Optimization Algorithm for Cost-Aware Test Suite Minimization in Software Testing

Faiz Baothman
Department of Computer Science, College of Computers and Information Technology Taif University, Taif, Saudi Arabia
f.baothman@tu.edu.sa

Abstract: Regression testing value is optimized by reducing subsets of check cases from a check suite while not compromising the check demand. Researchers have given varied test-suite reduction technique mistreatment coverage metrics and greedy search algorithms. Besides greedy algorithms, optimization-based algorithms have contended a significant role in check suite reduction. Consequently, we tend to developed a brand new optimization rule, rule to handle the variety drawback in generating new solutions whereas finding the optimum check cases. Here, a fitness operate is developed to pick the check cases optimally through the rule mistreatment 2 constraints, satisfying the complete check demand and minimizing the value live. The planned rule is experimented with 5 programs from SIR mistreatment four completely different analysis metrics. The empirical study on the performance of the rule is analyzed with varied parameters and therefore the comparison is completed with the greedy-based rule and therefore the pulsation Genetic Search (SGS) rule. The experimental outcome showed that the planned rule outperformed the present rule in reaching the marginal value necessities.

Keywords: BAT algorithm, Optimization, Test case, Test suite, Greedy, Objective based, minimization.

I. Introduction

The software applications in Banking, Medical, and Commercial Applications are subjected to extremely intricate verification and validation procedures which involve some different tasks [1]. One of the validation procedure which helps in improving the quality of the software is software testing and is the most important method which guarantees the quality of the developing software. Recently, Regression testing is the most often used maintenance process which revalidates the modified software. As the size of the test suite grows, the cost of regression testing increases. It happens because as the software is modified, the new test cases are added to test changed requirements. Test-suite size problem is addressed by two approaches namely test-suite reduction and test selection. Test-suite reduction is also known as test set minimization algorithms [3] which identify the minimized test suite that provides the same coverage of the software as the original test suite. In test-suite selection, a subset of the test suite that will execute code or entity changes is selected by the test selection algorithms. EIrreplaceability is a recent metric that enables decrementing the number of test cases through greedy search algorithm [2].

The approaches presented in the literature for test suite reduction are classified into four major types, i) Measure based test suite reduction, ii) Greedy search-based test suite reduction, iii) Optimization-Search based test suite reduction and iv) Multi-Objective-based test suite reduction. In measure based test suite reduction, coverage-based variants are widely applied as like [2, 3, 17, 18] for test suite reduction. The greedy search based techniques are utilized the different criteria and constraints to find the optimal test suite as like [6, 15]. In optimization-based testing, genetic algorithm, PSO algorithm is widely applied for test suite reduction. The genetic algorithm-based test suite reduction can be found in [5, 19, 23, 24, 25]. In this work, we bring an optimization algorithm called, PBAT (Poly BAT) algorithm to select test cases optimally with the constraint that test suites should satisfy all the test requirements. At first, initial solutions are generated randomly with the constraint that selected test cases in each and every solution should satisfy the entire test requirement. Then, fitness is evaluated using the total cost which is the aggregated execution time of all the selected test cases. The solution set which has the minimum aggregated cost measure is then selected as the best solution set with the help of the proposed PBAT algorithm.
II. Related Works

Irreplaceability metric is incorporated with the existing test case metric Ratio using the well-known test suite reduction algorithms, such as Greedy, GRE, and HGS. This method attains a low cost reduction strategy to yield a high level of test coverage. Reetika Nagar et al. proposed hybrid Particle Swarm Optimization (PSO) algorithm for test suite reduction [26]. This method is effective in choosing the minimum set of test cases that possess the possibility of the faults and bugs for which it takes minimum time. Martín Pedemonte et al. [28] proposed a Systolic Genetic Search (SGS) algorithm to solve the real-world problem like the Test Suite Minimization Problem (TSMP) existing in the field of software engineering. This algorithm serves as a best method and it is highly effective for the TSMP with the excellent scalable behavior. Irreplaceability and Irreplaceability for test suite reduction were introduced. The reduction of the test suite is done with these metrics and greedy search algorithm which is one of the popular algorithms for the search process. The objective of this research is to develop an effective test suite reduction approach for regression testing using an optimization algorithm called BAT algorithm [27]. This algorithm aims to overcome the challenges discussed above and reduce the test suite optimally without compromising the test requirements.

3-1 Representation Of Test Pool

A test case is a set of instructions which process input variables required by the software to produce desired results and test case requirements a specific software function, loop or branch. Here, the test case requirement is branch coverage and the output provided specifies whether the given test case covers the specific branch or not. Let us assume that the number of test case for the algorithm is \(d\) and number of test requirement is \(m\). Then, test pool can be represented as, \(P = \{c_{ij}; 0 < i < d; 0 < j < m\}\). \(c_{ij}\) may be zero or one based on the requirements satisfied by the test cases. Every value in \(P\) signifies whether the corresponding test case can satisfy the corresponding test requirement. The cost vector for each test case \(c_i\) is computed by finding the execution time of the test case. So, the cost vector for all the test cases can be indicated as, \(C = \{y_i; 0 < i < d\}\)

3-2 Test Suite Reduction

The test suite reduction is carried out in this paper that satisfies two constraints, such as, i) satisfying all test requirements, ii) Minimizing the cost value. Let \(PR\) be the selected test suite and \(x\) be the number of test cases removed. Then, the test suite reduction problem with cost minimization is formulated as the following objective

\[ PR = \{ckj; 0 < k < d-x; 0 < j < m\}\]

3-3 Pbat Search Algorithm

Initialization: Let us assume that \(n\) bats are randomly initialized their positions within the search space as, \(bp=bp1, bp2, ..., bpq\) where \(p=1,2, ..., n\) and \(q\) is the dimension of the solution which signifies the number of test cases taken for optimization. The variables such as, loudness \(A\), pulse rate \(r\), iteration \(t\), minimum frequency, maximum frequency \(Qmax\) and velocity \(Vi\) are initialized.

Evaluation: Every bat is then evaluated with fitness function and the best one having minimum fitness is stored as, \(xb\)
Where, is a random value which is used to update the frequency of the bat using $Q_{\min}$ and $Q_{\max}$. $\alpha$ ranges between -1 to 1. The frequency $Q_i$ is then utilized to update the velocity of the bats ($v_i^t$) using the best position of the bats $x_{b}$. Accordingly, the above equation of velocity can be written as, $v_i^t = v_{i}^{t-1} - \alpha Q_i (x_i^t - x_{b}) + \beta (U - x_{b}^t Q_i)$

Figure 1 means that the test cases selected through this solution encoding procedure is 1, 3, 4 and 6. In PBAT algorithm, bat population is represented as, $b_{pq}: 0 \leq n \leq 0 \leq d$. Here, $n$ is the number of bats considered and $d$ is the dimension of the solution or number of test cases.

**Fitness evaluation**: The fitness of every bat (solution) is evaluated using the fitness function, $F(b_p)$. This function computes the total cost of the selected test cases through the solution $b_p$ only if the selected test cases can satisfy the entire test requirement.

$$F(b_p) = \begin{cases} 0, & \text{if } S = m \\ \sum_{j=1}^{d} z_j - \sum_{k=1}^{d} c_{kj} > 0, & \text{otherwise} \end{cases}$$

**Algorithm**: PBAT
1. **Input**: $P$ → Test pool
2. $C_T$ → Cost Factor
3. **Output**: $x_0$ → best solution (Selected test cases)
4. **Begin**
5. Initialize variables such as $A$, $r$, $t$, $Q_{\min}$, $Q_{\max}$, $\alpha$, $\beta$
6. $p = 1$
7. While $p < t$
8. $b_{pq}$ → best solution
9. $x_{i}^t$ → Update frequency $Q_i$
10. $V_i^t$ → Update velocity $V_i^t$
11. Find fitness for $b_p$ using $P$ and $C_T$
12. Generate local solution around best solution
13. EndWhile
14. $x_{i}^0$ → Generate random solution
15. Find fitness of $x_{i}^0$ using $P$ and $C_T$

DOI: 10.9790/0661-2001010108 www.iosrjournals.org
PBAT algorithm: The input for the PBAT algorithm is test pool $P$ and cost vector $CT$. Table 1 shows the inputs test pool $P$ and cost vector $CT$ of the PBAT algorithm. The variables are initialized as, $n=5$, $t=2$; $A=0.5$, $r=0.5$, $Q_{\text{min}}=0$; $Q_{\text{max}}=1$; $d=7$. Table 3 shows the initialization of $x$, $Q$ and $v$. In the first iteration, for $\alpha=-0.35$, $\alpha=0.8$, $\beta=0.2$, frequency, velocity and position values are updated which is shown in Table 5. Then, fitness is computed for every solution of $x$: $\text{Fitness}(x(1))=\infty$; $\text{Fitness}(x(2))=\infty$; $\text{Fitness}(x(3))=\infty$; $\text{Fitness}(x(4))=\infty$; $\text{Fitness}(x(5))=\infty$. We obtained no solutions which satisfy the entire test requirement. So, again, $x(1)$ is taken as the best solution and is given in Table 6 at the end of first iteration. In the second iteration, for $\alpha=0.35$, $\alpha=0.8$, $\beta=0.2$, frequency, velocity and position values are updated as shown in Table 7. Then, fitness is computed for the new solution of $x$ values: $\text{Fitness}(x(1))=\infty$; $\text{Fitness}(x(2))=47$; $\text{Fitness}(x(3))=\infty$; $\text{Fitness}(x(4))=\infty$; $\text{Fitness}(x(5))=\infty$. At the end of second iteration, the minimum fitness is obtained for $x(2)$. So, we selected $x(2)$ as the best solution, shown in Table 8. After finishing two iterations, the selected test cases through best solution are $(c1j, c2j, c4j, c5j, c7j)$ and requirements solved are $(c11, c12, c13, c14, c15, c16, c17)$. The total cost required is $47 (1+2+11+23+10)$ which is obtained by doing the summation of all the cost values of selected test cases.

<table>
<thead>
<tr>
<th>Table 2: Test Pool $P$ and Cost Vector $CT$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{1j}$</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 3: Initialization of $X$, $Q$ and $V$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

| $Q$ | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| $v$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

<table>
<thead>
<tr>
<th>Table 4: Best Solution From Initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_0$</td>
</tr>
</tbody>
</table>

DOI: 10.9790/0661-2001010108 www.iosrjournals.org 4 | Page
### III. Results And Analysis

5-1 Experimental Setup

The proposed PBAT algorithm is implemented using Java 1.7 with NetBeans IDE 7.3. The experimentation is conducted on Windows 7 machines with Intel Core Duo processors and 2GB of memory. At first, required no of test cases are generated randomly through a synthetic program. Once we generate test cases for a subject program through synthetic program, branch coverage and cost is computed by applying test case to the corresponding subject program.

5-2 Performance Evaluation

Figure 3 presents the SCR graph for various values of minimum frequency. When the minimum frequency is increased from 0 to 0.4, the value of the median program is found to decrease from 85.33 to 71.81.
The value of the PBAT using SCR for the elevator programs is 90.37, 92.69, 93.28, 94.64, and 92.99 when the minimum frequency value increases as 0, 0.1, 0.2, 0.3, and 0.4 respectively. Likewise, the value of the trityp programs using the proposed TBAT and SCR is 9.66, 78.09, 87.22, 10.94, and 90.46 respectively with the increasing minimum frequency value from 0 to 0.4. Figure 4 shows the performance of PBAT using SCR for various values of maximum frequency. The maximum frequency is varied as 0.6, 0.7, 0.8, 0.9, and 1 for all the programs like the median, elevator, trityp, Apollo, and pool3. The value of the median program using the TBAT with the SCR reaches 84.34 from 91.31, elevator programs reaches 93.36 from 99.94, trityp attains 85.58 from 99.93, Apollo program attains 40.33 from 99.99, and pool3 attains 99.69 from 99.99. Figure 5 shows the performance of TBAT using SCR for various values of loudness. The value of loudness used for analysis is 0.2, 0.4, 0.6, 0.8, and 1 respectively. The SCR value of the median program when the loudness is 0.2 is 93.45, 76.84 for 0.4, 48.66 for 0.6, 27.27 for 0.8, and 70.33 for 1 as loudness.

**Figure 3** Performance of PBAT using SCR, a) for various numbers of bats b) for various minimum frequency

**Figure 4** Performance of PBAT using SCR, a) for various maximum frequency b) for various pulse rate
Optimization Algorithm for Cost-Aware Test Suite Minimization In Software Testing

Figure 5 Performance of PBAT using SCR, a) for various loudness b) for various iterations

5-3 Analysis

Test pool is directly given to the algorithms, PBAT, Systolic Genetic Search [28] and Greedy/Unreplaceability. The ultimate aim of these algorithms is to select test cases which should satisfy all the test requirements. Accordingly, the test suite is reduced by both the algorithms and the cost for all the selected test cases are computed and shown in Table 9. The total cost for the proposed PBAT algorithm is 30.32 msec for median program as compared to the value of 66.7 for the existing algorithm. PBAT achieved 89.2% improvement in the variance of 1% as compared with the existing algorithm which improves only 76.2% in the variance of 5%. The proposed PBAT obtained SCR values of 93.7%, 99.44%, and 99.5% for trityp, apollo and pool3 programs.

<table>
<thead>
<tr>
<th>Program</th>
<th>Original</th>
<th>PBAT Cost</th>
<th>Greedy/Unreplaceability Cost</th>
<th>Systolic Genetic Cost</th>
<th>PBA T-SCR</th>
<th>Greedy/Unreplaceability-SCR</th>
<th>Systolic Genetic-SCR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Median</td>
<td>280.826</td>
<td>30.32 ± 0.3</td>
<td>66716.7 ± 0.7</td>
<td>33.1 ± 0.45</td>
<td>89.2 ± 0.1</td>
<td>76.24 ± 0.7</td>
<td>92.24 ± 0.2</td>
</tr>
<tr>
<td>Elevator</td>
<td>16215 ± 6.3</td>
<td>10141 ± 3.210</td>
<td>15088.77 ± 256</td>
<td>12568 ± 451</td>
<td>93.74 ± 0.1</td>
<td>90.69 ± 0.7</td>
<td>99.44 ± 0.2</td>
</tr>
<tr>
<td>trityp</td>
<td>61145 ± 42</td>
<td>3822.9 ± 400</td>
<td>3721.18 ± 145</td>
<td>4122 ± 45</td>
<td>90.64</td>
<td>93.62</td>
<td>99.25 ± 3</td>
</tr>
<tr>
<td>Apollo</td>
<td>726283 ± 3</td>
<td>4652.5 ± 539</td>
<td>649642.95 ± 452</td>
<td>41563 ± 245</td>
<td>99.44 ± 3</td>
<td>91.55 ± 0.1</td>
<td>99.42 ± 01</td>
</tr>
<tr>
<td>Pool3</td>
<td>1.17E+1 ± 5.8</td>
<td>8E-08 ± 345</td>
<td>375181608 ± 5689</td>
<td>6.1E-08 ± 345</td>
<td>99.50 ± 0.25</td>
<td>99.68 ± 03</td>
<td>99.47 ± 01</td>
</tr>
</tbody>
</table>

Table 9 Reduction capability of algorithms (in msec) for input threshold of 0.75

IV. Conclusion

PBAT algorithm is to minimize the cost of regression testing. This algorithm was developed to handle the diversity problem in generating new movements of bats to reach the optimal solution easily. The minimization function developed here to improve the speed of convergence contains two constraints, satisfying the entire test requirement and minimizing cost measure. In the proposed PBAT algorithm, initial solutions are generated randomly and fitness is evaluated using the proposed minimization function. The generation of the new solution set is done by the proposed formula to reach the minimum cost function faster than greedy-based algorithms. The performance of the proposed PBAT algorithm is extensively analyzed with different parametric values to understand the best parameters of the proposed algorithm in test suite reduction.
Optimization Algorithm for Cost-Aware Test Suite Minimization In Software Testing

References

[27]. ShounakRushikeshSugave.MIT College of Engineering, Pune SuhasHaribhauPatil,BharatiVidyapeeth