: SVM based test case selection

Input:

1. Initialize the input sample (test case) $\mathbf{x}$ to classify
2. Consider the training database $X = \{(x_1, y_1), \dots (x_n, y_n)\}$
3. Assume *k* number of nearest neighbors

Output:

4. *Decision* $y_p \in \{-1,1\}$

Test case selection:

5. Obtain k sample (x_i, y_i) with minimal values of $K(x_i, x_j) - 2 * K(x_i, x)$
6. Train the SVM model on the selected k samples
7. Classify x and obtain the result y_p
8. Return y_p

IV. ILLUSTRATION

Jsoup is a Java library for working with real-world HTML. It provides a very convenient API for extracting and manipulating data, using the best of DOM, CSS, and jquery-like methods. Let, selected Training Test samples for jsoup-master.

Table- I: Training Test Samples

S. No.	Training Test Samples (Test Cases)
1.	testNamespacedTag
2.	testByAttributeStarting
3.	testByAttributeRegex
4.	deeper Descendant
5.	parentWithClassChild
6.	parentChildStar
7.	adjacentSiblings
8.	testCharactersInIdAndClass
9.	testPseudoLessThan
10.	testPseudoBetween

These test cases are tested by ARPT-1 and ARPT-2 selector selects test .The tested results are 1 if the test case is passed otherwise test result is 0. These test cases are the test components of the selector selects test in jsoup -maser. If the selector selects test is a success means the 'Y' is represented as -1.

X – Trained Test sample by ARPT1 and ARPT2										Y
1	2	3	4	5	6	7	8	9	10	
1	1	1	1	0	1	0	1	1	0	-1
1	0	1	0	1	0	1	0	0	1	-1
1	1	0	1	1	0	0	1	1	0	1
0	0	0	1	1	1	1	0	0	1	1
0	1	0	0	0	0	1	0	1	1	1
0	0	0	1	1	1	1	0	1	0	-1

Iteration 1

c_optimal=0.1000; alpha=0.1000

	X										Y
	1	2	3	4	5	6	7	8	9	10	
	0.6211	-0.7746	-1.1619	0.7746	0.7746	0.6211	-1.1619	0.9487	-0.9487	0.9487	-1
	0.6211	1.1619	0.7746	0.7746	-1.1619	1.1619	-0.6211	0.9487	0.9487	-0.9487	-1
	0.6211	-0.7746	0.7746	-1.1619	0.7746	-0.7746	1.4491	-0.9487	-0.9487	0.9487	1
	- 1.4491	1.1619	0.7746	0.7746	-1.1619	-0.7746	-0.6211	-0.9487	0.9487	-0.9487	1
	-1.4491	-0.7746	-1.1619	0.7746	0.7746	1.1619	1.4491	-0.9487	-0.9487	0.9487	1
	0.6211	-0.7746	0.7746	-1.1619	-1.1619	-0.7746	-0.6211	0.9487	-0.9487	0.9487	-1
W	-0.1242	-0.2324	-0.1549	-0.1549	0.0387	-0.0387	0.1242	0.1897	-0.1897	0.1897	

.
.
.

	X										Y
	1	2	3	4	5	6	7	8	9	10	
	0.6211	-0.7746	-1.1619	0.7746	0.7746	1.1619	-0.6211	-0.9487	-0.9487	0.9487	-1
	0.6211	1.1619	0.7746	0.7746	-1.1619	1.1619	-0.6211	0.9487	0.9487	-0.9487	-1
	0.6211	-0.7746	0.7746	-1.1619	0.7746	-0.7746	1.4491	-0.9487	-0.9487	0.9487	1
	- 1.4491	1.1619	0.7746	0.7746	-1.1619	-0.7746	-0.6211	-0.9487	0.9487	-0.9487	1
	-1.4491	-0.7746	-1.1619	0.7746	0.7746	1.1619	1.4491	-0.9487	-0.9487	0.9487	1
	0.6211	-0.7746	0.7746	-1.1619	-1.1619	-0.7746	-0.6211	0.9487	-0.9487	0.9487	-1
W	-0.8270	-0.3762	-0.6632	-0.3172	-0.1001	-0.1973	0.1391	1.1037	-0.0725	0.0725	

Iteration 10

alpha = 0.0983

N-Test - Test sample to predict										Y	Actual
1	2	3	4	5	6	7	8	9	10		
1	1	1	0	0	0	0	1	0	1		1
1	0	0	1	1	1	0	0	0	1		-1
0	1	1	1	0	0	0	0	1	0		-1
1	0	1	0	1	0	0	1	1	0		1
-.8270	**-.3762**	**-0.6632**	**-.3172**	**-.1001**	**-0.1973**	**0.1391**	**1.1037**	**-0.0725**	**0.0725**		

Iteration 1

N – Test - Test sample to predict										Y
1	2	3	4	5	6	7	8	9	10	
0.6210	-0.7745	0.7745	-1.1618	0.7745	-0.7745	-0.6210	0.9486	0.9486	-0.9486	0.9486
0.6210	1.1618	0.7745	-1.1618	1.1618	-0.7745	-0.6210	-0.9486	-0.9486	0.9486	-0.9486
0.6210	-0.7745	0.7745	-1.1618	0.7745	-0.7745	1.4491	0.9486	-0.9486	0.9486	-0.9486
0.6210	1.1618	-1.1618	0.0163	0.7745	-0.7745	-0.6210	0.9486	0.9486	-0.9486	0.9486

.
.

N – Test - Test sample to predict										Y
1	2	3	4	5	6	7	8	9	10	
0.6210	-0.7745	0.7745	-1.1618	0.7745	-0.7745	-0.6210	0.9486	0.9486	-0.9486	0.9486
0.6210	1.1618	0.7745	-1.1618	1.1618	-0.7745	-0.6210	-0.9486	-0.9486	0.9486	-0.9486
0.6210	-0.7745	0.7745	-1.1618	0.7745	-0.7745	1.4491	0.9486	-0.9486	0.9486	-0.9486
0.6210	1.1618	-1.1618	0.0163	0.7745	-0.7745	-0.6210	0.9486	0.9486	-0.9486	0.9486

Iteration 10

N – Test - Test sample to predict										Y
1	2	3	4	5	6	7	8	9	10	
0.6210	1.1618	0.7745	-1.1618	0.7745	-0.7745	-0.6210	0.9486	0.9486	-0.9486	0.9486
0.6210	1.1618	0.7745	-1.1618	0.7745	-0.7745	-0.6210	-0.9486	-0.9486	0.9486	-0.9486
0.6210	-0.7745	0.7745	-1.1618	0.7745	-0.7745	1.4491	0.9486	-0.9486	0.9486	-0.9486
0.6210	1.1618	-1.1618	-1.1618	0.7745	-0.7745	-0.6210	0.9486	0.9486	-0.9486	0.9486

The predicted result for test case result

Predicted	Actual
0.9486	**1**
-0.9486	**-1**
-0.9486	**-1**
0.9486	**1**

This illustration result is obtained from a trained model of ARPT-1 and ARPT-2 testing results. It shows the classification of the selected best test case by using the machine learning algorithm called Support vector machine. 'X' Trained Test sample by ARPT1 and ARPT2 and 'Y' class label, 'W' weighted value are used to obtain the process of selecting best test cases.

V. RESULT AND DISCUSSION

In this section, the experimental results are conducted and analyzed in terms of time consumption, defect detection efficiency, branch coverage and code coverage to prove the effectiveness of the proposed OARPT1-SVM and OARPT2-SVM testing strategies. Univocityparser, marc4j and jsoup software are used in the experiment. Table 2 shows the software description.

TABLE- II: Software Description [3]

S. No.	Software	No. of Test cases	No. of Programs
1.	Univocityparser	1000	137
2.	Marc4j	1000	93
3.	Jsoup	1000	56

A. Time Consumption

Time consumption is a measure of the amount of the time taken to test software by using the best test cases selected by the proposed method.

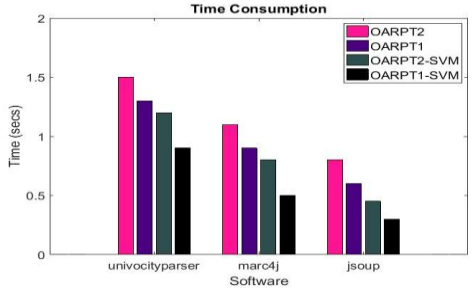

Fig. 1. Comparison of Time Consumption

Fig. 1 shows the comparison of time consumption by using OARPT1, OARPT2, OARPT1-SVM and OARPT2-SVM software testing strategies in terms of seconds. It is tested in three software are univocityparser, marc4j and jsoup software. In univocityparser software, the time consumption of OARPT2-SVM testing strategy is 20% lower than OARPT2 and the time consumption of OARPT1-SVM testing strategy is 30.77% lower than OARPT1. From the analysis, it is known that the OARPT1-SVM and OARPT2-SVM testing strategy has better time consumption than the other testing strategies.

B. Defect Detection Efficiency

Defect detection efficiency (DDE) is the number of defects detected during a phase/stage that is injected during that same phase divided by the total number of defects injected during that phase. It can be calculated by using the following formula [3]:

$$DDE = \frac{No.\,of\ defects\ injected\ AND\ Detected\ in\ a\ phase}{Total\ No.\,of\ Defects\ injected\ in\ that\ phase} \times 10$$

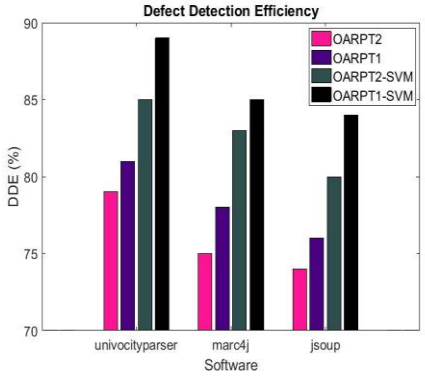

Fig. 2. Comparison of Defect Detection Efficiency

Fig. 2 shows the comparison of defect detection efficiency by using OARPT1, OARPT2, OARPT1-SVM and OARsPT2-SVM software testing strategy in terms of %. It is tested in three software are univocityparser, marc4j and jsoup software. In univocityparser software, the defect detection efficiency of OARPT2-SVM is 7.59% greater than OARPT2 and the defect detection efficiency of OARPT1-SVM is 9.88% greater than OARPT1. From the analysis, it is understood that the OARPT1-SVM and OARPT2-SVM testing strategy outperforms than the other testing strategies.

C. Code Coverage

Code coverage is a measure utilized to define the degree to which the source code of a program is executed when a particular test suite runs.

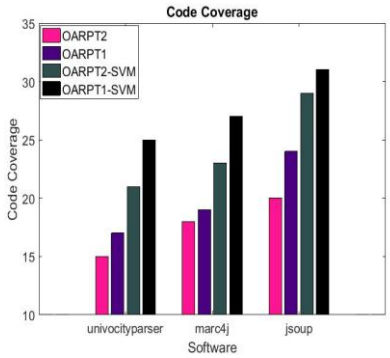

Fig. 3. Comparison of Code Coverage

Fig. 3 shows the comparison of code coverage by using OARPT1, OARPT2, OARPT1-SVM and OARPT2-SVM software testing strategy in terms of %. It is tested in three software are univocityparser, marc4j and jsoup software. In univocityparser software, the code coverage of OARPT2-SVM is 40% greater than OARPT2 and the code coverage of OARPT1-SVM 40.06% greater than OARPT1 testing strategy. From the analysis, it is proved that the OARPT1-SVM and OARPT-2 testing strategy has high code coverage than the other testing strategies.

D. Branch Coverage

Branch coverage is described in terms of test requirements and coverage measure. The branch coverage is calculated as,

$$C_{Branch} = \frac{Num_{exe_{branch}}}{Num_{branch}} \times 100\%$$

where,$'Num_{exe\_branch}'$ denotes the number of executed branches, $'Num_{branch}'$denotes the number of branches.

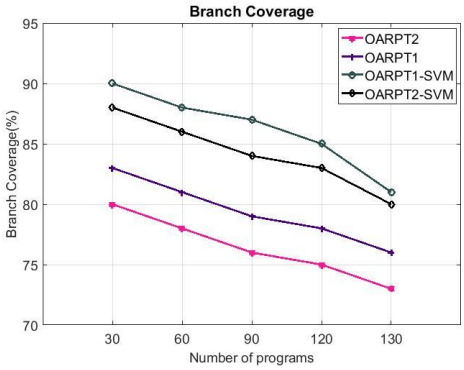

Fig. 4. Comparison of Branch Coverage for Univosityparser Software

Fig. 4 shows the comparison of branch coverage by using OARPT1, OARPT2, OARPT1-SVM and OARPT2-SVM software testing strategy for univocityparser software. When the number of programs is 90, the branch coverage of OARPT1-SVM is 10.13% greater than OARPT1 and the branch coverage of OARPT2-SVM is 10.53% greater than OARPT2. From this analysis, it is proved that the proposed OARPT1-SVM and OARPT2-SVM has better branch coverage than OARPT1 and OARPT2 testing strategies for univocityparser.

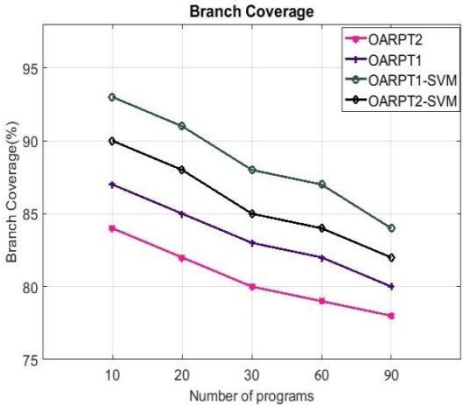

Fig. 5. Comparison of Branch Coverage for marc4j Software

Fig. 5 shows the comparison of branch coverage by using OARPT1, OARPT2, OARPT1-SVM and OARPT2-SVM software testing strategy for marc4jsoftware. When the number of programs is 30, the branch coverage of OARPT1-SVM is6.02% greater than OARPT1 and the branch coverage of OARPT2-SVM is 6.25% greater than OARPT2. From this analysis, it is proved that the proposed OARPT1-SVM and OARPT2-SVM has better branch coverage than OARPT1 and OARPT2 testing strategies for marc4j software.

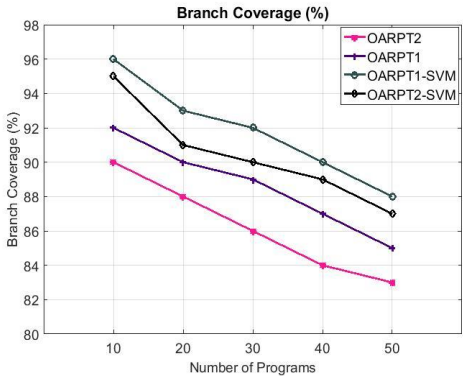

Fig. 6. Comparison of Branch Coverage for jsoup Software

Fig. 6 shows the comparison of branch coverage by using OARPT1, OARPT2, OARPT1-SVM and OARPT2-SVM software testing strategy for jsoup software. When the number of programs is 30, the branch coverage of OARPT1-SVM is 3.37% greater than OARPT1 and the branch coverage of OARPT2-SVM is 4.65% greater than OARPT2. From this analysis, it is proved that the proposed OARPT1-SVM and OARPT2-SVM has better branch coverage than OARPT1 and OARPT2 testing strategies for jsoup software.

E. Defect Detection

Each testing strategy chooses a particular number of test cases to test software. Following figure 7, 8 and 9 shows the number of test cases selected by OARPT1, OARPT2, OARPT1-SVM and OARPT2-SVM testing strategies and the red line indicates at that number of test cases the defects in the software are detected.

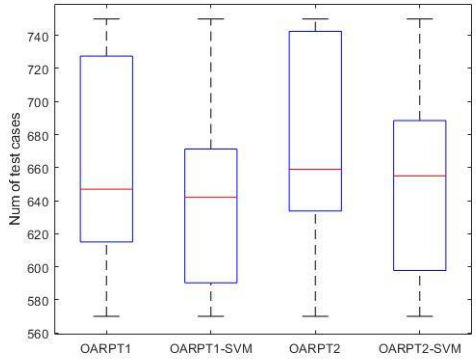

Fig. 7. Comparison of Defect Detection in Univosityparser Software

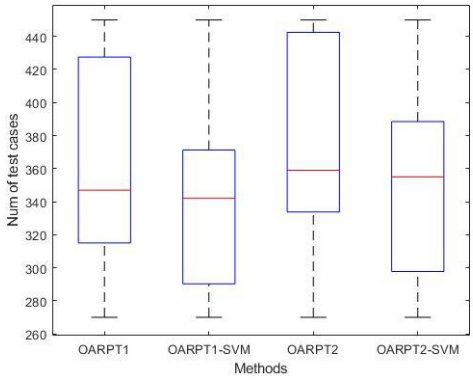

Fig. 8. Comparison of Defect Detection in marc4j Software

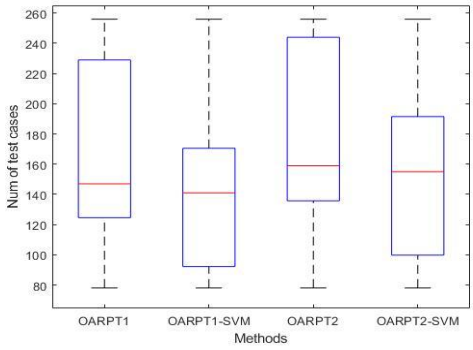

Fig. 9. Comparison of Defect Detection in jsoup Software

According to the above results, the performance of OARPT with SVM accomplishes better results such that it reduces time conception, improved defect detection efficacy with better code and branch coverage.

VI. CONCLUSION

In this paper, the computational overhead and time consumption of OARPT testing strategy is reduced by selecting the best test cases. An efficient machine learning algorithm called Support Vector Machine (SVM) is used to select the best test cases for OARPT-1 and OARPT-2 testing strategy based on code coverage and branch code coverage of test cases. The test cases which have high code coverage and branch coverage are chosen as best test cases in OARPT-1 and OARPT-2. Finally, the selected best test cases are used to test software. The experimental results prove that the proposed OARPT1-SVM and OARPT2-SVM have better time consumption, defect detection efficiency, branch coverage and code coverage than OARPT1 and OARPT2 testing strategies for Univocityparser, marc4j and jsoup software. In future, the parameters of ARPT 1 and ARPT 2 can be unified to get better results.

REFERENCES

1. F. G. De Oliveira Neto, R. Torkar, and P. D. Machado, “Full modification coverage through automatic similarity-based test case selection”, Inf. Softw. Technol., vol. 80, pp. 124-137, 2016.
2. K. Dhivya Rani Devika, and V. S. Meenakshi, “Improved Time Performance of Adaptive Random Partition Software Testing by Applying Clustering Algorithms”, Intern. J. Adv. Res. Comput. Sci., vol. 8, 2017.
3. K. Devika Rani Dhivya, and V. S. Meenakshi, “Comparative Analysis on Parameter Optimization for ARPT”, Int. J. Comput. Sci. Eng., vol. 6, pp. 349-354, 2017.
4. V. Garousi, and B. Küçük, “Smells in software test code: A survey of knowledge in industry and academia”, J. Syst. Softw., vol. 138, pp. 52-81, 2018.
5. A. Lawanna, “An effective test case selection for software testing improvement”, In: IEEE 2015 Int. Comput. Sci. Eng. Conf. (ICSEC), pp. 1-6, 2015.
6. O. A. L. Lemos, F. F. Silveira, F. C. Ferrari, and A. Garcia, “The impact of Software Testing education on code reliability: An empirical assessment”, J. Syst. Softw., vol. 137, pp. 497-511, 2018.
7. B. Li, D. Qiu, H. Leung, and D. Wang, “Automatic test case selection for regression testing of composite service based on extensible BPEL flow graph”, J. Syst. Softw., vol. 85, pp. 1300-1324, 2012.
8. J. Lv, H. Hu, K. Y. Cai, and T. Y. Chen, “Adaptive and random partition software testing”, IEEE Trans. Syst., Man, Cybern.: Syst., vol. 44, pp. 1649-1664, 2014.
9. J. Lv, B. B. Yin, and K. Y. Cai, “On the asymptotic behavior of adaptive testing strategy for software reliability assessment”, IEEE trans. Softw. Eng., vol. 40, pp. 396-412, 2014.
10. A. Panichella, R. Oliveto, M. Di Penta, and A. De Lucia, “Improving multi-objective test case selection by injecting diversity in genetic algorithms”, IEEE Trans. Softw. Eng., vol. 41, pp. 358-383, 2015.
11. G. Rishu, “Software Defects Prediction Using Support Vector Machine”, Int. J. Comput. Sci. Softw. Eng., vol.1, pp. 39-48, 2015.
12. D. Tengeri, Á. Beszédes, D. Havas, and T. Gyimóthy, “Toolset and program repository for code coverage-based test suite analysis and manipulation”, In 2014 IEEE 14th Int. Working Conf. Source Code Anal. Manip., pp. 47-52, 2014.
13. M. Pezze, and M. Young, “Software testing and analysis: process, principles, and techniques”, John Wiley & Sons, 2008.